

SIMATIC

WinCC WinCC Professional V14 SP1 - Programming reference

System Manual

Online help printout

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.
 WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.
 CAUTION
indicates that minor personal injury can result if proper precautions are not taken.
NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

 WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1	VB scripting.....	23
1.1	System functions (Panels, Comfort Panels, RT Advanced).....	23
1.1.1	AcknowledgeAlarm (Panels, Comfort Panels, RT Advanced).....	23
1.1.2	ActivatePLCCodeViewer (Panels, Comfort Panels, RT Advanced).....	23
1.1.3	ActivatePreviousScreen (Panels, Comfort Panels, RT Advanced).....	25
1.1.4	ActivateScreen (Panels, Comfort Panels, RT Advanced).....	26
1.1.5	ActivateScreenByNumber (Panels, Comfort Panels, RT Advanced).....	27
1.1.6	ActivateSystemDiagnosticsView (Panels, Comfort Panels, RT Advanced).....	28
1.1.7	ArchiveLogFile (Panels, Comfort Panels, RT Advanced).....	29
1.1.8	Back up RAM file system (Panels, Comfort Panels, RT Advanced).....	30
1.1.9	CalibrateTouchScreen (Panels, Comfort Panels, RT Advanced).....	31
1.1.10	ChangeConnection (Panels, Comfort Panels, RT Advanced).....	32
1.1.11	ChangeConnectionEIP (Panels, Comfort Panels, RT Advanced).....	33
1.1.12	ClearAlarmBuffer (Panels, Comfort Panels, RT Advanced).....	35
1.1.13	ClearAlarmBufferProtocol (Panels, Comfort Panels, RT Advanced).....	36
1.1.14	ClearDataRecord (Panels, Comfort Panels, RT Advanced).....	37
1.1.15	ClearDataRecordMemory (Panels, Comfort Panels, RT Advanced).....	38
1.1.16	ClearLog (Panels, Comfort Panels, RT Advanced).....	39
1.1.17	CloseAllLogs (Panels, Comfort Panels, RT Advanced).....	40
1.1.18	ControlSmartServer (Panels, Comfort Panels, RT Advanced).....	40
1.1.19	ControlWebServer (Panels, Comfort Panels, RT Advanced).....	41
1.1.20	CopyLog (Panels, Comfort Panels, RT Advanced).....	42
1.1.21	DecreaseTag (Panels, Comfort Panels, RT Advanced).....	43
1.1.22	EditAlarm (Panels, Comfort Panels, RT Advanced).....	44
1.1.23	Encode (Panels, Comfort Panels, RT Advanced).....	44
1.1.24	EncodeEx (Panels, Comfort Panels, RT Advanced).....	45
1.1.25	ExportDataRecords (Panels, Comfort Panels, RT Advanced).....	46
1.1.26	ExportDataRecordsWithChecksum (Panels, Comfort Panels, RT Advanced).....	48
1.1.27	ExportImportUserAdministration (Panels, Comfort Panels, RT Advanced).....	51
1.1.28	GetBrightness (Panels, Comfort Panels, RT Advanced).....	52
1.1.29	GetDataRecordFromPLC (Panels, Comfort Panels, RT Advanced).....	52
1.1.30	GetDataRecordName (Panels, Comfort Panels, RT Advanced).....	54
1.1.31	GetDataRecordTagsFromPLC (Panels, Comfort Panels, RT Advanced).....	56
1.1.32	GetGroupNumber (Panels, Comfort Panels, RT Advanced).....	56
1.1.33	GetPassword (Panels, Comfort Panels, RT Advanced).....	57
1.1.34	GetUserName (Panels, Comfort Panels, RT Advanced).....	58
1.1.35	GoToEnd (Panels, Comfort Panels, RT Advanced).....	58
1.1.36	GoToHome (Panels, Comfort Panels, RT Advanced).....	59
1.1.37	ImportDataRecords (Panels, Comfort Panels, RT Advanced).....	59
1.1.38	ImportDataRecordsWithChecksum (Panels, Comfort Panels, RT Advanced).....	61
1.1.39	IncreaseTag (Panels, Comfort Panels, RT Advanced).....	63
1.1.40	InverseLinearScaling (Panels, Comfort Panels, RT Advanced).....	63
1.1.41	InvertBit (Panels, Comfort Panels, RT Advanced).....	65
1.1.42	InvertBitInTag (Panels, Comfort Panels, RT Advanced).....	66
1.1.43	LinearScaling (Panels, Comfort Panels, RT Advanced).....	68
1.1.44	LoadDataRecord (Panels, Comfort Panels, RT Advanced).....	70

1.1.45	LogOff (Panels, Comfort Panels, RT Advanced).....	71
1.1.46	Logon (Panels, Comfort Panels, RT Advanced).....	71
1.1.47	LookupText (Panels, Comfort Panels, RT Advanced).....	72
1.1.48	NotifyUserAction (Panels, Comfort Panels, RT Advanced).....	73
1.1.49	OpenAllLogs (Panels, Comfort Panels, RT Advanced).....	74
1.1.50	OpenCommandPrompt (Panels, Comfort Panels, RT Advanced).....	75
1.1.51	OpenControlPanel (Panels, Comfort Panels, RT Advanced).....	76
1.1.52	OpenInternetExplorer (Panels, Comfort Panels, RT Advanced).....	77
1.1.53	OpenScreenKeyboard (Panels, Comfort Panels, RT Advanced).....	77
1.1.54	OpenTaskManager (Panels, Comfort Panels, RT Advanced).....	78
1.1.55	PageDown (Panels, Comfort Panels, RT Advanced).....	79
1.1.56	PageUp (Panels, Comfort Panels, RT Advanced).....	79
1.1.57	PrintReport (Panels, Comfort Panels, RT Advanced).....	80
1.1.58	PrintScreen (Panels, Comfort Panels, RT Advanced).....	80
1.1.59	ResetBit (Panels, Comfort Panels, RT Advanced).....	81
1.1.60	ResetBitInTag (Panels, Comfort Panels, RT Advanced).....	82
1.1.61	SafelyRemoveHardware (Panels, Comfort Panels).....	84
1.1.62	SaveDataRecord (Panels, Comfort Panels, RT Advanced).....	85
1.1.63	SendEmail (Panels, Comfort Panels, RT Advanced).....	86
1.1.64	SetAcousticSignal (Panels, Comfort Panels, RT Advanced).....	87
1.1.65	SetAlarmReportMode (Panels, Comfort Panels, RT Advanced).....	88
1.1.66	SetAndGetBrightness (Panels, Comfort Panels, RT Advanced).....	89
1.1.67	SetBit (Panels, Comfort Panels, RT Advanced).....	89
1.1.68	SetBitInTag (Panels, Comfort Panels, RT Advanced).....	91
1.1.69	SetBrightness (Panels, Comfort Panels, RT Advanced).....	93
1.1.70	SetConnectionMode (Panels, Comfort Panels, RT Advanced).....	94
1.1.71	SetDataRecordTagsToPLC (Panels, Comfort Panels, RT Advanced).....	95
1.1.72	SetDataRecordToPLC (Panels, Comfort Panels, RT Advanced).....	96
1.1.73	SetDaylightSavingTime (Panels, Comfort Panels, RT Advanced).....	97
1.1.74	SetDeviceMode (Panels, Comfort Panels, RT Advanced).....	98
1.1.75	SetDisplayMode (Panels, Comfort Panels, RT Advanced).....	98
1.1.76	SetLanguage (Panels, Comfort Panels, RT Advanced).....	99
1.1.77	SetPLCDateTime (Panels, Comfort Panels, RT Advanced).....	100
1.1.78	SetRecipeTags (Panels, Comfort Panels, RT Advanced).....	101
1.1.79	SetScreenKeyboardMode (Panels, Comfort Panels, RT Advanced).....	102
1.1.80	SetTag (Panels, Comfort Panels, RT Advanced).....	103
1.1.81	ShiftAndMask (Panels, Comfort Panels, RT Advanced).....	104
1.1.82	ShowAlarmWindow (Panels, Comfort Panels, RT Advanced).....	106
1.1.83	ShowOperatorNotes (Panels, Comfort Panels, RT Advanced).....	107
1.1.84	ShowPopUpScreen (Panels, Comfort Panels, RT Advanced).....	108
1.1.85	ShowPopupScreenSizable (Panels, Comfort Panels, RT Advanced).....	109
1.1.86	ShowSlideInScreen (Panels, Comfort Panels, RT Advanced).....	111
1.1.87	ShowSoftwareVersion (Panels, Comfort Panels, RT Advanced).....	112
1.1.88	ShowSystemAlarm (Panels, Comfort Panels, RT Advanced).....	113
1.1.89	ShowSystemDiagnosticsWindow (Panels, Comfort Panels).....	113
1.1.90	StartLogging (Panels, Comfort Panels, RT Advanced).....	114
1.1.91	StartNextLog (Panels, Comfort Panels, RT Advanced).....	114
1.1.92	StartProgram (Panels, Comfort Panels, RT Advanced).....	115
1.1.93	StopLogging (Panels, Comfort Panels, RT Advanced).....	117
1.1.94	StopRuntime (Panels, Comfort Panels, RT Advanced).....	118
1.1.95	TerminatePROFIsafe (Panels, Comfort Panels, RT Advanced).....	119
1.1.96	WinACMPGetVersion (Panels, Comfort Panels).....	119

1.1.97	WinACMPSetStartAtBoot (Panels, Comfort Panels).....	120
1.1.98	WinACSetStartMode (Panels, Comfort Panels).....	120
1.2	System functions (RT Professional).....	121
1.2.1	ActivateScreen (RT Professional).....	121
1.2.2	ActivateScreenInScreenWindow (RT Professional).....	122
1.2.3	DecreaseTag (RT Professional).....	123
1.2.4	ExportImportUserAdministration (RT Professional).....	124
1.2.5	GetParentScreen (RT Professional).....	125
1.2.6	GetParentScreenWindow (RT Professional).....	127
1.2.7	IncreaseTag (RT Professional).....	128
1.2.8	InverseLinearScaling (RT Professional).....	129
1.2.9	InvertBit (RT Professional).....	130
1.2.10	InvertBitInTag (RT Professional).....	132
1.2.11	LinearScaling (RT Professional).....	133
1.2.12	LookupText (RT Professional).....	135
1.2.13	ResetBit (RT Professional).....	136
1.2.14	ResetBitInTag (RT Professional).....	137
1.2.15	SetBit (RT Professional).....	139
1.2.16	SetBitInTag (RT Professional).....	141
1.2.17	SetLanguage.....	143
1.2.18	SetPropertyByConstant (RT Professional).....	143
1.2.19	SetPropertyByProperty (RT Professional).....	145
1.2.20	SetPropertyByTag (RT Professional).....	147
1.2.21	SetPropertyByTagIndirect (RT Professional).....	149
1.2.22	SetTag (RT Professional).....	151
1.2.23	SetTagByProperty (RT Professional).....	152
1.2.24	SetTagByTagIndirect (RT Professional).....	153
1.2.25	SetTagIndirect (RT Professional).....	154
1.2.26	SetTagIndirectByProperty (RT Professional).....	155
1.2.27	SetTagIndirectByTagIndirect (RT Professional).....	156
1.2.28	SetTagWithOperatorEvent (RT Professional).....	157
1.2.29	ShowBlockInTIAPortalFromAlarm (RT Professional).....	158
1.2.30	ShowLogonDialog (RT Professional).....	160
1.2.31	ShowPLCCodeViewFromAlarm (RT Professional).....	160
1.2.32	StopRuntime (RT Professional).....	161
1.3	VBScript for Windows (Panels, Comfort Panels, RT Advanced, RT Professional).....	162
1.3.1	VBScript for Windows (Panels, Comfort Panels, RT Advanced, RT Professional).....	162
1.4	VBScript for Windows CE (Panels, Comfort Panels).....	163
1.4.1	VBScript for Windows CE (Panels, Comfort Panels).....	163
1.4.2	CreateObject (Panels, Comfort Panels).....	163
1.4.3	Control element (Panels, Comfort Panels).....	164
1.4.4	Properties (Panels, Comfort Panels).....	166
1.4.4.1	Attr (Panels, Comfort Panels).....	166
1.4.4.2	EOF (Panels, Comfort Panels).....	166
1.4.4.3	Loc (Panels, Comfort Panels).....	167
1.4.4.4	LOF (Panels, Comfort Panels).....	168
1.4.4.5	Seek (Panels, Comfort Panels).....	168
1.4.5	Methods (Panels, Comfort Panels).....	169
1.4.5.1	Close (Panels, Comfort Panels).....	169
1.4.5.2	Dir (Panels, Comfort Panels).....	169
1.4.5.3	FileCopy (Panels, Comfort Panels).....	170

1.4.5.4	FileDateTime (Panels, Comfort Panels).....	171
1.4.5.5	FileLen (Panels, Comfort Panels).....	172
1.4.5.6	Get (Panels, Comfort Panels).....	172
1.4.5.7	GetAttr (Panels, Comfort Panels).....	174
1.4.5.8	Input (Panels, Comfort Panels).....	175
1.4.5.9	InputB (Panels, Comfort Panels).....	175
1.4.5.10	InputFields (Panels, Comfort Panels).....	176
1.4.5.11	Kill (Panels, Comfort Panels).....	177
1.4.5.12	LineInputString (Panels, Comfort Panels).....	178
1.4.5.13	LinePrint (Panels, Comfort Panels).....	178
1.4.5.14	MkDir (Panels, Comfort Panels).....	179
1.4.5.15	MoveFile (Panels, Comfort Panels).....	180
1.4.5.16	Open (Panels, Comfort Panels).....	180
1.4.5.17	Put (Panels, Comfort Panels).....	181
1.4.5.18	RmDir (Panels, Comfort Panels).....	182
1.4.5.19	SetAttr (Panels, Comfort Panels).....	183
1.4.5.20	WriteFields (Panels, Comfort Panels).....	184
1.5	VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional).....	186
1.5.1	VBS object model (RT Professional).....	186
1.5.2	Objects (Panels, Comfort Panels, RT Advanced).....	188
1.5.2.1	HMIRuntime (Panels, Comfort Panels, RT Advanced).....	188
1.5.2.2	Screen object (list) (Panels, Comfort Panels, RT Advanced).....	189
1.5.2.3	Screen (Panels, Comfort Panels, RT Advanced).....	191
1.5.2.4	ScreenItem (Panels, Comfort Panels, RT Advanced).....	192
1.5.2.5	ScreenItems (Panels, Comfort Panels, RT Advanced).....	194
1.5.2.6	SmartTags (Panels, Comfort Panels, RT Advanced).....	195
1.5.2.7	SmartTag (Panels, Comfort Panels, RT Advanced).....	197
1.5.3	Objects (RT Professional).....	199
1.5.3.1	Alarm (RT Professional).....	199
1.5.3.2	Alarms (list) (RT Professional).....	200
1.5.3.3	AlarmLogs (list) (RT Professional).....	201
1.5.3.4	DataItem (RT Professional).....	203
1.5.3.5	DataLogs (list) (RT Professional).....	204
1.5.3.6	DataSet (list) (RT Professional).....	206
1.5.3.7	HMIRuntime (RT Professional).....	208
1.5.3.8	Item (RT Professional).....	210
1.5.3.9	Layer (RT Professional).....	210
1.5.3.10	Layers (list) (RT Professional).....	212
1.5.3.11	Logging (RT Professional).....	213
1.5.3.12	Project (RT Professional).....	214
1.5.3.13	Screen (RT Professional).....	215
1.5.3.14	ScreenItem (RT Professional).....	218
1.5.3.15	ScreenItems (list) (RT Professional).....	220
1.5.3.16	Screen object (list) (RT Professional).....	222
1.5.3.17	SmartTag (RT Professional).....	224
1.5.3.18	SmartTags (RT Professional).....	225
1.5.3.19	Tag (RT Professional).....	228
1.5.3.20	Tags (list) (RT Professional).....	231
1.5.3.21	TagSet (list) (RT Professional).....	232
1.5.4	Object types (Panels, Comfort Panels, RT Advanced, RT Professional).....	233
1.5.4.1	Object types available in VBS (Panels, Comfort Panels, RT Advanced, RT Professional).....	233
1.5.4.2	Objects A-I (Panels, Comfort Panels, RT Advanced, RT Professional).....	236

1.5.4.3	Objects K-Z (Panels, Comfort Panels, RT Advanced, RT Professional).....	338
1.5.5	Properties (Panels, Comfort Panels, RT Advanced, RT Professional).....	494
1.5.5.1	Properties A (Panels, Comfort Panels, RT Advanced, RT Professional).....	494
1.5.5.2	Properties B (Panels, Comfort Panels, RT Advanced, RT Professional).....	530
1.5.5.3	Properties C (Panels, Comfort Panels, RT Advanced, RT Professional).....	591
1.5.5.4	Properties D (Panels, Comfort Panels, RT Advanced, RT Professional).....	644
1.5.5.5	Properties E-F (Panels, Comfort Panels, RT Advanced, RT Professional).....	661
1.5.5.6	Properties G-H (Panels, Comfort Panels, RT Advanced, RT Professional).....	720
1.5.5.7	Properties I-J (Panels, Comfort Panels, RT Advanced, RT Professional).....	744
1.5.5.8	Properties K-L (Panels, Comfort Panels, RT Advanced, RT Professional).....	762
1.5.5.9	Properties M-N (Panels, Comfort Panels, RT Advanced, RT Professional).....	801
1.5.5.10	Properties O-P (Panels, Comfort Panels, RT Advanced, RT Professional).....	841
1.5.5.11	Properties Q-R (Panels, Comfort Panels, RT Advanced, RT Professional).....	879
1.5.5.12	Properties S (Panels, Comfort Panels, RT Advanced, RT Professional).....	896
1.5.5.13	Properties T (Panels, Comfort Panels, RT Advanced, RT Professional).....	988
1.5.5.14	Properties U-W (Panels, Comfort Panels, RT Advanced, RT Professional).....	1146
1.5.5.15	Properties X-Z (Panels, Comfort Panels, RT Advanced, RT Professional).....	1229
1.5.6	Methods (Panels, Comfort Panels, RT Advanced, RT Professional).....	1254
1.5.6.1	Methods A-G (Panels, Comfort Panels, RT Advanced, RT Professional).....	1254
1.5.6.2	Methods H-R (Panels, Comfort Panels, RT Advanced, RT Professional).....	1318
1.5.6.3	Methods S-Z (Panels, Comfort Panels, RT Advanced, RT Professional).....	1346
2	C scripting (RT Professional).....	1375
2.1	System functions (RT Professional).....	1375
2.1.1	ActivateNextScreen (RT Professional).....	1375
2.1.2	ActivatePreviousScreen (RT Professional).....	1376
2.1.3	ActivateScreen (RT Professional).....	1377
2.1.4	ActivateScreenInScreenWindow (RT Professional).....	1378
2.1.5	ActivateStartScreen (RT Professional).....	1379
2.1.6	ActivateStoredScreen (RT Professional).....	1380
2.1.7	DecreaseTag (RT Professional).....	1381
2.1.8	GetLocalScreen (RT Professional).....	1383
2.1.9	GetLinkedTag (RT Professional).....	1385
2.1.10	GetLanguageByLocaleID (RT Professional).....	1386
2.1.11	GetParentScreen (RT Professional).....	1388
2.1.12	GetParentScreenWindow (RT Professional).....	1390
2.1.13	GetProp (RT Professional).....	1392
2.1.13.1	GetPropBOOL (RT Professional).....	1392
2.1.13.2	GetPropChar (RT Professional).....	1393
2.1.13.3	GetPropDouble (RT Professional).....	1395
2.1.13.4	GetPropLong (RT Professional).....	1396
2.1.14	GetServerTagPrefix (RT Professional).....	1397
2.1.15	GetTag (RT Professional).....	1400
2.1.15.1	GetTag functions (RT Professional).....	1400
2.1.15.2	GetTagDateTime function (RT Professional).....	1401
2.1.15.3	GetTagMultiStateQCWait functions (RT Professional).....	1402
2.1.15.4	GetTagMultiStateWait functions (RT Professional).....	1403
2.1.15.5	GetTagMultiWait functions (RT Professional).....	1405
2.1.15.6	GetTagState functions (RT Professional).....	1406
2.1.15.7	GetTagStateQC functions (RT Professional).....	1409
2.1.15.8	GetTagStateQCWait functions (RT Professional).....	1411
2.1.15.9	GetTagStateWait functions (RT Professional).....	1414

2.1.15.10	GetTagValue functions (RT Professional).....	1416
2.1.15.11	GetTagValueStateQC functions (RT Professional).....	1418
2.1.15.12	GetTagValueStateQCWait functions (RT Professional).....	1420
2.1.15.13	GetTagValueWait functions (RT Professional).....	1421
2.1.15.14	GetTagWait functions (RT Professional).....	1422
2.1.16	IncreaseTag (RT Professional).....	1424
2.1.17	InquireLanguage (RT Professional).....	1425
2.1.18	InverseLinearScaling (RT Professional).....	1427
2.1.19	InvertBit (RT Professional).....	1429
2.1.20	InvertBitInTag (RT Professional).....	1430
2.1.21	IsUserAuthorized (RT Professional).....	1432
2.1.22	LinearScaling (RT Professional).....	1434
2.1.23	ReportJob (RT Professional).....	1435
2.1.24	ResetBit (RT Professional).....	1436
2.1.25	ResetBitInTag (RT Professional).....	1438
2.1.26	Set_Focus (RT Professional).....	1439
2.1.27	SetBit (RT Professional).....	1440
2.1.28	SetBitInTag (RT Professional).....	1441
2.1.29	SetLanguageByLocaleID (RT Professional).....	1443
2.1.30	SetProp (RT Professional).....	1445
2.1.30.1	SetPropBOOL (RT Professional).....	1445
2.1.30.2	SetPropChar (RT Professional).....	1447
2.1.30.3	SetPropDouble (RT Professional).....	1448
2.1.30.4	SetPropLong (RT Professional).....	1450
2.1.31	SetPropertyByConstant (RT Professional).....	1451
2.1.32	SetPropertyByProperty (RT Professional).....	1453
2.1.33	SetPropertyByTag (RT Professional).....	1455
2.1.34	SetPropertyByTagIndirect (RT Professional).....	1457
2.1.35	SetTag (RT Professional).....	1458
2.1.35.1	SetTag functions (RT Professional).....	1458
2.1.35.2	SetTagDateTime (RT Professional).....	1461
2.1.35.3	SetTagMultiStateWait functions (RT Professional).....	1462
2.1.35.4	SetTagMultiWait functions (RT Professional).....	1463
2.1.35.5	SetTagState functions (RT Professional).....	1466
2.1.35.6	SetTagStateWait functions (RT Professional).....	1468
2.1.35.7	SetTagValue functions (RT Professional).....	1471
2.1.35.8	SetTagValueWait functions (RT Professional).....	1472
2.1.35.9	SetTagWait functions (RT Professional).....	1474
2.1.35.10	SetTag (RT Professional).....	1476
2.1.35.11	SetTagByProperty (RT Professional).....	1477
2.1.35.12	SetTagByTagIndirect (RT Professional).....	1479
2.1.35.13	SetTagIndirect (RT Professional).....	1480
2.1.35.14	SetTagIndirectByProperty (RT Professional).....	1481
2.1.35.15	SetTagIndirectByTagIndirect (RT Professional).....	1482
2.1.35.16	SetTagIndirectWithOperatorEvent (RT Professional).....	1483
2.1.35.17	SetTagWithOperatorEvent (RT Professional).....	1483
2.1.36	StartProgram (RT Professional).....	1484
2.1.37	StopRuntime (RT Professional).....	1485
2.1.38	StoreScreen (RT Professional).....	1486
2.1.39	TriggerOperatorEvent (RT Professional).....	1487
2.1.40	UA (Recipe) (RT Professional).....	1488
2.1.40.1	uaArchiveClose (RT Professional).....	1488

2.1.40.2	uaArchiveDelete (RT Professional).....	1489
2.1.40.3	uaArchiveExport (RT Professional).....	1490
2.1.40.4	uaArchiveGetCount (RT Professional).....	1491
2.1.40.5	uaArchiveGetFieldLength (RT Professional).....	1491
2.1.40.6	uaArchiveGetFieldName (RT Professional).....	1492
2.1.40.7	uaArchiveGetFields (RT Professional).....	1493
2.1.40.8	uaArchiveGetFieldType (RT Professional).....	1493
2.1.40.9	uaArchiveGetFieldValueDate (RT Professional).....	1494
2.1.40.10	uaArchiveGetFieldValueDouble (RT Professional).....	1495
2.1.40.11	uaArchiveGetFieldValueFloat (RT Professional).....	1496
2.1.40.12	uaArchiveGetFieldValueLong (RT Professional).....	1496
2.1.40.13	uaArchiveGetFieldValueString (RT Professional).....	1497
2.1.40.14	uaArchiveGetFilter (RT Professional).....	1498
2.1.40.15	uaArchiveGetID (RT Professional).....	1499
2.1.40.16	uaArchiveGetName (RT Professional).....	1499
2.1.40.17	uaArchiveGetSor (RT Professional).....	1500
2.1.40.18	uaArchiveImport (RT Professional).....	1501
2.1.40.19	uaArchiveInsert (RT Professional).....	1502
2.1.40.20	uaArchiveMoveFirst (RT Professional).....	1502
2.1.40.21	uaArchiveMoveLast (RT Professional).....	1503
2.1.40.22	uaArchiveMoveNext (RT Professional).....	1503
2.1.40.23	uaArchiveMovePrevious (RT Professional).....	1504
2.1.40.24	uaArchiveOpen (RT Professional).....	1505
2.1.40.25	uaArchiveReadTagValues (RT Professional).....	1505
2.1.40.26	uaArchiveReadTagValuesByName (RT Professional).....	1506
2.1.40.27	uaArchiveRequery (RT Professional).....	1507
2.1.40.28	uaArchiveSetFieldValueDate (RT Professional).....	1508
2.1.40.29	uaArchiveSetFieldValueDouble (RT Professional).....	1509
2.1.40.30	uaArchiveSetFieldValueFloat (RT Professional).....	1510
2.1.40.31	uaArchiveSetFieldValueLong (RT Professional).....	1510
2.1.40.32	uaArchiveSetFieldValueString (RT Professional).....	1511
2.1.40.33	uaArchiveSetFilter (RT Professional).....	1512
2.1.40.34	uaArchiveSetSort (RT Professional).....	1513
2.1.40.35	uaArchiveUpdate (RT Professional).....	1514
2.1.40.36	uaArchiveWriteTagValues (RT Professional).....	1514
2.1.40.37	uaArchiveWriteTagValuesByName (RT Professional).....	1515
2.1.40.38	uaConnect (RT Professional).....	1516
2.1.40.39	uaDisconnect (RT Professional).....	1517
2.1.40.40	uaGetArchive (RT Professional).....	1517
2.1.40.41	uaGetField (RT Professional).....	1518
2.1.40.42	uaGetLastError (RT Professional).....	1519
2.1.40.43	uaGetLastHResult (RT Professional).....	1521
2.1.40.44	uaGetNumArchives (RT Professional).....	1521
2.1.40.45	uaGetNumFields (RT Professional).....	1522
2.1.40.46	uaQueryArchive (RT Professional).....	1522
2.1.40.47	uaQueryArchiveByName (RT Professional).....	1523
2.1.40.48	UaQueryConfiguration (RT Professional).....	1524
2.1.40.49	uaReleaseArchive (RT Professional).....	1525
2.1.40.50	uaReleaseConfiguration (RT Professional).....	1525
2.2	C-bib (RT Professional).....	1526
2.2.1	ctype functions (RT Professional).....	1526
2.2.2	Function group c_bib (RT Professional).....	1527

2.2.3	math functions (RT Professional).....	1528
2.2.4	memory functions (RT Professional).....	1529
2.2.5	multibyte functions (RT Professional).....	1529
2.2.6	stdio functions (RT Professional).....	1530
2.2.7	stdlib functions (RT Professional).....	1531
2.2.8	string functions (RT Professional).....	1532
2.2.9	time functions (RT Professional).....	1533
2.3	Structure definition (RT Professional).....	1534
2.3.1	Structure definition CCAPErrExecute (RT Professional).....	1534
2.3.2	Structure definition CCAPTime (RT Professional).....	1535
2.3.3	Structure definition CMN_ERROR (RT Professional).....	1536
2.3.4	Structure definition DM_TYPEREF (RT Professional).....	1537
2.3.5	Structure definition DM_VAR_UPDATE_STRUCT (RT Professional).....	1538
2.3.6	Structure definition DM_VAR_UPDATE_STRUCTEX (RT Professional).....	1539
2.3.7	Structure definition DM_VARKEY (RT Professional).....	1540
2.3.8	Structure definition LINKINFO (RT Professional).....	1540
2.3.9	Structure definition MSG_FILTER_STRUCT (RT Professional).....	1542
2.3.10	Structure definition MSG_RTDATA_STRUCT (RT Professional).....	1545
3	Runtime API (RT Professional).....	1547
3.1	Runtime API (RT Professional).....	1547
3.2	Data management functions (RT Professional).....	1548
3.2.1	Basics (RT Professional).....	1548
3.2.1.1	Overview of structures (RT Professional).....	1548
3.2.1.2	Overview of the functions (RT Professional).....	1549
3.2.1.3	Quality codes of HMI tags (RT Professional).....	1549
3.2.1.4	Constants (RT Professional).....	1554
3.2.1.5	Error messages (RT Professional).....	1558
3.2.1.6	Conversion routines (Control Center) (RT Professional).....	1559
3.2.2	Structures (RT Professional).....	1565
3.2.2.1	DM_CONNECTION_DATA (RT Professional).....	1565
3.2.2.2	DM_CONNKEY (RT Professional).....	1566
3.2.2.3	DM_CYCLE_INFO (RT Professional).....	1567
3.2.2.4	DM_DATA_SERVICE (RT Professional).....	1568
3.2.2.5	DM_DIRECTORY_INFO (RT Professional).....	1569
3.2.2.6	DM_DLGOPTIONS (RT Professional).....	1570
3.2.2.7	DM_FORMAT_INFO (RT Professional).....	1571
3.2.2.8	DM_MACHINE_TABLE (RT Professional).....	1572
3.2.2.9	DM_PROJECT_INFO (RT Professional).....	1573
3.2.2.10	DM_SEND_DATA_STRUCT (RT Professional).....	1574
3.2.2.11	DM_SD_TARGET_MACHINE (RT Professional).....	1576
3.2.2.12	DM_SD_TARGET_APP (RT Professional).....	1577
3.2.2.13	DM_TYPEREF (RT Professional).....	1578
3.2.2.14	DM_VAR_UPDATE_STRUCT (RT Professional).....	1579
3.2.2.15	DM_VAR_UPDATE_STRUCTEX (RT Professional).....	1581
3.2.2.16	DM_VARFILTER (RT Professional).....	1584
3.2.2.17	DM_VARGRP_DATA (RT Professional).....	1586
3.2.2.18	DM_VARGRPKEY (RT Professional).....	1587
3.2.2.19	DM_VARIABLE_DATA (RT Professional).....	1588
3.2.2.20	DM_VARIABLE_DATA4 (RT Professional).....	1590
3.2.2.21	DM_VARKEY (RT Professional).....	1592

3.2.2.22	DM_VARLIMIT (RT Professional).....	1595
3.2.2.23	MCP_NEWVARIABLE_DATA (RT Professional).....	1596
3.2.2.24	MCP_NEWVARIABLE_DATA_4 (RT Professional).....	1598
3.2.2.25	MCP_NEWVARIABLE_DATA_5 (RT Professional).....	1600
3.2.2.26	MCP_NEWVARIABLE_DATA_EX (RT Professional).....	1602
3.2.2.27	MCP_NEWVARIABLE_DATA_EX4 (RT Professional).....	1604
3.2.2.28	MCP_VARIABLE_COMMON (RT Professional).....	1606
3.2.2.29	MCP_VARIABLE_COMMON_EX (RT Professional).....	1608
3.2.2.30	MCP_VARIABLE_LIMITS (RT Professional).....	1610
3.2.2.31	MCP_VARIABLE_LIMITS5 (RT Professional).....	1612
3.2.2.32	MCP_VARIABLE_LIMITS_EX (RT Professional).....	1614
3.2.2.33	MCP_VARIABLE_PROTOCOL (RT Professional).....	1615
3.2.2.34	MCP_VARIABLE_PROTOCOL_EX (RT Professional).....	1617
3.2.2.35	MCP_VARIABLE_SCALES (RT Professional).....	1618
3.2.3	General functions (RT Professional).....	1619
3.2.3.1	DMActivateRTProject (RT Professional).....	1619
3.2.3.2	DMAddNotify (RT Professional).....	1620
3.2.3.3	DMChangeDataLocale (RT Professional).....	1627
3.2.3.4	DMConnect (RT Professional).....	1628
3.2.3.5	DM_NOTIFY_PROC (RT Professional).....	1631
3.2.3.6	DMDeactivateRTProject (RT Professional).....	1633
3.2.3.7	DMDDisconnect (RT Professional).....	1634
3.2.3.8	DMEnumNumberFormats (RT Professional).....	1636
3.2.3.9	DM_ENUM_FORMATS_PROC (RT Professional).....	1637
3.2.3.10	DMEnumUpdateCycles (RT Professional).....	1639
3.2.3.11	DM_ENUM_CYCLES_PROC (RT Professional).....	1640
3.2.3.12	DMExitWinCC (RT Professional).....	1642
3.2.3.13	DMExitWinCCEx (RT Professional).....	1643
3.2.3.14	DMFireNotifyData (RT Professional).....	1644
3.2.3.15	DMGetConnectionState (RT Professional).....	1646
3.2.3.16	DMGetDataLocale (RT Professional).....	1647
3.2.3.17	DMGetHotkey (RT Professional).....	1648
3.2.3.18	DMGetMachineInfo (RT Professional).....	1649
3.2.3.19	DMGetMachineTable (RT Professional).....	1650
3.2.3.20	DMRemoveNotify (RT Professional).....	1651
3.2.4	Functions for project management (RT Professional).....	1654
3.2.4.1	DMEnumOpenedProjects (RT Professional).....	1654
3.2.4.2	DM_ENUM_OPENED_PROJECTS_PROC (RT Professional).....	1655
3.2.4.3	DMGetProjectDirectory (RT Professional).....	1657
3.2.4.4	DMGetProjectInformation (RT Professional).....	1658
3.2.4.5	DMGetRuntimeProject (RT Professional).....	1660
3.2.4.6	DMOpenProjectDocPlus (RT Professional).....	1661
3.2.4.7	DMOpenProjectPlus (RT Professional).....	1662
3.2.5	Data transport channels (RT Professional).....	1664
3.2.5.1	DMClearBlockQueue (RT Professional).....	1664
3.2.5.2	DMEnumDataServices (RT Professional).....	1665
3.2.5.3	DM_ENUM_DATA_SERVICE_PROC (RT Professional).....	1666
3.2.5.4	DM_DATA_SERVICE_PROC (RT Professional).....	1668
3.2.5.5	DMGetNumPendingBlocks (RT Professional).....	1669
3.2.5.6	DMInstallDataService (RT Professional).....	1670
3.2.5.7	DMSendApplicationData (RT Professional).....	1672
3.2.5.8	DMSetBlockQueueSize (RT Professional).....	1673

3.2.6	Functions for processing tags (RT Professional).....	1674
3.2.6.1	DMEnumVarData (RT Professional).....	1674
3.2.6.2	DM_ENUM_VARIABLE_PROC (RT Professional).....	1676
3.2.6.3	DMEnumVarData4 (RT Professional).....	1677
3.2.6.4	DM_ENUM_VARIABLE_PROC4 (RT Professional).....	1680
3.2.6.5	DMEnumVarGrpData (RT Professional).....	1681
3.2.6.6	DMEnumVarGrpDataExStr (RT Professional).....	1683
3.2.6.7	DM_ENUM_VARGRP_PROC (RT Professional).....	1685
3.2.6.8	DMEnumVariables (RT Professional).....	1686
3.2.6.9	DM_ENUM_VAR_PROC (RT Professional).....	1688
3.2.6.10	DMGetValue (RT Professional).....	1689
3.2.6.11	DMGetValueEx (RT Professional).....	1691
3.2.6.12	DMGetValueExStr (RT Professional).....	1693
3.2.6.13	DMGetValueWait (RT Professional).....	1702
3.2.6.14	DMGetValueWaitEx (RT Professional).....	1704
3.2.6.15	DMGetValueWaitExStr (RT Professional).....	1706
3.2.6.16	DMGetVarInfo (RT Professional).....	1708
3.2.6.17	DMGetVarInfoExStr (RT Professional).....	1710
3.2.6.18	DMGetVarLimits (RT Professional).....	1719
3.2.6.19	DMGetVarLimitsExStr (RT Professional).....	1721
3.2.6.20	DMGetVarType (RT Professional).....	1726
3.2.6.21	DMGetVarTypeExStr (RT Professional).....	1728
3.2.6.22	DMSetValue (RT Professional).....	1732
3.2.6.23	DMSetValueExStr (RT Professional).....	1735
3.2.6.24	DMSetValueMessage (RT Professional).....	1738
3.2.6.25	DMSetValueMessageExStr (RT Professional).....	1740
3.2.6.26	DMSetValueWait (RT Professional).....	1742
3.2.6.27	DMSetValueWaitExStr (RT Professional).....	1744
3.2.6.28	DMSetValueWaitMessage (RT Professional).....	1747
3.2.6.29	DMSetValueWaitMessageExStr (RT Professional).....	1749
3.2.6.30	DM_COMPLETION_PROC (RT Professional).....	1752
3.2.6.31	DMShowVarPropertiesExStr (RT Professional).....	1754
3.2.6.32	DMShowVarDatabase (RT Professional).....	1756
3.2.6.33	DMShowVarDatabaseExStr (RT Professional).....	1758
3.2.6.34	DMShowVarDatabaseMulti (RT Professional).....	1761
3.2.6.35	DMShowVarDatabaseMultiExStr (RT Professional).....	1764
3.2.6.36	DM_NOTIFY_SELECT_VAR_PROC (RT Professional).....	1767
3.2.6.37	GAPICreateNewVariable (RT Professional).....	1769
3.2.6.38	GAPICreateNewVariable4 (RT Professional).....	1770
3.2.6.39	GAPICreateNewVariable5 (RT Professional).....	1772
3.2.6.40	GAPICreateNewVariableEx4 (RT Professional).....	1773
3.2.7	Functions for processing structured tags (RT Professional).....	1775
3.2.7.1	GAPIEnumTypeMembers (RT Professional).....	1775
3.2.7.2	DM_ENUM_TPEMEMBERS_PROC (RT Professional).....	1777
3.2.7.3	GAPIEnumTypeMembersEx (RT Professional).....	1778
3.2.7.4	GAPIEnumTypeMembersExStr (RT Professional).....	1780
3.2.7.5	DM_ENUM_TPEMEMBERS_PROC_EX (RT Professional).....	1781
3.2.7.6	GAPIEnumTypeMembersEx4 (RT Professional).....	1783
3.2.7.7	DM_ENUM_TPEMEMBERS_PROC_EX4 (RT Professional).....	1784
3.2.7.8	GAPIEnumTypes (RT Professional).....	1786
3.2.7.9	DM_ENUM_TYPES_PROC (RT Professional).....	1787
3.2.8	Functions for processing connections (RT Professional).....	1789

3.2.8.1	DMEnumConnectionData (RT Professional).....	1789
3.2.8.2	DMEnumConnectionDataExStr (RT Professional).....	1791
3.2.8.3	DM_ENUM_CONNECTION_PROC (RT Professional).....	1792
3.2.9	Functions for the work platform (RT Professional).....	1793
3.2.9.1	DMGetOSVersion (RT Professional).....	1793
3.2.9.2	DMGetSystemLocale (RT Professional).....	1795
3.2.9.3	DMSetLanguage (RT Professional).....	1797
3.2.9.4	DMShowLanguageDialog (RT Professional).....	1798
3.2.10	Functions for updating tags (RT Professional).....	1799
3.2.10.1	DMBeginStartVarUpdate (RT Professional).....	1799
3.2.10.2	DMEndStartVarUpdate (RT Professional).....	1801
3.2.10.3	DMResumeVarUpdate (RT Professional).....	1802
3.2.10.4	DMStartVarUpdate (RT Professional).....	1804
3.2.10.5	DM_NOTIFY_VARIABLE_PROC (RT Professional).....	1806
3.2.10.6	DMStartVarUpdateEx (RT Professional).....	1807
3.2.10.7	DMStartVarUpdateExStr (RT Professional).....	1810
3.2.10.8	DM_NOTIFY_VARIABLEEX_PROC (RT Professional).....	1817
3.2.10.9	DMStopAllUpdates (RT Professional).....	1818
3.2.10.10	DMStopVarUpdate (RT Professional).....	1819
3.2.10.11	DMSuspendVarUpdate (RT Professional).....	1821
3.2.11	Examples (RT Professional).....	1823
3.2.11.1	Connection to DM (RT Professional).....	1823
3.2.11.2	Enum Data of Tags (RT Professional).....	1826
3.2.11.3	Enum open projects (RT Professional).....	1828
3.2.11.4	Enumerate all structured types (RT Professional).....	1830
3.2.11.5	Enumerate all connections (RT Professional).....	1832
3.2.11.6	Inquire project informations (RT Professional).....	1834
3.2.11.7	OnTestDeactivateRuntimeProject (RT Professional).....	1835
3.2.11.8	OnTestEnumGroupsAll (RT Professional).....	1835
3.2.11.9	OnTestEnumVariables (RT Professional).....	1836
3.2.11.10	OnTestEnumConnectionDataAll (RT Professional).....	1837
3.2.11.11	OnTestMachines (RT Professional).....	1838
3.2.11.12	OnTestProjectInfo (RT Professional).....	1839
3.2.11.13	OnTestProjectPaths (RT Professional).....	1840
3.2.11.14	OnTestOpenProject (RT Professional).....	1841
3.2.11.15	OnTestOpenProjects (RT Professional).....	1842
3.2.11.16	OnTestRuntimeProject (RT Professional).....	1842
3.2.11.17	OnTestSystemLocale (RT Professional).....	1843
3.2.11.18	OnTestUpdateCycles (RT Professional).....	1843
3.2.11.19	OnTestVariablenBeginstartvarupdate (RT Professional).....	1844
3.2.11.20	OnTestVariablenEndstartvarupdate (RT Professional).....	1845
3.2.11.21	OnTestVariablenGetvalue (RT Professional).....	1846
3.2.11.22	OnTestVariablenGetvaluawait (RT Professional).....	1849
3.2.11.23	OnTestVariablenGetVarInfo (RT Professional).....	1853
3.2.11.24	OnTestVariablenGetvarlimits (RT Professional).....	1854
3.2.11.25	OnTestVariablenGetvartype (RT Professional).....	1855
3.2.11.26	OnTestVariablenResumevarupdate (RT Professional).....	1856
3.2.11.27	OnTestVariablenSetvalue (RT Professional).....	1857
3.2.11.28	OnTestVariablenSetvaluawait (RT Professional).....	1858
3.2.11.29	OnTestVariablenStopallupdates (RT Professional).....	1859
3.2.11.30	OnTestVariablenStopvarupdate (RT Professional).....	1859
3.2.11.31	OnTestVariablenSuspendvarupdate (RT Professional).....	1860

3.2.11.32	OnTestWinCCShutdown (RT Professional).....	1860
3.2.11.33	Open project by means of dialog (RT Professional).....	1861
3.2.11.34	Read tag (RT Professional).....	1862
3.2.11.35	Write tag (RT Professional).....	1867
3.3	Functions of the graphic system (RT Professional).....	1869
3.3.1	Basics (RT Professional).....	1869
3.3.1.1	Overview of the functions (RT Professional).....	1869
3.3.1.2	Overview of structures (RT Professional).....	1870
3.3.1.3	Error messages (RT Professional).....	1870
3.3.1.4	Constants (RT Professional).....	1872
3.3.1.5	List of object properties (A-K) (Graphics Designer) (RT Professional).....	1873
3.3.1.6	List of object properties (L-Z) (Graphics Designer) (RT Professional).....	1884
3.3.1.7	API calls to OCXs (RT Professional).....	1896
3.3.2	Structures (RT Professional).....	1896
3.3.2.1	LINKINFO (RT Professional).....	1896
3.3.2.2	MULTILINK (RT Professional).....	1898
3.3.2.3	MULTILINKINFO (RT Professional).....	1899
3.3.2.4	FOCUSINFO (RT Professional).....	1900
3.3.3	General functions (RT Professional).....	1900
3.3.3.1	PDLRTCclosePicture (RT Professional).....	1900
3.3.3.2	PDLRTDisableClosePicture (RT Professional).....	1902
3.3.3.3	PDLRTEnableClosePicture (RT Professional).....	1904
3.3.3.4	PDLRTGotoPicture (RT Professional).....	1905
3.3.3.5	PDLRTInquireFreeArea (RT Professional).....	1907
3.3.3.6	PDLRTOpenPicture (RT Professional).....	1908
3.3.3.7	PDLRTPictureNavigation (RT Professional).....	1911
3.3.3.8	PDLRTShowApp (RT Professional).....	1912
3.3.3.9	PDLRT_CALLBACK (RT Professional).....	1913
3.3.4	Functions for affecting the runtime cursor (RT Professional).....	1915
3.3.4.1	PDLRTGetCursorKeys (RT Professional).....	1915
3.3.4.2	PDLRTGetFocus (RT Professional).....	1917
3.3.4.3	PDLRTSetCursorKeys (RT Professional).....	1919
3.3.4.4	PDLRTSetFocus (RT Professional).....	1921
3.3.5	Functions for processing object properties (RT Professional).....	1923
3.3.5.1	PDLRTGetDefPropEx (RT Professional).....	1923
3.3.5.2	PDLRTGetPropEx (RT Professional).....	1925
3.3.5.3	PDLRTSetPropEx (RT Professional).....	1928
3.3.6	Functions for processing dynamics (RT Professional).....	1933
3.3.6.1	PDLRTGetLink (RT Professional).....	1933
3.3.6.2	PDLRTSetLink (RT Professional).....	1935
3.3.6.3	PDLRTSetMultiLink (RT Professional).....	1937
3.4	Scripting functions (RT Professional).....	1939
3.4.1	Basics (RT Professional).....	1939
3.4.1.1	Overview of the functions (RT Professional).....	1939
3.4.1.2	Overview of structures (RT Professional).....	1939
3.4.1.3	Error messages (RT Professional).....	1940
3.4.1.4	Constants (RT Professional).....	1942
3.4.2	Structures (RT Professional).....	1944
3.4.2.1	AP_ACT_KEY (RT Professional).....	1944
3.4.2.2	AP_ACT_RESULT_STRUCT (RT Professional).....	1946
3.4.2.3	CREATE_USER_HEADER_FILE (RT Professional).....	1947

3.4.2.4	GENERATE_COMPILE (RT Professional).....	1948
3.4.2.5	GET_ACTION_STREAM (RT Professional).....	1948
3.4.3	General functions (RT Professional).....	1949
3.4.3.1	APConnect (RT Professional).....	1949
3.4.3.2	APDisconnect (RT Professional).....	1951
3.4.3.3	APSetLanguage (RT Professional).....	1953
3.4.3.4	AP_RT_PROC (RT Professional).....	1954
3.4.4	Functions for processing Source Code (RT Professional).....	1957
3.4.4.1	APCompile (RT Professional).....	1957
3.4.4.2	APCompileEx (RT Professional).....	1959
3.4.4.3	GSCGenCompile (RT Professional).....	1962
3.4.4.4	GSCGenCompileUserFunctions (RT Professional).....	1963
3.4.5	Functions for processing actions (RT Professional).....	1965
3.4.5.1	GSCGenGetActionStream (RT Professional).....	1965
3.4.6	Functions of the action programming (RT Professional).....	1966
3.4.6.1	APActive (RT Professional).....	1966
3.4.6.2	APEndAct (RT Professional).....	1968
3.4.6.3	APFreeResultStruct (RT Professional).....	1969
3.4.6.4	APInactive (RT Professional).....	1970
3.4.6.5	APStart (RT Professional).....	1972
3.4.6.6	APTransact (RT Professional).....	1975
3.4.7	Examples (RT Professional).....	1977
3.4.7.1	Establish connection to script programming (RT Professional).....	1977
3.5	Functions of the user administration (RT Professional).....	1979
3.5.1	Basics (RT Professional).....	1979
3.5.1.1	Overview of the functions (RT Professional).....	1979
3.5.1.2	Overview of structures (RT Professional).....	1980
3.5.1.3	Error messages (RT Professional).....	1981
3.5.1.4	Constants (RT Professional).....	1982
3.5.2	Structures (RT Professional).....	1982
3.5.2.1	PWGEN_GROUPINFO (RT Professional).....	1982
3.5.2.2	PWGEN_LEVELINFO (RT Professional).....	1983
3.5.2.3	PWGEN_USERINFO (RT Professional).....	1984
3.5.3	General functions (RT Professional).....	1984
3.5.3.1	PWGENConnect (RT Professional).....	1984
3.5.3.2	PWGENDisconnect (RT Professional).....	1986
3.5.4	Functions for processing users (RT Professional).....	1987
3.5.4.1	PWGENAddUser (RT Professional).....	1987
3.5.4.2	PWGENAddUserEx (RT Professional).....	1989
3.5.4.3	PWGENChangePassword (RT Professional).....	1991
3.5.4.4	PWGENCheckUser (RT Professional).....	1992
3.5.4.5	PWGENDeleteUser (RT Professional).....	1994
3.5.4.6	PWGENEnumUsers (RT Professional).....	1995
3.5.4.7	PWGEN_ENUM_USERS_CALLBACK (RT Professional).....	1997
3.5.5	Functions for processing user groups (RT Professional).....	1998
3.5.5.1	PWGENAddGroup (RT Professional).....	1998
3.5.5.2	PWGENEnumGroups (RT Professional).....	2000
3.5.5.3	PWGEN_ENUM_GROUPS_CALLBACK (RT Professional).....	2001
3.5.6	Functions for processing authorizations (RT Professional).....	2002
3.5.6.1	PWGENAddPermLevel (RT Professional).....	2002
3.5.6.2	PWGENCheckPermission (RT Professional).....	2004
3.5.6.3	PWGENDeletePermLevel (RT Professional).....	2005

3.5.6.4	PWGENEnumPermLevels (RT Professional).....	2006
3.5.6.5	PWGEN_ENUM_LEVELS_CALLBACK (RT Professional).....	2008
3.5.6.6	PWGENReadUserPerm (RT Professional).....	2009
3.5.6.7	PWRTCheckPermission (RT Professional).....	2010
3.5.6.8	PWRTCheckPermissionOnPicture (RT Professional).....	2011
3.5.6.9	PWRTPermissionLevelDialog (RT Professional).....	2012
3.5.6.10	PWRTPermissionLevelDialogEx (RT Professional).....	2013
3.5.6.11	PWRTPermissionToString (RT Professional).....	2014
3.5.7	Functions for logon, logoff (RT Professional).....	2016
3.5.7.1	PWRTGetCurrentUser (RT Professional).....	2016
3.5.7.2	PWRTGetLoginPriority (RT Professional).....	2017
3.5.7.3	PWRTIsLoggedInByCard (RT Professional).....	2018
3.5.7.4	PWRTLogin (RT Professional).....	2019
3.5.7.5	PWRTLogout (RT Professional).....	2020
3.5.7.6	PWRTLogoutEx (RT Professional).....	2021
3.5.7.7	PWRTSilentLogin (RT Professional).....	2022
3.5.7.8	PWRTSilentLoginEx (RT Professional).....	2023
3.5.8	Examples (RT Professional).....	2026
3.5.8.1	PWRT check permission (RT Professional).....	2026
3.5.8.2	Checks admission of a certain level for a screen (RT Professional).....	2027
3.5.8.3	Gets a string associated with the permission number (RT Professional).....	2028
3.5.8.4	Authorization level query through a dialog with specifying a possible error (RT Professional).....	2029
3.5.8.5	Return the name of the current user (RT Professional).....	2030
3.5.8.6	Queries the current logon priority (RT Professional).....	2031
3.5.8.7	Checks if the user has been logged on by card (RT Professional).....	2031
3.5.8.8	PWRT login - dialog provided by WinCC itself (RT Professional).....	2033
3.5.8.9	PWRT logoff (RT Professional).....	2034
3.5.8.10	Silent logoff with priority level (RT Professional).....	2035
3.5.8.11	Logon without using a dialog (RT Professional).....	2036
3.5.8.12	Silent logon with priority level (RT Professional).....	2037
3.6	Functions of the text system (RT Professional).....	2038
3.6.1	Basics (RT Professional).....	2038
3.6.1.1	Overview of the functions (RT Professional).....	2038
3.6.1.2	Error messages (RT Professional).....	2038
3.6.1.3	Language codes (RT Professional).....	2040
3.6.2	General functions (RT Professional).....	2041
3.6.2.1	TXTCloseProject (RT Professional).....	2041
3.6.2.2	TXTGetMaxTextID (RT Professional).....	2043
3.6.2.3	TXTOpenProject (RT Professional).....	2044
3.6.2.4	TXTRTConnect (RT Professional).....	2045
3.6.2.5	TXTRTDisconnect (RT Professional).....	2046
3.6.3	Functions for processing infotext (RT Professional).....	2048
3.6.3.1	TXTEnumInfoText (RT Professional).....	2048
3.6.3.2	TXT_ENUM_INFOTEXTS_PROC (RT Professional).....	2050
3.6.3.3	TXTUpdateRuntime (RT Professional).....	2051
3.6.3.4	TXTRTGetInfoText (RT Professional).....	2052
3.6.3.5	TXTRTGetInfoTextMC (RT Professional).....	2054
3.6.4	Functions for processing languages (RT Professional).....	2056
3.6.4.1	TXTEnumLanguages (RT Professional).....	2056
3.6.4.2	TXT_ENUM_LANGUAGES_PROC (RT Professional).....	2057
3.6.4.3	TXTGetFont (RT Professional).....	2059

3.6.4.4	TXShowLanguagesDialog (RT Professional).....	2060
3.6.4.5	TXRTGetLanguageID (RT Professional).....	2062
3.6.4.6	TXRTSetLanguage (RT Professional).....	2063
3.6.5	Examples (RT Professional).....	2065
3.6.5.1	Get infotext (RT Professional).....	2065
3.6.5.2	Enumerate infotexts (RT Professional).....	2067
3.7	Report system functions (RT Professional).....	2069
3.7.1	Basics (RT Professional).....	2069
3.7.1.1	Overview of the functions (RT Professional).....	2069
3.7.1.2	Constants (RT Professional).....	2070
3.7.1.3	Error messages (RT Professional).....	2072
3.7.1.4	List of object properties (RT Professional).....	2073
3.7.1.5	General procedure for processing print job properties (Report Designer) (RT Professional).....	2078
3.7.2	Functions for establishing connections (RT Professional).....	2081
3.7.2.1	RPJAttach (RT Professional).....	2081
3.7.2.2	RPJDetach (RT Professional).....	2082
3.7.2.3	RPJMemFree (RT Professional).....	2083
3.7.3	Functions for processing project properties (RT Professional).....	2084
3.7.3.1	RPJGetNumProjectProperties (RT Professional).....	2084
3.7.3.2	RPJGetProjectPropertyAt (RT Professional).....	2085
3.7.3.3	RPJGetProjectProperty (RT Professional).....	2086
3.7.3.4	RPJProjectLock (RT Professional).....	2087
3.7.3.5	RPJProjectUnlock (RT Professional).....	2089
3.7.3.6	RPJProjectUnlockAll (RT Professional).....	2090
3.7.4	Functions for processing print jobs (RT Professional).....	2091
3.7.4.1	RPJCreateJob (RT Professional).....	2091
3.7.4.2	RPJDeleteJob (RT Professional).....	2092
3.7.4.3	RPJCreatePropertyHandle (RT Professional).....	2094
3.7.4.4	RPJDeletePropertyHandle (RT Professional).....	2095
3.7.4.5	RPJGetJobNameAt (RT Professional).....	2097
3.7.4.6	RPJGetNumJobs (RT Professional).....	2098
3.7.4.7	RPJJobLock (RT Professional).....	2099
3.7.4.8	RPJJobUnlock (RT Professional).....	2101
3.7.4.9	RPJJobUnlockAll (RT Professional).....	2102
3.7.5	Functions for processing print job methods (RT Professional).....	2104
3.7.5.1	RPJCallJobMethod (RT Professional).....	2104
3.7.5.2	RPJGetJobMethodAt (RT Professional).....	2105
3.7.5.3	RPJGetNumJobMethods (RT Professional).....	2106
3.7.6	Functions for processing print job methods (RT Professional).....	2107
3.7.6.1	RPJGetJobPropertyAt (RT Professional).....	2107
3.7.6.2	RPJGetJobProps (RT Professional).....	2109
3.7.6.3	RPJGetNumJobProperties (RT Professional).....	2110
3.7.6.4	RPJGetProperty (RT Professional).....	2111
3.7.6.5	RPJPropertyClear (RT Professional).....	2114
3.7.6.6	RPJSetProperty (RT Professional).....	2115
3.7.7	Examples (RT Professional).....	2119
3.7.7.1	Get print job method name (RT Professional).....	2119
3.7.7.2	Get print job names (RT Professional).....	2122
3.7.7.3	Get print job properties (RT Professional).....	2125
3.7.7.4	Modify print job properties (RT Professional).....	2128
3.7.7.5	Show print job preview (RT Professional).....	2133

3.8	Tag and log functions (RT Professional).....	2136
3.8.1	Basics (RT Professional).....	2136
3.8.1.1	Overview of the functions (RT Professional).....	2136
3.8.1.2	Overview of structures (RT Professional).....	2137
3.8.1.3	Error messages (RT Professional).....	2138
3.8.1.4	Constants (RT Professional).....	2139
3.8.2	Structures (RT Professional).....	2149
3.8.2.1	TLG_ARCHIV_STR (RT Professional).....	2149
3.8.2.2	TLG_ARCHIVDATARAW (RT Professional).....	2152
3.8.2.3	TLG_BACKUP_TABLE_INFO (RT Professional).....	2155
3.8.2.4	TLG_CURVESCALEX (RT Professional).....	2157
3.8.2.5	TLG_CURVSCALEY (RT Professional).....	2162
3.8.2.6	TLG_GETARCHIVDATA (RT Professional).....	2166
3.8.2.7	TLG_IO_BACKUP_SELECT (RT Professional).....	2167
3.8.2.8	TLG_PROT_CURVE_INFOS (RT Professional).....	2168
3.8.2.9	TLG_SCAL_STR (RT Professional).....	2170
3.8.2.10	TLG_TABLE_INFO (RT Professional).....	2171
3.8.2.11	TLG_TABLESCALE (RT Professional).....	2172
3.8.2.12	TLG_TEMPLATEITEM_INFO (RT Professional).....	2175
3.8.2.13	TLG_TIME_STR (RT Professional).....	2177
3.8.2.14	TLG_TIMEDATA (RT Professional).....	2178
3.8.2.15	TLG_TPLITEM_CURVE (RT Professional).....	2179
3.8.2.16	TLG_TPLITEM_INFO (RT Professional).....	2180
3.8.2.17	TLG_TPLITEM_TABLE (RT Professional).....	2181
3.8.2.18	TLG_VAR_STR (RT Professional).....	2182
3.8.2.19	TLG_VARIABLE_INFO (RT Professional).....	2187
3.8.3	General functions (RT Professional).....	2188
3.8.3.1	TLGCSCConnect (RT Professional).....	2188
3.8.3.2	TLGCSCConnectEx (RT Professional).....	2190
3.8.3.3	TLGCSDisConnect (RT Professional).....	2191
3.8.3.4	TLGChangeLanguage (RT Professional).....	2192
3.8.3.5	TLGConnect (RT Professional).....	2193
3.8.3.6	TLGDisconnect (RT Professional).....	2195
3.8.4	Project management functions (RT Professional).....	2196
3.8.4.1	TLGCloseProject (RT Professional).....	2196
3.8.4.2	TLGOpenProject (RT Professional).....	2198
3.8.4.3	TLGEnumProject (RT Professional).....	2200
3.8.4.4	TLG_ENUM_PROJECT_NAME_CALLBACK (RT Professional).....	2201
3.8.5	Functions for processing tags (RT Professional).....	2202
3.8.5.1	TLGEnumVariables (RT Professional).....	2202
3.8.5.2	TLG_ENUM_VARIABLE_NAME_CALLBACK (RT Professional).....	2203
3.8.5.3	TLGEnumVariablesEx (RT Professional).....	2204
3.8.5.4	TLG_ENUMVARIABLES (RT Professional).....	2206
3.8.5.5	TLGReadVariable (RT Professional).....	2207
3.8.6	Functions for processing logs (RT Professional).....	2209
3.8.6.1	TLGEnumArchives (RT Professional).....	2209
3.8.6.2	TLG_ENUM_ARCHIV_CALLBACK (RT Professional).....	2210
3.8.6.3	TLGEnumArchivs (RT Professional).....	2212
3.8.6.4	TLGEnumArchivsEx (RT Professional).....	2214
3.8.6.5	TLGEnumArchivsSel (RT Professional).....	2215
3.8.6.6	TLG_ENUMTABLES (RT Professional).....	2217
3.8.6.7	TLGFreeMemory (RT Professional).....	2219

3.8.6.8	TLGGetArchivData (RT Professional).....	2220
3.8.6.9	TLG_GETARCHIVDATA_CALLBACK (RT Professional).....	2222
3.8.6.10	TLGGetArchivDataEx (RT Professional).....	2224
3.8.6.11	TLGGetClosestTime (RT Professional).....	2226
3.8.6.12	TLGGetClosestTimeEx (RT Professional).....	2228
3.8.6.13	TLGInsertArchivData (RT Professional).....	2230
3.8.6.14	TLGLockArchiv (RT Professional).....	2233
3.8.6.15	TLGLockVariable (RT Professional).....	2234
3.8.6.16	TLGReadArchiv (RT Professional).....	2236
3.8.7	Functions for trend and table views (RT Professional).....	2237
3.8.7.1	TLGCloseWindow (RT Professional).....	2237
3.8.7.2	TLGDrawCurvesInDC (RT Professional).....	2238
3.8.7.3	TLGInsertTemplateItem (RT Professional).....	2240
3.8.7.4	TLGPressToolBarButton (RT Professional).....	2241
3.8.7.5	TLGSetRulerWindowVisible (RT Professional).....	2243
3.8.7.6	TLGShowWindow (RT Professional).....	2244
3.8.8	Functions for processing time systems (RT Professional).....	2245
3.8.8.1	TLGEnumTime (RT Professional).....	2245
3.8.8.2	TLG_ENUM_TIME_NAME_CALLBACK (RT Professional).....	2247
3.8.8.3	TLGEnumTimes (RT Professional).....	2248
3.8.8.4	TLG_ENUMTIMES_CALLBACK (RT Professional).....	2250
3.8.8.5	TLGReadTime (RT Professional).....	2251
3.8.9	Functions for saving and restoring (RT Professional).....	2252
3.8.9.1	TLGEnumBackupEntries (RT Professional).....	2252
3.8.9.2	TLG_ENUMBACKUP_ENTRIES (RT Professional).....	2254
3.8.9.3	TLGExport (RT Professional).....	2255
3.8.9.4	TLGGetBackupSize (RT Professional).....	2257
3.8.10	Examples (RT Professional).....	2259
3.8.10.1	Edit curve template - Example 1 (RT Professional).....	2259
3.8.10.2	Enum all acquisition and logging times (RT Professional).....	2262
3.8.10.3	Enum logs (RT Professional).....	2265
3.8.10.4	Enum tags of a log (RT Professional).....	2268
3.8.10.5	Enumerate logs (RT Professional).....	2270
3.8.10.6	Read log (RT Professional).....	2272
3.8.10.7	Read parameters of time object (RT Professional).....	2276
3.9	Recipe functions (RT Professional).....	2278
3.9.1	Basics (RT Professional).....	2278
3.9.1.1	Overview of the functions (RT Professional).....	2278
3.9.1.2	Structure of recipes (RT Professional).....	2280
3.9.1.3	Dependencies in call sequence of API functions (RT Professional).....	2282
3.9.1.4	Error messages (RT Professional).....	2283
3.9.1.5	Constants (RT Professional).....	2284
3.9.2	General functions (RT Professional).....	2286
3.9.2.1	uaGetLastError (RT Professional).....	2286
3.9.2.2	uaIsActive (RT Professional).....	2288
3.9.2.3	uaUsers (RT Professional).....	2289
3.9.2.4	uaSetLocalEvents (RT Professional).....	2289
3.9.3	Functions for generating connections (RT Professional).....	2290
3.9.3.1	uaConnect (RT Professional).....	2290
3.9.3.2	uaDisconnect (RT Professional).....	2292
3.9.3.3	uaQueryArchive (RT Professional).....	2293
3.9.3.4	uaQueryArchiveByName (RT Professional).....	2295

3.9.3.5	UaQueryConfiguration (RT Professional).....	2297
3.9.3.6	uaReleaseArchive (RT Professional).....	2298
3.9.3.7	uaReleaseConfiguration (RT Professional).....	2299
3.9.4	Functions for recipe processing (RT Professional).....	2301
3.9.4.1	uaArchiveClose (RT Professional).....	2301
3.9.4.2	uaArchiveDelete (RT Professional).....	2302
3.9.4.3	uaArchiveExport (RT Professional).....	2303
3.9.4.4	uaArchiveGetID (RT Professional).....	2304
3.9.4.5	uaArchiveGetName (RT Professional).....	2305
3.9.4.6	uaArchiveGetSort (RT Professional).....	2307
3.9.4.7	uaArchiveImport (RT Professional).....	2308
3.9.4.8	uaArchiveOpen (RT Professional).....	2309
3.9.4.9	uaArchiveUpdate (RT Professional).....	2310
3.9.4.10	uaGetArchive (RT Professional).....	2311
3.9.4.11	uaGetNumArchives (RT Professional).....	2313
3.9.4.12	uaOpenArchives (RT Professional).....	2313
3.9.5	Functions for recipe element processing (RT Professional).....	2314
3.9.5.1	uaArchiveGetCount (RT Professional).....	2314
3.9.5.2	uaArchiveGetFilter (RT Professional).....	2316
3.9.5.3	uaArchiveInsert (RT Professional).....	2317
3.9.5.4	uaArchiveMoveFirst (RT Professional).....	2318
3.9.5.5	uaArchiveMoveLast (RT Professional).....	2319
3.9.5.6	uaArchiveMoveNext (RT Professional).....	2320
3.9.5.7	uaArchiveMovePrevious (RT Professional).....	2321
3.9.5.8	uaGetNumFields (RT Professional).....	2322
3.9.6	Functions for field processing (RT Professional).....	2323
3.9.6.1	uaArchiveGetFieldLength (RT Professional).....	2323
3.9.6.2	uaArchiveGetFieldName (RT Professional).....	2324
3.9.6.3	uaArchiveGetFields (RT Professional).....	2325
3.9.6.4	uaArchiveGetFieldType (RT Professional).....	2326
3.9.6.5	uaArchiveGetFieldValueDate (RT Professional).....	2327
3.9.6.6	uaArchiveGetFieldValueDouble (RT Professional).....	2328
3.9.6.7	uaArchiveGetFieldValueLong (RT Professional).....	2330
3.9.6.8	uaArchiveGetFieldValueString (RT Professional).....	2331
3.9.6.9	uaArchiveSetFieldValueDate (RT Professional).....	2332
3.9.6.10	uaArchiveSetFieldValueDouble (RT Professional).....	2333
3.9.6.11	uaArchiveSetFieldValueLong (RT Professional).....	2334
3.9.6.12	uaArchiveSetFieldValueString (RT Professional).....	2336
3.9.6.13	uaGetField (RT Professional).....	2337
3.9.7	Filter and sort functions. (RT Professional).....	2338
3.9.7.1	uaArchiveRequery (RT Professional).....	2338
3.9.7.2	uaArchiveSetFilter (RT Professional).....	2339
3.9.7.3	uaArchiveSetSort (RT Professional).....	2341
3.9.8	Functions for recipe view processing (RT Professional).....	2342
3.9.8.1	uaOpenViews (RT Professional).....	2342
3.9.9	Functions for variables processing (RT Professional).....	2343
3.9.9.1	uaArchiveReadTagValues (RT Professional).....	2343
3.9.9.2	uaArchiveReadTagValuesByName (RT Professional).....	2344
3.9.9.3	uaArchiveWriteTagValues (RT Professional).....	2345
3.9.9.4	uaArchiveWriteTagValuesByName (RT Professional).....	2347
3.10	Alarm functions (RT Professional).....	2348
3.10.1	Basics (RT Professional).....	2348

3.10.1.1	Overview of the functions (RT Professional).....	2348
3.10.1.2	Overview of structures (RT Professional).....	2349
3.10.1.3	Error messages (RT Professional).....	2350
3.10.1.4	Constants (RT Professional).....	2352
3.10.2	Structures (RT Professional).....	2363
3.10.2.1	MSG_RTCREATE_STRUCT_PLUS (RT Professional).....	2363
3.10.2.2	MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (RT Professional).....	2365
3.10.2.3	MSG_FILTER_STRUCT_PLUS (RT Professional).....	2369
3.10.2.4	MSG_RTDATA_INSTANCE_STRUCT_PLUS (RT Professional).....	2373
3.10.2.5	MSG_CLASS_STRUCT_PLUS (RT Professional).....	2376
3.10.2.6	MSG_CSDATA_STRUCT_PLUS (RT Professional).....	2378
3.10.2.7	MSG_RTGROUPENUM_STRUCT_PLUS (RT Professional).....	2381
3.10.2.8	MSG_RTGROUPSET_STRUCT_PLUS (RT Professional).....	2382
3.10.2.9	MSG_COMMENT_INSTANCE_STRUCT_PLUS (RT Professional).....	2383
3.10.2.10	MSG_INFOTEXT_STRUCT_PLUS (RT Professional).....	2384
3.10.2.11	MSG_BACKUP_STRUCT_PLUS (RT Professional).....	2385
3.10.2.12	MSG_RTLOCK_STRUCT_PLUS (RT Professional).....	2387
3.10.2.13	MSG_TEXTVAL256_STRUCT_PLUS (RT Professional).....	2388
3.10.3	General functions (RT Professional).....	2389
3.10.3.1	MSRTEnumArchivDataPlus (RT Professional).....	2389
3.10.3.2	MSRTStartMsgServicePlus (RT Professional).....	2391
3.10.3.3	MSRTStopMsgServicePlus (RT Professional).....	2394
3.10.3.4	MSRTQuitHornPlus (RT Professional).....	2396
3.10.3.5	MSG_SERVICE_NOTIFY_PROCPlus (RT Professional).....	2397
3.10.3.6	MSRTWebClientPlus (RT Professional).....	2400
3.10.4	Functions for protocol processing (RT Professional).....	2400
3.10.4.1	MSRTActivateMProtPlus (RT Professional).....	2400
3.10.4.2	MSRTEnumProtDataPlus (RT Professional).....	2401
3.10.4.3	MSRTPrintMProtPlus (RT Professional).....	2403
3.10.5	Functions for alarm processing (RT Professional).....	2404
3.10.5.1	MSRTEnumArchivInstancePlus (RT Professional).....	2404
3.10.5.2	MSRTCreateMsgInstanceWithCommentPlus (RT Professional).....	2406
3.10.5.3	MSRTCreateMsgInstancePlus (RT Professional).....	2407
3.10.5.4	MSRTDialogMsgLockPlus (RT Professional).....	2410
3.10.5.5	MSRTCreateMsgPlus (RT Professional).....	2411
3.10.5.6	MSRTEnumLockedMsgPlus (RT Professional).....	2413
3.10.5.7	MSRTEnumMsgRTDataPlus (RT Professional).....	2414
3.10.5.8	MSRTGetClassInfoPlus (RT Professional).....	2415
3.10.5.9	MSRTGetLastMsgWithCommentPlus (RT Professional).....	2417
3.10.5.10	MSRTGetMsgActualPlus (RT Professional).....	2418
3.10.5.11	MSRTGetMsgCSDDataPlus (RT Professional).....	2420
3.10.5.12	MSRTGetMsgPriorityPlus (RT Professional).....	2421
3.10.5.13	MSRTGetMsgQuitPlus (RT Professional).....	2423
3.10.5.14	MSRTGetMsgTextPlus (RT Professional).....	2424
3.10.5.15	MSRTGetSelectedMsgPlus (RT Professional).....	2426
3.10.5.16	MSRTResetMsgPlus (RT Professional).....	2428
3.10.5.17	MSRTLoopInAlarmPlus (RT Professional).....	2430
3.10.6	Functions for alarm group processing (RT Professional).....	2431
3.10.6.1	MSRTEnumGroupMsgPlus (RT Professional).....	2431
3.10.6.2	MSRTLockGroupPlus (RT Professional).....	2433
3.10.6.3	MSRTQuitGroupPlus (RT Professional).....	2434
3.10.7	Functions for alarm filter processing (RT Professional).....	2435

3.10.7.1	MSRTGetFilterDataPlus (RT Professional).....	2435
3.10.7.2	MSRTCheckWinFilterPlus (RT Professional).....	2437
3.10.7.3	MSRTSetMsgFilterPlus (RT Professional).....	2439
3.10.7.4	MSRTSetMsgWinFilterPlus (RT Professional).....	2440
3.10.8	Functions for processing alarm view (RT Professional).....	2441
3.10.8.1	MSRTMsgWinCommandPlus (RT Professional).....	2441
3.10.9	Functions for processing comments (RT Professional).....	2443
3.10.9.1	MSRTGetCommentInstancePlus (RT Professional).....	2443
3.10.9.2	MSRTSetCommentInstancePlus (RT Professional).....	2445
3.10.10	Functions for info text processing (RT Professional).....	2446
3.10.10.1	MSRTGetInfotextPlus (RT Professional).....	2446
3.10.10.2	MSRTSetInfotextPlus (RT Professional).....	2448
3.10.11	Archiving functions (RT Professional).....	2449
3.10.11.1	MSRTEnumBackupListPlus (RT Professional).....	2449
3.10.11.2	MSRTExportPlus (RT Professional).....	2451
3.11	Functions for displaying the PLC code (RT Professional).....	2452
3.11.1	Display in STEP 7 (RT Professional).....	2452
3.11.1.1	Basics (RT Professional).....	2452
3.11.1.2	OpenTIAPortalProject (RT Professional).....	2453
3.11.1.3	OpenTIAPortalIECPLByCall (RT Professional).....	2455
3.11.1.4	OpenTIAPortalIECPLByAssignment (RT Professional).....	2457
3.11.1.5	OpenTIAPortalS7GraphByBlock (RT Professional).....	2459
3.11.1.6	Example: Integration in a WinCC function (RT Professional).....	2462
3.11.2	Display in the PLC code display (RT Professional).....	2463
3.11.2.1	Basics (RT Professional).....	2463
3.11.2.2	OpenViewerS7GraphByBlock (RT Professional).....	2463
3.11.2.3	OpenViewerIECPLByCall (RT Professional).....	2466
3.11.2.4	OpenViewerIECPLByFCCall (RT Professional).....	2469
3.11.2.5	OpenViewerIECPLByAssignment (RT Professional).....	2472
3.11.3	Error-handling (RT Professional).....	2474
3.12	Functions for displaying the PLC code in an external application (RT Professional).....	2476
3.12.1	Basics (RT Professional).....	2476
3.12.2	Querying static information of networks (RT Professional).....	2478
3.12.2.1	Initialize (RT Professional).....	2478
3.12.2.2	SetLanguage (RT Professional).....	2478
3.12.2.3	RequestByAssignment (RT Professional).....	2479
3.12.2.4	RequestByCall (RT Professional).....	2481
3.12.2.5	CancelRequest (RT Professional).....	2482
3.12.2.6	Terminate (RT Professional).....	2483
3.12.2.7	Structure of the XML data (RT Professional).....	2484
3.12.3	Querying dynamic statistical information of networks (RT Professional).....	2487
3.12.3.1	StartDynamicDataSubscription (RT Professional).....	2487
3.12.3.2	StopDynamicSubscription (RT Professional).....	2488
3.13	Troubleshooting (RT Professional).....	2488
3.13.1	CMN_ERROR (RT Professional).....	2488
3.13.2	CCStorageError.h (RT Professional).....	2490
Index		2501

VB scripting

1.1 System functions (Panels, Comfort Panels, RT Advanced)

1.1.1 AcknowledgeAlarm (Panels, Comfort Panels, RT Advanced)

Description

Acknowledges all selected alarms.

This system function is used when the HMI device does not have an ACK key or when the integrated key of the alarm view should not be used.

This system function can only be used for function keys.

Use in the function list

AcknowledgeAlarm

Use in user-defined functions

AcknowledgeAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.2 ActivatePLCCodeViewer (Panels, Comfort Panels, RT Advanced)

Description

Performs a screen change to the given screen with the PLC code view. The PLC code view shows the program code of the relevant network.

You configure the ActivatePLCCodeView system function at the "Click PLC code view button" event of a GRAPH overview or at an event of a button.

Screen change from the GRAPH overview

You configure the system function at the "Click PLC code view button" event in a GRAPH overview to display the GRAPH step sequencer in the PLC code view. If no error is pending, the sequence of the GRAPH step sequencer is displayed.

If an error occurs, the faulty step is displayed in the PLC code view after jump of the faulty step. If multiple errors have occurred, the first faulty step of the sequencer is displayed after the jump. If you have corrected the error, the next faulty step is automatically displayed in the view. In the detail view, the transition or the interlock is displayed depending on the type of error.

Screen change from an alarm

You configure the system function at an event of a button. When the button is pressed in Runtime, the system function checks whether the alarm selected last in the configured alarm view is a supervision alarm or a GRAPH alarm. If a jump is possible for this alarm, the configured PLC code view is opened with the corresponding program code.

The jump from an alarm in the alarm view to the PLC code view is made possible for the following supervision alarms:

- With global supervisions, only for interlock supervisions (Interlock)
- With local supervisions, for all basic supervisions at input parameters

You can find more information on supervisions in the section "Programming PLC > Supervising machinery and plants with ProDiag".

The jump to the PLC code view is possible for all GRAPH alarms. After the jump, the GRAPH sequencer and the faulty step are displayed in the PLC code view.

If a supervision alarm and an interlock alarm are pending simultaneously for the same step, the interlock network is always displayed first after triggering the system function in the PLC code view regardless of whether the supervision alarm or the interlock alarm was selected.

For the jump from a supervision alarm to the PLC code view, the instance name must conform to the following naming convention when using supported local operands in a function block: <FB-Name>_DB.

The jump to a function or an organization block is only possible if only global operands are used.

Note

The jump in connection with the alarm buffer or the alarm log is not supported.

Use in the function list

ActivatePLCCodeView(Screen name, Screen object)

Use in user-defined functions

ActivatePLCCodeView (Screen_name, Object_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Note

Use of the system function "ActivatePLCCodeView" in a VB script has no effect at a GRAPH overview.

Parameters

Screen name

Name of the screen that contains the PLC code view.

Note

Screen change to a pop-up screen or a slide-in screen is not supported.

Screen object

Object name of the PLC code view.

1.1.3 ActivatePreviousScreen (Panels, Comfort Panels, RT Advanced)

Description

Performs a screen change to the screen which was activated before the current screen. The screen change is not performed if no screen was activated beforehand.

The last 10 screens that were called up are saved. A system alarm is output when you change to a screen which is no longer saved.

Note

If you want to use the system function, the screen to which you change has to be used in the navigation structure.

Use in the function list

ActivatePreviousScreen

Use in user-defined functions

ActivatePreviousScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.4 ActivateScreen (Panels, Comfort Panels, RT Advanced)

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent area or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent area when the system function is called up, the permanent area maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The following program code activates "Screen_2" with the ActivateScreen function, for example, when you click a button.

```
Sub ActivateScreen_2()  
  
  'User-defined code  
  '' i. e. when pressing a button  
  
  ActivateScreen "Screen_2",0
```

1.1.5 ActivateScreenByNumber (Panels, Comfort Panels, RT Advanced)

Description

Performs a screen change to a screen depending on a tag value.

The screen is identified by its screen number.

Use in the function list

ActivateScreenByNumber (Screen number, Object number)

Use in user-defined functions

ActivateScreenByNumber (Screen_number, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Screen number**

Tag which contains the screen number of the destination screen.

When a change from the root screen to the permanent area is desired, specify "0" or "-1":

0 = Change from root screen to permanent area.

-1 = Change from permanent area to root screen

Object number

The number of the screen object which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent area when the system function is called up, the permanent area maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

1.1.6 ActivateSystemDiagnosticsView (Panels, Comfort Panels, RT Advanced)**Description**

Activates the system diagnostics view. The system diagnostics view shows the detail view of the device concerned.

Use in the function list

ActivateSystemDiagnosticsView (Screen name, Screen object)

Use in user-defined functions

ActivateSystemDiagnosticsView (Screen_name, Object_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Screen name**

Name of the screen contained in the system diagnostics view.

Screen object

Object name of the system diagnostics view.

1.1.7 ArchiveLogFile (Panels, Comfort Panels, RT Advanced)

Description

This system function moves or copies a log to another storage location for long-term logging.

With Audit Trails, always use the "Move" (hmiMove)" mode, otherwise you will be violating the FDA guideline by duplicating the data stored.

Prior to "ArchiveLogFile" always run the system function "CloseAllLogs".

After this system function, run the "OpenAllLogs" system function.

In "Copy log" mode, the logs are not reopened until the log has been successfully copied, or unless a timeout occurs during the copy. In "Move log" mode, the logs to be moved are renamed and the new logs are opened immediately. To move the renamed logs, a job is submitted which attempts to move them every 300 seconds if the server cannot be reached. The job is also retained after Runtime is restarted until it is executed.

Use in the function list

ArchiveLogFile (Log type, Log, Directory name, Mode)

Use in user-defined functions

ArchiveLogFile (Log_type, Log, Directory_name, Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Tag log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive)= Audit trail log available for GMP compliant projects.. Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log being archived.

Directory name

Path for saving the log.

Mode

0 (hmiCopy) = Copy log

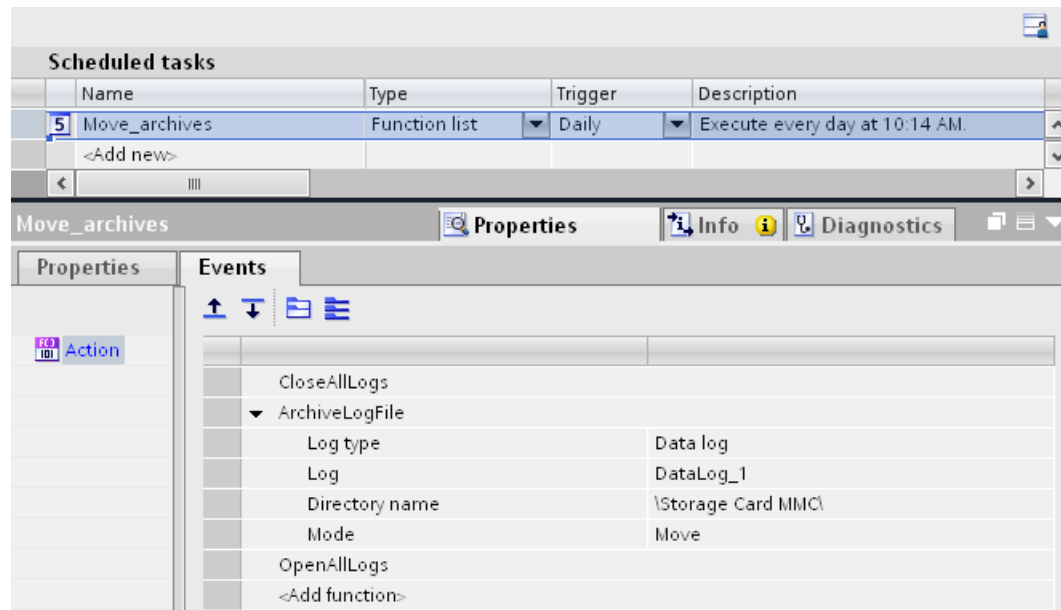
1 (hmiMove) = Move log

Application example

You want to move a log file from a local storage medium to the server in order to generate a backup copy of this file at cyclic intervals.

Notes on configuring

Set up a task in the scheduler which is executed at a specific time on a daily basis. Configure the following function list for the task:



Procedure on HMI device

- All log files will be closed.
- The log file you specified is moved to the server.
- All closed log files will be opened again.

1.1.8 Back up RAM file system (Panels, Comfort Panels, RT Advanced)

Description

Backs up the RAM file system in the memory medium of the HMI device.

After restarting the HMI device, the data is automatically reloaded in the RAM file system.

Applications such as the Internet Explorer save data (e.g. the last web sites called up) temporarily to the DRAM file system of the HMI device.

Use in the function list

BackupRAMFileSystem

Use in user-defined functions

BackupRAMFileSystem

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.9 CalibrateTouchScreen (Panels, Comfort Panels, RT Advanced)**Description**

Calls a program for calibrating the touch screen.

During the calibration process, there is a prompt to touch five positions on the screen display. Touch the screen display within 30 seconds, to confirm the calibration process. If the calibration is not completed within this time span, the calibration settings are discarded. The user prompt is in English.

Use this system function the first time you start the HMI device.

Use in the function list

CalibrateTouchScreen

Use in user-defined functions

CalibrateTouchScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Note

The CalibrateTouchScreen system function cannot be simulated.

1.1.10 ChangeConnection (Panels, Comfort Panels, RT Advanced)

Description

Disconnects the connection to the PLC currently in use and establishes a connection to a PLC with another address. The newly connected PLC must belong to the same device class (S7-300, S7-400, etc).

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address
- MPI address

The follows PLC types are supported:

- SIMATIC S7 300/400
- SIMATIC S7 200
- SIMATIC S7 1200 (device version up to V11)
- SIMATIC S7 LOGO!
- SIMATIC S7-NC
- SIMOTION

Use in the function list

ChangeConnection (Connection, Address, Slot, Rack)

Use in user-defined functions

ChangeConnection (Connection, Address, Slot, Rack)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

MPI/PROFIBUS or IP address of the PLC to which the connection will be established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data types:

- Ethernet connection: "String" data type
 - MPI connection: "Int" data type
-

Slot

Slot of the PLC to which the connection will be established.

Rack

Rack of the PLC to which the connection will be established.

Application example

You want to operate one HMI device on several machines. Configure a controller in the project. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. All tags in this connection are now updated from the new controller.

The PLC which you have indicated when creating the project will be used as default.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnection" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

1.1.11 ChangeConnectionEIP (Panels, Comfort Panels, RT Advanced)

Description

Terminates the connection to the Allen Bradley controller currently in use and establishes a connection to another Allen Bradley controller in Runtime. "EIP" stands for "Ethernet/IP".

The newly connected controller must belong to the same device class (Allen Bradley).

1.1 System functions (Panels, Comfort Panels, RT Advanced)

For the connection to be established, the CPU type of the controller to be connected must match the CPU type of the controller already connected, for example, both controllers have the CPU type "SLC/Micrologix PLC".

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address

Use in the function list

ChangeConnectionEIP (Connection, Address, Communication path)

Use in user-defined functions

ChangeConnectionEIP (Connection, Address, Communication Path)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

IP address of the controller to which the connection is being established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data type "String".

Note

The stored IP address must consist of 4 decimal numbers with the value range 0 to 255.

Communication path

CIP path from the Ethernet module to the controller to which the connection is being established.

You specify the communication path either with a String tag or an HMI tag of the "String" data type.

Application example

You want to operate one HMI device on several machines. To do so, you configure a PLC in the project. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. All tags of this connection are now updated from the new PLC.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnectionEIP" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

1.1.12 ClearAlarmBuffer (Panels, Comfort Panels, RT Advanced)

Description

Deletes alarms from the alarm buffer on the HMI device.

Note

Alarms which have not yet been acknowledged are also deleted.

Use in the function list

ClearAlarmBuffer (Alarm class number)

Use in user-defined functions

ClearAlarmBuffer (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Determines which alarms are to be deleted from the alarm buffer:

0 (hmiAll) = All alarms

1 (hmiAlarms) = Alarms of alarm class "Error"

2 (hmiEvents) = Alarms of alarm class "Warning"

3 (hmiSystem) = Alarms of alarm class "System event"

4 (hmiS7Diagnosis) = Alarms of alarm class "S7 diagnostic alarm"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

1.1.13 ClearAlarmBufferProtool (Panels, Comfort Panels, RT Advanced)

Description

The system function exists to ensure compatibility.

It has the same functionality as the "ClearAlarmBuffer" system function, but uses the old ProTool numbering.

Use in the function list

ClearAlarmBufferProTool (Alarm class number)

Use in user-defined functions

ClearAlarmBufferProtoolLegacy (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Alarm class number whose messages are to be deleted:

-1 (hmiAllProtoolLegacy) = All alarms/events

0 (hmiAlarmsProtoolLegacy) = Alarms of alarm class "Errors"

1 (hmiEventsProtoolLegacy) = Alarms of alarm class "Warnings"

2 (hmiSystemProtoolLegacy) = Alarms of alarm class "System"

3 (hmiS7DiagnosisProtoolLegacy) = Alarms of alarm class "Diagnosis Events"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

1.1.14 ClearDataRecord (Panels, Comfort Panels, RT Advanced)

Description

Deletes a recipe data record.

Several data records can be deleted from one or more recipes.

Use in the function list

ClearDataRecord (Recipe number/name, Data record number/name, Confirmation, Output status message, Processing status)

Use in user-defined functions

ClearDataRecord

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe from which recipe data records are deleted. "0" is specified if you want to delete recipe data records from all available recipes.

Data record number/name

Number or name of the recipe data record to be deleted. "0" is specified if you want to delete all recipe data records.

Confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.15 ClearDataRecordMemory (Panels, Comfort Panels, RT Advanced)

Description

Deletes all recipes data records from the specified storage medium.

Use in the function list

ClearDataRecordMemory (Storage location, After confirmation, Output status message, Processing status)

Use in user-defined functions

ClearDataRecordMemory (Storage_location, Confirmation, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Storage location

Determines the storage location:

0 (hmiFlashMemory) = Flash memory: Internal flash memory of the HMI device

1 (hmiStorageCard) = Storage card

2 (hmiStorageCard2) = Storage card

3 (hmiStorageCard3) = Storage card MultiMediaCard

4 (hmiStorageCard4) = Storage card

With confirmation

Determines whether the operator should confirm the deletion:

0 (hmiOff) = Off: Deletion is begun without confirmation.

1 (hmiOn) = On: The starting of the deletion process must be confirmed.

Output status message

Determines whether a status message is output after the deletion:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.16 ClearLog (Panels, Comfort Panels, RT Advanced)**Description**

Deletes all data records in the given log.

Use in the function list

ClearLog (Log type, Log)

Use in user-defined functions

ClearLog (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Log type**

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail log Available for GMP-compliant projects Additional information is available under "Activating GMP-compliant configuration".

Log

Name of the log from which all entries are deleted.

1.1.17 CloseAllLogs (Panels, Comfort Panels, RT Advanced)

Description

Disconnects the connection between WinCC and all logs.

Use in the function list

CloseAllLogs

Use in user-defined functions

CloseAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "CloseAllLogs" system function on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When you press the "Close Archive" button, the opened logs are closed. The data medium can be changed. The logging continues during the change of the data medium. The process values to be logged are cached. You open all logs with the "Open Archive" button. The logging process will continue in the specified log. The buffered process values are later added to the log.

1.1.18 ControlSmartServer (Panels, Comfort Panels, RT Advanced)

Description

Starts or stops the Sm@rtServer.

Use in the function list

ControlSmartServer (Mode)

Use in user-defined functions

ControlSmartServer (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Mode**

Specifies whether the Sm@rtServer is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: Sm@rtServer is stopped

1 (hmiStart) = Start: Sm@rtServer is started

1.1.19 ControlWebServer (Panels, Comfort Panels, RT Advanced)**Description**

Starts or stops the Web server.

Use in the function list

ControlWebServer (Mode)

Use in user-defined functions

ControlWebServer (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Mode**

Specifies whether the Web server is started or stopped.

-1 (hmiToggle) = Toggle: Toggles between the two modes

0 (hmiStop) = Stop: The Web server is stopped.

1 (hmiStart) = Start: The Web server is started.

1.1.20 CopyLog (Panels, Comfort Panels, RT Advanced)

Description

Copies the contents of a log to another log. Tag values can only be copied to other data logs and alarms only to other alarm logs, however.

Note

If you copy a log using the "CopyLog" system function, it is possible that external applications will be unable to read certain country-specific special characters in the logged message text of the log copy. This does not apply to WinCC Runtime. WinCC Runtime can read the copied log files without errors.

Note

80% of the log entries are copied when copying the circular logs. 20% of entries are not copied because the space is reserved for buffer overflow by default.

Use in the function list

CopyLog (Log type, Destination log, Source log, Mode, Delete source log)

Use in user-defined functions

CopyLog (Log_type, Destination log, Source_log, Mode, Delete_source_log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

Destination log

Name of the log into which the entries are copied (Destination log).

Source log

Name of the log from which the entries are copied (Source log).

Mode

Determines what is done with copied entries in the destination log:

0 (hmiOverwrite) = Overwrite: Existing entries are overwritten.

2 (hmiAppend) = Append: The entries are inserted at the end of the destination log. When the configured size of the log has been reached, the destination log is treated like a circular log.

Delete source log

Determines whether the source log is deleted after copying.

0 (hmiNo) = No: Is not deleted.

1 (hmiYes) = Yes: Is deleted.

1.1.21 DecreaseTag (Panels, Comfort Panels, RT Advanced)

Description

Subtracts the given value from the tag value.

$X = X - a$

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. The auxiliary tags are assigned to the tag value with the "SetTag" system function.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Tag**

The tag from which the given value is subtracted.

Value

The value which is subtracted.

1.1.22 EditAlarm (Panels, Comfort Panels, RT Advanced)

Description

Triggers the "Edit" event for the selected alarm or, with multiple selection, the last selected alarm.

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

This system function can only be used for function keys.

Use in the function list

EditAlarm

Use in user-defined functions

EditAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.23 Encode (Panels, Comfort Panels, RT Advanced)

Description

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

Use in the function list

Encode (Byte array, String, Coding)

Use in user-defined functions

Encode (Byte_array, String, Encoding)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Byte array**

The tag that contains the converted value.

Note

The Byte array must be twice the size of the string length + 2. The Byte array must contain 242 array elements if the string has a length of 120 characters.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encode

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

1.1.24 EncodeEx (Panels, Comfort Panels, RT Advanced)**Description**

Adapts the "String" data type of a tag for transfer to the automation system (AS). The tag of WinCC data type "String" is converted into the AS data type "Array of byte". The result is written to a tag.

By contrast to the Encode system function, you can define the Line break parameter. Using the Line break parameter you can delete line breaks or replace these with predefined characters.

Use in the function list

EncodeEx (Byte-Array, String, Encoding, Line break)

Use in user-defined functions

EncodeEx (Byte_array, String, Encoding, Line_break)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Byte array**

The tag that contains the converted value.

Note

The byte array must be twice the size of the string length +2. If the string has a length of 120 characters, the byte array must contain 242 array elements.

Characters are either truncated or not converted if the size is insufficient.

String

The tag of data type "String" that is converted.

Encoding

0 (hmiEncodeUTF16LE) - String is encoded in UTF16LE (Unicode 16 Little Endian).

Line break

All line breaks are either deleted or replaced with predefined characters. Do not replace line breaks if set as the default value.

0 (replace with "\r\n" (0x000D, 0x000A)) - the line break is replaced with "\r\n".

1 (replace with "\n" (0x000A)) - the line break is replaced with "\n".

2 (do not replace) - the line break is not replaced.

3 (remove line breaks) - the line breaks are removed.

1.1.25 ExportDataRecords (Panels, Comfort Panels, RT Advanced)**Description**

Exports one or all data records of a recipe to a CSV file of a TXT file.

One file is created per recipe.

Use in the function list

ExportDataRecords (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecords (Recipe_number/name, Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

File name

Name of the file to which the recipe data records are exported. Enter the storage location and the file extension (*.CSV or *.TXT), for example "C:\TEMP\Orange.csv".

For Basic Panel, enter the file name as follows: "\USB_X60.1\<Name>"

For additional devices:

- If a memory card is used as storage location, specify the storage location as follows: "\StorageCard\<Name>"
- If you only specify one file name and no path, the file will be saved in a system directory, for example, "C:\Documents and Settings\[User]".
- When only a path and no file name is specified, the file name is automatically created from the recipe name. It is a requirement that the folder "D:\Data", for example, has been created in the specified folder. If the folder "D:\Temp" has not been created, the folder name will be used as the prefix of the file name, Temp_Recipe names.csv.

Overwrite

Determines whether an existing export file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: Export file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Export is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithPrompting) = With confirmation: Export file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

You want to export all data records using a key.

Notes on configuring

Configure the "ExportDataRecords" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv (for Basic Panels "\USB_X60.1\orange.csv")
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file. If the file already exists, it will be overwritten.

After exporting the data records, a system alarm is output.

1.1.26 ExportDataRecordsWithChecksum (Panels, Comfort Panels, RT Advanced)

Description

Exports one or all data records of a recipe to a CSV file and generates a checksum for each line in the file.

One file is created per recipe.

Use in the function list

ExportDataRecordsWithChecksum (Recipe number/name, Data record number/name, File name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ExportDataRecordsWithChecksum (Recipe_number/name, Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which the data records are exported. Specify "0" if recipe data records are to be exported from all available recipes.

Data record number/name

Number or name of the recipe data record to be exported. Specify "0" if all recipe data records are to be exported.

File name

Name of the CSV file to which the recipe data records are exported. Enter the path and the file extension e.g. "C:\TEMP\Orange.CSV".

If a storage card is used as storage location, specify the storage location as follows: "\StorageCard\<File name>".

If you define only a file name without specifying a path, the file is saved to the directory from which Runtime was started. If write access to this directory is not enabled in the Windows 7 operating system, the file is saved to the "VirtualStore" folder of the user directory.

When only one path for the export is specified, the file name is automatically created from the respective recipe name. This requires, for example, that the directory "D:\Temp\" has been created. If the directory "D:\Temp" does not exist, the directory name is used as the prefix for the file name, Temp_Recipe names.csv.

Overwrite

Determines whether an existing CSV file with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: CSV file will not be overwritten. The export process will not be carried out.

1 (hmiOverwriteAlways) = Yes: CSV file is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: CSV file is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the export:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Export format of the recipe data records

If ".csv" is chosen as a file extension for the export file, then only valid characters from the ANSI character set will be supported. This also applies to separators in decimal numbers and list elements. The separators used are defined in the country/regional settings of the operating system of the exporting computer.

You may also set "Unicode text" (".txt") format for the export file. This file format supports the WinCC and WinCC Runtime character set. Again, the separators used are specified in the country/regional settings of the operating system of the exporting computer. This file format always uses tab separators in list elements.

The corresponding file import function also supports the ".csv" and ".txt" (Unicode) file formats.

Application example

Using a key, you want to export all data records and assign a checksum.

Notes on configuring

Configure the "ExportDataRecordsWithChecksum" system function on the "Press" event for the desired key. Transfer the following parameters:

- Recipe number/name = 1
- Data record number/name = 0
- File name = c:\temp\orange.csv
- Overwrite = 1
- Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data records are exported.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated. All data records of Recipe 1 are exported to the orange.csv file and assigned checksums. If the file already exists, it will be overwritten.

After exporting the data records, a system message is output.

1.1.27 ExportImportUserAdministration (Panels, Comfort Panels, RT Advanced)

Description

Exports all users of the user administration of the currently active project to the given file or imports the users from the given file into the currently active project.

Users, their passwords and rights are saved in the user administration.

All users are overwritten when importing. The imported users are valid immediately.

Note

Exporting and importing of the user administration from V14 is not supported on HMI devices with a Runtime version older than V14.

Use in the function list

ExportImportUserAdministration (File name, Direction)

Use in user-defined functions

ExportImportUserAdministration (File_name, Direction)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

File name

Name of the file which contains the passwords or to which the passwords are written. Enter the file location and the file extension (*.txt), for example, "C:\TEMP\Passwords.txt".

Note

If a storage card is used as file location, specify the file location as follows: "\\StorageCard \<FileName>".

Direction

Specifies whether passwords are exported or imported:

0 (hmiExport) = Export: Passwords are exported.

1 (hmiImport) = Import: Passwords are imported.

1.1.28 GetBrightness (Panels, Comfort Panels, RT Advanced)

Description

Reads the value of the brightness.

Use in the function list

GetBrightness (Brightness)

Use in user-defined functions

GetBrightness (Brightness)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Brightness

The tag to which the value is written.

1.1.29 GetDataRecordFromPLC (Panels, Comfort Panels, RT Advanced)

Description

Transfers the selected recipe data record from the PLC to the storage medium of the HMI device.

Use in the function list

GetDataRecordFromPLC (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

GetDataRecordFromPLC (Recipe_number/name, Data_record_number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data records are transferred.

Data record number/name

Number or name of the recipe data record which is transferred from the PLC to the data medium of the HMI device.

Overwrite

Determines whether an existing recipe data record with the same name is overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data record is not overwritten. The transfer process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data record is overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data record is not overwritten until confirmed.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to transfer a data record from the PLC to the data medium of the HMI device using a key.

Notes on configuring

Configure the "GetDataRecordFromPLC" system function on the "Press" event for the desired key. Transfer the following parameters:

Recipe number/name = 1

Data record number/name = 1

Overwrite = 1

Output status message = 1

Tags can also be specified in place of the constants. Depending on the configuration, the operator can enter the desired values in the I/O field or read from the PLC. In this way, the operator can determine which recipe data record is transferred from the PLC.

Procedure on HMI device

As soon as the key is activated, the system function is triggered. The constants or tags are evaluated and the first data record of Recipe 1 is transferred from the PLC to the data medium of the HMI device. If the recipe data record already exists, it will be overwritten.

A system message is output after the transfer.

1.1.30 **GetDataRecordName (Panels, Comfort Panels, RT Advanced)**

Description

Writes the name of the given recipe and recipe data record to the given tags.

Note

If the recipe or the recipe data record do not exist, wildcards ("###") are written to the tags.

Note

Only internal tags or external tags are supported as tags.

Use in the function list

GetDataRecordName (Recipe number, Data record number, Recipe name, Data record name, Processing status)

Use in user-defined functions

GetDataRecordName (Recipe_number, Data_record_number, Recipe_name, Data_record_name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Recipe number**

Number of the recipe whose name is written to the given tag.

Data record number

Number of the recipe data record whose name is written to the given tag.

Recipe name

The tag to which the recipe name is written. The tag must be of the STRING type.

Data record name

The tag to which the name of the recipe data record is written. The tag must be of the STRING type.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Application example

You want to output the names of the displayed recipes and the name of the displayed recipe data record on the HMI device.

Configure the following tags:

- RecNumber" of type INTEGER
- RecDataNumber" of type INTEGER
- RecName" of type STRING
- RecDataName" of type STRING

Configure a recipe view with the tags "RecNumber" for the recipe number and "RecDataNumber" for the data record number.

Configure the "GetDataRecordName" system function on the "Press" event for a button and pass the following parameters:

- Recipe number: RecNumber
- Data record number: RecDataNumber
- Recipe name: RecName
- Data record name: RecDataName

Configure two output fields and connect these to the "RecName" and "RecDataName" tags.

Select a recipe and a data record number from the recipe view. As soon as the button is activated, the system function is triggered and the name of the recipe and the recipe data record are displayed in both output fields.

1.1.31 GetDataRecordTagsFromPLC (Panels, Comfort Panels, RT Advanced)

Description

Transfers the values of the recipe data record loaded in the PLC to the corresponding recipe tags.

This system function is used, for example, during teach-in mode on a machine.

Use in the function list

GetDataRecordTagsFromPLC (Recipe number/name, Processing status)

Use in user-defined functions

GetDataRecordTagsFromPLC (Recipe_number/name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe data record whose values are written from the PLC to the tags.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.32 GetGroupNumber (Panels, Comfort Panels, RT Advanced)

Description

Reads the number of the group to which the user logged on to the HMI device belongs, and writes it to the given tag.

Use in the function list

GetGroupNumber (Tag)

Use in user-defined functions

GetGroupNumber (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Tag**

The tag to which the number of the group is written.

1.1.33 GetPassword (Panels, Comfort Panels, RT Advanced)**Description**

Writes the password of the user currently logged on to the HMI device in the given tag.

Note

Make sure that the value of the given tag is not displayed in another place in the project.

Note

The passwords of SIMATIC Logon users cannot be read.

Note

The "GetPassword" system function is not available for devices as of device version 14.0.0.0.

If this system feature is used in a project older than version V14, it is removed after migration to V14. The event linked to the system function is thereby invalidated. The "GetPassword" system function appears to be faulty in the configuration. A warning is generated during compiling. The "GetPassword" system function is no longer included in the compiled Runtime project.

Use in the function list

GetPassword (Tag)

Use in user-defined functions

GetPassword (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the password is written.

1.1.34 GetUserName (Panels, Comfort Panels, RT Advanced)

Description

Writes the user name of the user currently logged on to the HMI device in the given tag.

If the given tag has a control connection, the user name is also available in the PLC connection. This system function makes it possible, for example, to implement a user-dependent release of certain functionalities.

Use in the function list

GetUserName (Tag)

Use in user-defined functions

GetUserName (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the user name is written.

1.1.35 GoToEnd (Panels, Comfort Panels, RT Advanced)

Description

Executes the key function <End> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToEnd

Use in user-defined functions

GoToEnd

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.36 GoToHome (Panels, Comfort Panels, RT Advanced)**Description**

Executes the key function <Home> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToHome

Use in user-defined functions

GoToHome

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.37 ImportDataRecords (Panels, Comfort Panels, RT Advanced)**Description**

Imports one or all data records of a recipe from a CSV file or a TXT file.

When a path is specified, all records of the given directory are imported.

Use in the function list

ImportDataRecords (File name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

ImportDataRecords (File_name, Data_record number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

File name

Name of the file from which the recipe data records are imported. Enter the storage location and the file extension (*.CSV or *.TXT), for example "C:\TEMP\Orange.csv".

For Basic Panel, enter the file name as follows: "\USB_X60.1\<Name>"

For additional devices: If a memory card is used as storage location, specify the storage location as follows: "\StorageCard\<Name>"

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithPrompting) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Configurable objects

Object	Event
Tag	Value change Upper limit exceeded Lower limit violated
Function key (global)	Release Press
Function key (local)	Release Press
Screen	Loaded Cleared
Screen object	Press Release Click Change (or toggle for switch) Switch on Switch off Enable Disable
Scheduler	Time expired

1.1.38 ImportDataRecordsWithChecksum (Panels, Comfort Panels, RT Advanced)

Description

Imports one or all data records of a recipe from a CSV file with a checksum and verifies the checksum.

Use in the function list

ImportDataRecordsWithChecksum (File name, Data record number/name, Overwrite, Output status message, Processing status, verify checksum)

Use in user-defined functions

ImportDataRecordsWithChecksum (File_name, Data_record number/name, Overwrite, Output_status_message, Processing_status, Verify_Checksum)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**File name**

Name of the CSV file from which the recipe data records are imported. Enter the path and the file extension, for example, "C:\TEMP\Orange.CSV".

If a storage card is used as storage medium, specify the storage location as follows:
"\StorageCard\<File name>".

If you specify a directory instead of an individual CSV file, all files in the specified directory will be imported.

Data record number/name

Number or name of the recipe data record to be imported. Specify "0" if all recipe data records are to be imported.

Overwrite

Determines whether existing recipe data records are to be overwritten:

0 (hmiOverwriteForbidden) = No: Recipe data records are not overwritten. The import process will not be carried out.

1 (hmiOverwriteAlways) = Yes: Recipe data records are overwritten without prompting.

2 (hmiOverwriteWithConfirmation) = With confirmation: Recipe data records are not overwritten until confirmed.

Output status message

Determines whether a status message is output after the import:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example to delay execution of other system functions, until this system function has been successfully completed.

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

Verify checksum

Determines if the checksum should be verified during import:

0 (hmiFalse) = No: Checksum is not verified.

1 (hmiTrue) = Yes: Checksum is verified.

1.1.39 IncreaseTag (Panels, Comfort Panels, RT Advanced)

Description

Adds the given value to the value of the tags.

$X = X + a$

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, help tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the given value is added.

Value

The value that is added.

1.1.40 InverseLinearScaling (Panels, Comfort Panels, RT Advanced)

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function $X = (Y - b) / a$.

1.1 System functions (Panels, Comfort Panels, RT Advanced)

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InvertLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n", varX);
...
}
```

The saved return value can be processed in the following code.

1.1.41 InvertBit (Panels, Comfort Panels, RT Advanced)

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the Boolean tag bStatus and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Invert variable
invertBit(bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
    ...
}
```

1.1.42 InvertBitInTag (Panels, Comfort Panels, RT Advanced)**Description**

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Tag**

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
    ...
}
```

1.1.43 LinearScaling (Panels, Comfort Panels, RT Advanced)**Description**

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function $Y = (a * X) + b$.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, an auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

The saved return value can be processed in the following code.

1.1.44 LoadDataRecord (Panels, Comfort Panels, RT Advanced)

Description

Loads the given recipe data record from the memory medium of the HMI device in the recipe tags. This system function is used, for example, to display a recipe data record in the recipe screen.

Activate the "Synchronize recipe view and recipe tags" option in the synchronization settings of the recipe. If this option is deactivated, the system function has no effect.

Use in the function list

LoadDataRecord (Recipe number/name, Data record number/name, Processing status)

Use in user-defined functions

LoadDataRecord (Recipe_number/name, Data_record_number/name, Confirmation, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which a recipe data record is loaded.

Data record number/name

Number or name of the recipe data record to be loaded.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.45 LogOff (Panels, Comfort Panels, RT Advanced)**Description**

Logs off the current user on the HMI device.

Use in the function list

Logoff

Use in user-defined functions

Logoff

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.46 Logon (Panels, Comfort Panels, RT Advanced)**Description**

Logs on the current user on the HMI device.

Use in the function list

Logon (Password, User name)

Use in user-defined functions

Logon (Password, User_name)

1.1 System functions (Panels, Comfort Panels, RT Advanced)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Password

The tag from which the password for the user logging on is read.

If the user is logged on, the password in the tag is deleted.

User name

The tag from which the user name for the user logging on is read.

1.1.47 **LookupText (Panels, Comfort Panels, RT Advanced)**

Description

Identifies an entry from a text list. The result depends on the value and on the selected runtime language. The result is saved to a tag of data type "String".

Use in the function list

LookupText (Output text, Index, Language, Text list)

Use in user-defined functions

LookupText (Output_text, Index, Language, Text_list)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Output text

The tag to which the result is written.

Index

The tag that defines the value of the list entry.

Language

Defines the runtime language in which the list entry is identified.

- Runtime language
Language code as per VBScript reference, for example, "de-DE" for German (Germany) or "en-US" for English (United States of America). The selection depends on the activated runtime languages.
- Tag
The tag that contains the language. Enter the runtime language as decimal value of the country ID, for example, 1031 for German - Standard, 1033 for English - USA. Details are available in the VBScript basics, "Locale identifier (LCID) diagram".
- Integer
The number that corresponds to the order of runtime languages for language switching. The selection depends on the activated runtime languages, for example, "0" for the language that appears when you first start runtime. You can find more information under the topic of "Languages in Runtime".

Text list

Defines the text list. The list entry is read from the text list.

1.1.48 NotifyUserAction (Panels, Comfort Panels, RT Advanced)**Description**

This system function is used to log user actions that are not automatically logged in the Audit Trail. You can also use this system function to require the user to enter an acknowledgment or an electronic signature and a comment for the operator action. A requirement for the use of the system function is that the GMP-compliant configuration is activated under "Runtime settings > GMP".

If you use the "NotifyUserAction" system function in a function, the debugger may open if you cancel your input by clicking "Cancel". To compensate for this reaction, you can use the "On Error Resume Next" statement in a function. With this instruction, the next instruction is executed after a runtime error. If you use the "On Error Resume Next" statement, output of system events is also suppressed.

Use in the function list

NotifyUserAction (Confirmation type, Mandatory commenting, Category, Object name, Description)

Use in user-defined functions

NotifyUserAction (Confirmation_type, Mandatory_commenting, Category, Object_name, Description)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Confirmation type

Establishes how the action must be confirmed

0 = (None): No confirmation required, an entry is created in the Audit Trail

1 = (Acknowledgement): Acknowledgment, the user must acknowledge the action; an entry is created in the Audit Trail

2 = (Digital Signature): Electronic signature; a dialog window opens in which the user must enter the electronic signature - an entry is created in the Audit Trail

Mandatory commenting

Establishes if the user has to enter a comment. The comment is logged in the Audit Trail.

0 = (True): True; a dialog window opens in which the user must enter a comment

1 = (False): False; no comment required

Category

Category or class name of the modified object

Object name

Name of the modified object

Description

Text which describes the archiving user action.

1.1.49 OpenAllLogs (Panels, Comfort Panels, RT Advanced)

Description

Reestablishes the connection between WinCC and the logs. Logging can be continued.

Note

Run the "StartLogging" system function in order to restart logging.

Use in the function list

OpenAllLogs

Use in user-defined functions

OpenAllLogs

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. You open all logs with the "Open Archive" button. Logging will be continued in the specified log.

1.1.50 OpenCommandPrompt (Panels, Comfort Panels, RT Advanced)**Description**

Opens a Windows system prompt.

This function is used, e.g., to copy files or to call up another application.

Use in the function list

OpenCommandPrompt

Use in user-defined functions

OpenCommandPrompt

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.51 OpenControlPanel (Panels, Comfort Panels, RT Advanced)**Description**

Opens a dialog that you can use to edit selected Control Panel settings.

This system function lets you set the following on the HMI device:

- Properties and value of the IP address
- User identification on the network
- WinCC Internet settings

Note**Project security**

The "OpenControlPanelDialog" system function is used to bypass the SecureMode on the HMI device. Take the corresponding precautions to ensure the security of your project.

Use in the function list

OpenControlPanelDialog (Dialog)

Use in user-defined functions

-

Parameters**Dialog**

Sets the Control Panel dialog to be opened.

- PROFINET_X1: Setting of the IP address and Ethernet parameters
- PROFINET_X3: Setting of the IP address and Ethernet parameters; only for Comfort Panel KP 1500, TP 1500; TP1900, TP2200
- WinCC Internet settings: Setting for Web Server, e-mail notification, provided the HMI device supports these functions
- Network ID: Setting for identification on the network, provided the HMI device supports these functions

1.1.52 OpenInternetExplorer (Panels, Comfort Panels, RT Advanced)

Description

Opens the Internet Explorer on the HMI device.

If the Internet Explorer is already open, it will be closed and reopened when you call the system function.

Note

The Internet Explorer saves data temporarily in the DRAM file system of the HMI device, e.g. the last web sites that were be called up.

This data can be saved using the "BackupRAMFileSystem" system function so that it is still available when the HMI device is restarted.

Use in the function list

OpenInternetExplorer (Start page)

Use in user-defined functions

OpenInternetExplorer (Start_page)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Start page

The page which is loaded when Internet Explorer is started, e.g. "www.siemens.com".

1.1.53 OpenScreenKeyboard (Panels, Comfort Panels, RT Advanced)

Description

Hides or shows the screen keyboard.

The screen keyboard remains open until the screen keyboard is explicitly closed. In this way, the screen keyboard can also be used in other applications.

Use in the function list

OpenScreenKeyboard (Display mode)

Use in user-defined functions

OpenScreenKeyboard (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Specifies whether the window is opened minimized or maximized with the screen keyboard:

0 (hmiScreenKeyboardMinimized) = Minimized

1 (hmiScreenKeyboardMaximized) = Maximized

1.1.54 OpenTaskManager (Panels, Comfort Panels, RT Advanced)

Description

Shows the task manager.

The task manager allows changing to other open applications on the HMI device.

Note

The appearance of the task manager depends on the operating system installed.

Use in the function list

OpenTaskManager

Use in user-defined functions

OpenTaskManager

Can be used if the configured device supports user-defined scripts. For additional information, refer to "Device dependency".

Parameter

--

1.1.55 PageDown (Panels, Comfort Panels, RT Advanced)

Description

Executes the key function <Pagedown> on the HMI device.

This system function can only be used for function keys.

Use in the function list

PageDown

Use in user-defined functions

PageDown

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

1.1.56 PageUp (Panels, Comfort Panels, RT Advanced)

Description

Executes the key function <PageUp> on the HMI device.

This system function can only be used for function keys and tasks with a time trigger.

Use in the function list

PageUp

Use in user-defined functions

PageUp

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

1.1.57 PrintReport (Panels, Comfort Panels, RT Advanced)

Description

Prints the given report from the printer which is connected to the HMI device. The report is printed in the language which is set on the HMI device.

Note

If runtime is closed whilst log data are being printed using the system function, the report will cease to be supplied with data as soon as runtime stops.

Use in the function list

PrintReport (Report)

Use in user-defined functions

PrintReport (Report)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Report

Name of the report to be printed.

Note

If you have set up via the "Function list" dialog box a new report for the "PrintReport" function, when compiling, the following warning appears: "The report "Report_1" has no printed pages."

In order to remedy the warning, open the "Report_1" via the project view and recompile the project.

1.1.58 PrintScreen (Panels, Comfort Panels, RT Advanced)

Description

Prints the screen currently being displayed on the HMI device from the printer which is connected to the HMI device.

Opened windows are also printed.

Note

The printer settings are taken over from the current settings in the Windows operation system.

Use in the function list

PrintScreen

Use in user-defined functions

PrintScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

1.1.59 ResetBit (Panels, Comfort Panels, RT Advanced)**Description**

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Tag**

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code sets the value of the Boolean tag bStatus to 0 by means of the ResetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
  BOOL bStatus = 1;
  BOOL bSaved = bStatus;

  //Reset bit
  ResetBit (bStatus);

  //print current and saved value
  printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

1.1.60 ResetBitInTag (Panels, Comfort Panels, RT Advanced)**Description**

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Tag**

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit to 0 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult

//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
    ...
}
```

1.1.61 SafelyRemoveHardware (Panels, Comfort Panels)**Description**

Checks whether there is read or write access to the external storage medium. If there is no access, the external storage medium can be removed without data loss.

Use in the function list

SafelyRemoveHardware(Path, Result)

Use in user-defined functions

SafelyRemoveHardware(Path, Result)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Result**

The tag in which the result is entered.

TRUE: The storage medium can be removed safely. A corresponding system alarm will be output.

FALSE: The storage medium cannot be removed safely. A corresponding system alarm will be output.

Path

Path of storage medium, e.g. \Storage Card USB\

1.1.62 SaveDataRecord (Panels, Comfort Panels, RT Advanced)**Description**

Saves the current values of the recipe tags as data record to the memory medium of the HMI device.

This system function is used, for example, to save a recipe data record in the recipe screen.

Use in the function list

SaveDataRecord (Recipe number/name, Data record number/name, Overwrite, Output status message, Processing status)

Use in user-defined functions

SaveDataRecord (Recipe_number/name, Data_record_number/name, Overwrite, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Recipe number/name

Number or name of the recipe to which a recipe data record is saved.

Data record number/name

Number or name of the recipe data record to be saved. A new data record will be created if no record of this name or number was found in the recipe, independent of the value at the "Overwrite" parameter.

Overwrite

Specifies whether an existing data record is overwritten:

0 (hmiOverwriteForbidden) = No: The recipe data record is not overwritten, the data record is not saved.

1 (hmiOverwriteAlways) = Yes: The recipe data record is overwritten without a prompt for confirmation.

2 (hmiOverwriteWithConfirmation) = With confirmation: The recipe data record is overwritten only with confirmation by the user.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.63 SendEMail (Panels, Comfort Panels, RT Advanced)

Description

Sends an e-mail from the HMI device to the given addressee.

This system function is used, for example, when, in the case of service, the alarm is to be passed on directly to the service technician.

Note

To send alarms as e-mails, the HMI system must have an e-mail client at its disposal.

Use in the function list

SendEMail (Address, Subject, Text, Return address)

Use in user-defined functions

SendEMail (Address, Subject, Text, Return_address)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Address**

The e-mail address of the addressee.

Subject

The subject line of the e-mail.

Text

The text sent in the e-mail.

Return address

The e-mail address to which the addressee of this e-mail should send the reply.

1.1.64 SetAcousticSignal (Panels, Comfort Panels, RT Advanced)**Description**

Configures the acoustic feedback of touch screen operation on the HMI device.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetAcousticSignal (Volume)

Use in user-defined functions

SetAcousticSignal (Volume)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Volume**

Determines whether and how loud an acoustic signal is emitted:

-1 (hmiToggle) = Toggle: Toggles the emission of the acoustic signal as follows: Muted > Quiet > Loud.

0 (hmiMuted) = Mute: no acoustic signal

1 (hmiQuiet) = Quiet: quiet acoustic signal

2 (hmiLoud) = Loud: loud acoustic signal

1.1.65 SetAlarmReportMode (Panels, Comfort Panels, RT Advanced)**Description**

Switches the automatic reporting of alarms on the printer on or off.

Use in the function list

SetAlarmReportMode (Mode)

Use in user-defined functions

SetAlarmReportMode (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Mode**

Determines whether alarms are reported automatically on the printer:

0 (hmiDisablePrinting) = Reporting off: Alarms are not printed automatically.

1 (hmiEnablePrinting) = Reporting On: Alarms are printed automatically.

-1 (hmiToggle) = Toggle: Toggles between the two modes.

1.1.66 SetAndGetBrightness (Panels, Comfort Panels, RT Advanced)

Description

Determines the brightness of the MP 377 Touch daylight readable display. The brightness value can be interpreted as absolute or relative to the current value.

Note

The configuration that is set in the Control Panel will be reestablished when you restart the HMI device.

Use in the function list

SetAndGetBrightness (Brightness, Mode, Current value)

Use in user-defined functions

SetAndGetBrightness (Brightness, Mode, Actual brightness)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Brightness

New value for the brightness.

Mode

Specifies if the new brightness value is set as absolute or relative to the current value.

Current value

Tag in which the current brightness value is stored.

1.1.67 SetBit (Panels, Comfort Panels, RT Advanced)

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the Boolean tag bStatus to 1 by means of the SetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Set bit
SetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

1.1.68 SetBitInTag (Panels, Comfort Panels, RT Advanced)

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=2
SetBitInTag(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
...
}
```

1.1.69 SetBrightness (Panels, Comfort Panels, RT Advanced)

Description

Determines the brightness of the display.

Note

The configuration that is set in the Control Panel / Start Center will be reestablished when you restart the HMI device.

For Basic Panels 2nd Generation, Mobile Panels and Comfort Panels:

The value for the system function "SetBrightness" can be set between 0% and 100%. The set value is transferred to the HMI device. The brightness settings on the HMI device can be viewed and edited in "Start Center > Settings > Display". The HMI devices support a brightness setting between 10% and 100%.

If the system function "SetBrightness" is assigned a value of 0%, the display of the HMI device is switched off by default in Runtime. If the operator touches the display, the display switches to the previous brightness setting.

If the system function "SetBrightness" is assigned a value between 1% and 10% and the operator opens the display settings in the Start Center, brightness is reset to 10%.

Use in the function list

SetBrightness (value)

Use in user-defined functions

SetBrightness (Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Value

New value for the brightness.

1.1.70 SetConnectionMode (Panels, Comfort Panels, RT Advanced)

Description

Connects or disconnects the given connection.

Note

A connection to the PLC cannot be established until the "Online" operating mode has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

If the connection is loaded in "Offline" mode, the connection is closed again each time the mode switches to "Offline". To re-establish the connection after switching to "Online" mode, set the connection again to "Online" mode.

Use in the function list

SetConnectionMode (Mode, Connection)

Use in user-defined functions

SetConnectionMode (Mode, Connection)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether a connection to the PLC is established or disconnected:

0 (hmiOnline) = online: Connection is established.

1 (hmiOffline) = offline: Connection is disconnected.

Connection

The PLC to which the HMI device is connected. You specify the name of the PLC in the connection editor.

Multiple use of the system function in a user-defined function

If you use the "SetConnectionMode" system function for different connections, it may be possible that not all system functions are executed correctly. Proceed as follows to prevent this situation:

1. Create a "BOOL" type tag with the start value "0".
2. Configure the "SetConnectionMode" system function on the "Value change" event of the HMI tags. If you want to disconnect three connections, for example, you must configure the system function three times.
3. In the user-defined function, apply the "InvertBit" system function to the HMI tag.

Application example

Two typical application examples for this system function are as follows:

- **Test**
As long as no PLC is connected to the HMI device, no error messages will be output during the test on the HMI device. If the HMI device is connected to a PLC, the connection to the PLC can be established by pressing a key.
- **Commissioning**
Several PLCs are to be configured for a system. At first, all PLCs except one are configured "Offline". After commissioning of the first PLC, the connection to each of the other PLCs is established by pressing a key. In this way, the other PLCs are started up one after another.

1.1.71 SetDataRecordTagsToPLC (Panels, Comfort Panels, RT Advanced)

Description

Transfers the values of the recipe tags to the PLC. The recipe tags contain the values of the data record which is displayed on the HMI device.

Use in the function list

SetDataRecordTagsToPLC (Recipe number/name, Processing status)

Use in user-defined functions

SetDataRecordTagsToPLC (Recipe_ number/name, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Recipe number/name

Number or name of the recipe from which recipe data record is transferred to the PLC.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.72 SetDataRecordToPLC (Panels, Comfort Panels, RT Advanced)**Description**

Transfers the given recipe data record directly from the data medium of the HMI device to the PLC with which the HMI device is connected.

Note

The values of the recipe data record don't need to be displayed on the HMI device.

Use in the function list

SetDataRecordToPLC (Recipe number/name, Data record number/name, Output status message, Processing status)

Use in user-defined functions

SetDataRecordToPLC (Recipe_number/name, Data_record_number/name, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Recipe number/name**

Number or name of the recipe from which recipe data record is transferred to the PLC.

Data record number/name

Number or name of the recipe data record to be transferred to the PLC.

Output status message

Determines whether a status message is output after the transfer:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.73 SetDaylightSavingTime (Panels, Comfort Panels, RT Advanced)

Description

The system function "SetDaylightSavingTime" changes the setting of the HMI device from daylight saving to standard time and vice versa.

Time settings will take place immediately following system function.

Note

The "SetDaylightSavingTime" system function does not support time zones without daylight saving time.

Note**Windows 7**

The system function "SetDaylightSavingTime" is not supported for PC-based HMI devices under Windows 7.

Use in the function list

SetDaylightSavingTime(DaylightSavingTime)

Use in user-defined functions

SetDaylightSavingTime (Daylight_saving_time)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Daylight Saving Time**

Determines whether Daylight Saving Time is set on the HMI device:

1.1 System functions (Panels, Comfort Panels, RT Advanced)

0 = Daylight Saving Time is not activated

1 = Daylight Saving Time is activated

1.1.74 SetDeviceMode (Panels, Comfort Panels, RT Advanced)

Description

Toggles the operating mode on the HMI device. The following types of operation are possible: "Online", "Offline" and "Loading".

Use in the function list

SetDeviceMode (Operating mode)

Use in user-defined functions

SetDeviceMode (Operating_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Operating mode

Determines the operating mode of the HMI device:

0 (hmiOnline) = online: The connection to the PLC is established. The configured connection status is always set in this process. The states that were last used in Runtime are not considered.

1 (hmiOffline) = offline: The connection to the PLC is disconnected.

2 (hmiTransfer) = Load: A project can be transferred from the configuration computer to the HMI device.

Note

If you use a PC as an HMI device, the runtime software will be exited when you switch operating mode after "Load".

1.1.75 SetDisplayMode (Panels, Comfort Panels, RT Advanced)

Description

Changes the settings of the screen in which the runtime software runs.

The runtime software runs in full-screen mode as default. The Windows task switch is disabled.

Use in the function list

SetDisplayMode (Layout)

Use in user-defined functions

SetDisplayMode (Display mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines the settings for the screen in which the runtime software runs.

1 (hmiScreenFull): Full-screen: Title bar of the screen is not visible

2 (hmiScreenMaximized): Maximized

3 (hmiScreenRestore): Restore: The last used screen setting is used. This layout can only be used when the window is displayed minimized or maximized.

4 (hmiScreenMinimized): Minimized

5 (hmiScreenAutoAdjust): Automatic: The size of the window is set so that all screen objects in it will be visible.

6 (hmiScreenOnTop): Foreground; either the window appears in the foreground or the program icon associated with the window flashes on the taskbar depending on the Windows setting. The setting can be changed in the Windows configuration and applies to all Windows applications.

1.1.76 SetLanguage (Panels, Comfort Panels, RT Advanced)

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage (Language)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.
- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).
An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

1.1.77 SetPLCDateTime (Panels, Comfort Panels, RT Advanced)

Description

Changes the data and the time of the linked PLC

The system function "SetPLCDateTime" can only be configured for the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

Use in the function list

SetPLCDateTime (connection, time)

Use in user-defined functions

-

Parameters

Connection

Connection of PLC and HMI device.

Time

Transfers the date and the time of the HMI device to the PLC. The PLC applies the date and the time of the HMI device.

1.1.78 SetRecipeTags (Panels, Comfort Panels, RT Advanced)**Description**

Changes the status of the recipe tags from "Online" to "Offline" and vice versa.

This system function is used, for example, when recipe data record values are fine tuned when starting up a machine.

Use in the function list

SetRecipeTags (Recipe number/name, Status, Output status message, Processing status)

Use in user-defined functions

SetRecipeTags (Recipe_number/name, Status, Output_status_message, Processing_status)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Recipe number/name**

Number or name of the recipe in which the recipe data record is saved.

Status

Determines the status of the recipe tags:

0 (hmiOnline) = Online: Value changes of the recipe tags are transferred immediately to the PLC connected to the HMI device.

1 (hmiOffline) = Offline: Value changes to the recipe tags are only transferred to the PLC connected to the HMI device when, for example, the "SetDataRecordTagsToPLC" system function is executed.

Output status message

Determines whether a status message is output after saving:

0 (hmiOff) = Off: Status message is not output.

1 (hmiOn) = On: Status message is output.

Processing status

Returns the processing status of the system function. You can use the return value, for example, to delay execution of other system functions until this system function has been successfully completed:

2 = System function is being performed.

4 = System function was successfully completed.

12 = System function was not performed because an error has occurred.

1.1.79 SetScreenKeyboardMode (Panels, Comfort Panels, RT Advanced)**Description**

Enables or disables the automatic display of the screen keyboard on the HMI device.

This system function is also used to prevent the display of the screen keyboard, e.g. because an external keyboard is connected to the HMI device.

Note

To enable the "SetScreenKeyboardMode" ("SetScreenKeyboardMode") system function on an HMI other than a Touch Panel device, set the "Use on-screen keyboard" check box in the "Runtime settings" dialog of the device settings.

Use in the function list

SetScreenKeyboardMode (Mode)

Use in user-defined functions

SetScreenKeyboardMode (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Mode**

Determines whether the screen keyboard is hidden or shown:

0 (hmiOff) = Off: Screen keyboard is hidden

1 (hmiOn) = On: Screen keyboard is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes.

1.1.80 SetTag (Panels, Comfort Panels, RT Advanced)

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );
    bvalue = GetTagBit("gs_tag_bit");
    printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

1.1.81 ShiftAndMask (Panels, Comfort Panels, RT Advanced)

Description

This system function converts the input bit pattern of the source tags into an output bit pattern of the target tags. This involves bit shifting and masking.

Note

If the source and target tag have a different number of bits, using the system function in the target tag can result in a violation of the value range.

Use in the function list

ShiftAndMask (Source tag, Target tag, Bits to shift, Bits to mask)

Use in user-defined functions

ShiftAndMask (Source_tag, Target_tag, Bits_to_shift, Bits_to_mask)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Source tag

The tag includes the input bit pattern. Integer-type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The actual value 72 is set at the 16-bit integer source tag: 0000000001001000.

Target tag

The output bit pattern is saved in the tag. Integer type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The shifted input bit pattern is multiplied by the bit mask, with bit-by-bit logical AND operation: 0000000000001001. The resultant decimal value "8" is saved to the target tag.

Please note the following:

- The source and target tags have the same number of bits.
- The number of bits to shift is less than the number of bits in the source tag and target tag.
- Bits to mask does not have more bits than the source tag and the target tag.

Bits to shift

Number of bits by which the input bit pattern is shifted right. A negative value shifts the input bit pattern to the left.

Example: "Bits to shift" has the value "+3". The input bit pattern is shifted right by three bits when the system function is called: 0000000000001001.

Bits to the left are padded with "0". Three bits are truncated on the right. The new decimal value is "9".

Note

The left bit is "1" in a source tag of the data type with negative signed integer. This sign bit is padded with "0" when the bits are shifted right. The sign changes to "+".

Bits to mask

An integer serves as bit mask. The bit pattern is used to multiply the shifted input bit pattern. Example: Integer "2478" with the bit pattern "0000100110101110".

You can enter the bit mask in three different ways:

- Hexadecimal: First enter the prefix "0h" or "0H", followed by an optional space for better readability. Then group the bit pattern in blocks of four (0000)(1001)(1010)(1110) and set each block in hexadecimal code: (0)(9)(A)(E). Only the characters 0-9, A-F, a-f are allowed: "0h 09AE".
- Binary: First enter the prefix "0b" or "0B", followed by an optional space for better readability. Then group the binary bit pattern into blocks of four 0000 1001 1010 1110 with spaces in between as a check. Only the characters "0" or "1" are allowed: "0b 0000 1001 1010 1110".
- Decimal: Enter the value "2478" directly, without a prefix.

Note

If you change the device version of the target HMI device after configuration (e.g. "13.1.0" to "13.0.0" or vice versa), you must check and test the parameters of this system function.

You can use the "Char" and "Word" data types for the "Source tag" and "Target tag" parameters as of device version 13.1.0. In the device versions before 13.1.0, these parameters must be assigned other data types:

- Use "SInt" instead of "Char"
- Use "Int" instead of "Word"

Otherwise, there may be unwanted effects, for example incorrect or unexpected behavior of the configured system functions.

1.1.82 ShowAlarmWindow (Panels, Comfort Panels, RT Advanced)

Description

Hides or shows the alarm window on the HMI device.

Use in the function list

ShowAlarmWindow (Object name, Display mode)

Use in user-defined functions

ShowAlarmWindow (Object_name, Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Object name

Name of the alarm view which is hidden or shown.

Display mode

Determines whether the alarm window is hidden or shown:

0 (hmiOff) = Off: Alarm view is hidden

1 (hmiOn) = On: Alarm view is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

1.1.83 ShowOperatorNotes (Panels, Comfort Panels, RT Advanced)**Application**

Displays the tooltip configured for the selected object.

If the system function is configured on a function key, the tooltip for the screen object that currently has the focus is displayed. If a tooltip is configured for the screen itself, you can switch to this text by pressing <Enter> or by double-clicking on the help window.

If the system function is configured on a button, only the tooltip for the current screen is displayed. If a tooltip is configured on the button itself, initially only the tooltip for the button is displayed. You can press <Enter> or double-click on the help window to switch to the tooltip for the current screen.

Note

No other screen object can be used while the help window is open. To use the screen object, close the help window.

Closing the help window

You can close the help window in the following ways:

Using the keys:

- By pressing the <HELP> key again
- By pressing the <ESC> key

Using the touch screen:

- By pressing the  button

Use in the function list

ShowOperatorNotes (Layout)

Use in user-defined functions

ShowOperatorNotes (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines whether the configured tooltip is hidden or shown:

0 (hmiOff) = Off: Configured tooltip is hidden

1 (hmiOn) = On: Configured tooltip is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

1.1.84 ShowPopUpScreen (Panels, Comfort Panels, RT Advanced)

Description

Opens the pop-up-screen, for example, when a button is pressed.

You can enter a constant value or assign a tag as coordinates.

If the configured pop-up screen is not visible or only partially visible, the coordinates are set to 0.0.

Regardless of the size of the permanent area, the start position of the coordinates is always 0.0.

Use in the function list

ShowPopupScreen (Name of the screen, X coordinate, Y coordinate, Display mode, Animation, Animation speed)

Use in user-defined functions

ShowPopupScreen (Name of the Screen, X coordinate, Y coordinate, Display mode, Animation, Animation speed)

Parameters

Name of the screen

Specifies the name of the pop-up screen that appears in Runtime when a button is pressed.

X coordinate

Position of the pop-up screen in the current screen on the X axis

Y coordinate

Position of the pop-up screen in the current screen on the X axis

Display mode

Specifies the mode for the pop-up screen:

Switching

Off

On

Animation

Specifies the direction from which the pop-up screen is shown:

Off

Left

Top

Right

Bottom

Animation speed

Specifies the speed at which the pop-up screen is shown:

Slow

Medium

Fast

1.1.85 ShowPopupScreenSizable (Panels, Comfort Panels, RT Advanced)

Description

Opens the pop-up screen with a defined size. You can open a pop-up screen in a different size with this system function; the scroll bars are displayed for navigating in the content of the pop-up screen.

You can enter a constant value or assign a tag as coordinates. If the configured pop-up screen is not visible or only partially visible, the coordinates are set to 0.0. Regardless of the size of the permanent area, the start position of the coordinates is always 0.0.

You also define the displayed width and height of the pop-up screen, which can deviate from the sizes of the configured pop-up screen. The scroll bars are displayed in the pop-up screen in this case.

Note**Operation on HMI devices with keys**

You scroll in pop-up screens using the keyboard shortcut <ALT>+<arrow key>.

Display only one scroll bar

If you specify only one of the sizes smaller than the size of the configured pop-up screen in the parameters of the system function, the two scroll bars are displayed in the pop-up screen. You have two options to configure only one scroll bar in the pop-up screen:

- Reduce the width/height of the configured pop-up screen by the size of the device-specific scroll bar.
- Specify a value that corresponds to the actual size of the pop-up screen for one of the size parameters and add the device-specific size of the scroll bar. You specify the value for the other size as needed.

Example:

You want to open a pop-up screen with the configured size of 500 x 420 in the size 500 x 300 with a vertical scroll bar. Specify the value 534 for the width parameter which corresponds to the actual width of the configured pop-up screen plus the size of the scroll bar of 34. Enter the value 300 for the height. The pop-up screen is opened in the size 500 x 300 in Runtime and only the vertical scroll bar is displayed.

Device type (version V14)	Width	Height
HMI devices with 4" display Comfort Panels with 19" display Runtime Advanced	25	25
HMI devices with 9" display Comfort Panels with 15" display Comfort Panels with 22" display	27	27
HMI devices with 7" display HMI devices with 12" display	34	34

Use in the function list

ShowPopupScreenSizable (Name of the screen, X coordinate, Y coordinate, Width, Height, Display mode, Animation, Animation speed)

Use in user-defined functions

ShowPopupScreenSizable (Name of the Screen, X coordinate, Y coordinate, Width, Height, Display mode, Animation, Animation speed)

Parameters

Name of the screen

Specifies the name of the pop-up screen that appears in Runtime when a button is pressed.

X coordinate

Position of the pop-up screen in the current screen on the X axis

Y coordinate

Position of the pop-up screen in the current screen on the X axis

Width

Specifies the width of the pop-up screen. The maximum width must not exceed the screen width of the configured HMI device.

Height

Specifies the height of the pop-up screen. The maximum height must not exceed the screen width of the configured HMI device.

Display mode

Specifies the mode for the pop-up screen:

Switching

Off

On

Animation

Specifies the direction from which the pop-up screen is shown:

Off

Left

Top

Right

Bottom

Animation speed

Specifies the speed at which the pop-up screen is shown:

Slow

Medium

Fast

1.1.86 ShowSlideInScreen (Panels, Comfort Panels, RT Advanced)

Description

Calls the slide-in screen, for example, when operating a button.

Use in the function list

ShowSlideinScreen (screen name, mode)

Use in user-defined functions

ShowSlideInScreen (SlideInScreen_name, Mode)

1.1 System functions (Panels, Comfort Panels, RT Advanced)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Specifies the slide-in screen that appears in Runtime when a button is pressed:

Top Slide-in screen

Bottom Slide-in screen

Left Slide-in screen

Right Slide-in screen

Mode

Specifies the mode for the slide-in screen:

Switching

Off

On

1.1.87 ShowSoftwareVersion (Panels, Comfort Panels, RT Advanced)

Description

Hides or shows the version number of the runtime software.

Use this system function if during servicing, for example, you required the version of the runtime software used.

Use in the function list

ShowSoftwareVersion (Display mode)

Use in user-defined functions

ShowSoftwareVersion (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines whether the version number is shown:

0 (hmiOff) = Off: Version number is not shown

1 (hmiOn) = On: Version number is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

1.1.88 ShowSystemAlarm (Panels, Comfort Panels, RT Advanced)

Description

Displays the value of the parameter passed as a system event to the HMI device.

Use in the function list

ShowSystemAlarm (Text/value)

Use in user-defined functions

ShowSystemAlarm (Text/value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Text/Value

The text or the value, which was output as a system alarm.

1.1.89 ShowSystemDiagnosticsWindow (Panels, Comfort Panels)

Description

Hides or shows the system diagnostic window on the HMI device. The system diagnostic view is only available in the global screen for Comfort Panels and for WinCC Runtime Advanced.

Use in the function list

ShowSystemDiagnosticsWindow (Screen object)

Use in user-defined functions

ShowSystemDiagnosticsWindow (Target_Object_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen object

Name of the system diagnostic window which is hidden or shown.

1.1.90 StartLogging (Panels, Comfort Panels, RT Advanced)

Description

Starts the logging of data or alarms in the specified log. The function can also be applied to audit trails.

You can interrupt logging at runtime using the "StopLogging" system function.

Use in the function list

StartLogging (Log type, Log)

Use in user-defined functions

StartLogging (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail

Log

Name of the log which is started.

1.1.91 StartNextLog (Panels, Comfort Panels, RT Advanced)

Description

Stops the logging of data or alarms for the given log.

Logging is continued in the next log of the segmented circular log you configured for the specified log.

If you did not configure a segmented circular log for the specified log, the system function has no effect.

Use in the function list

StartNextLog (Log type, Log)

Use in user-defined functions

StartNextLog (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

Log

Name of the log for which the logging is stopped and continued in the next log.

1.1.92 StartProgram (Panels, Comfort Panels, RT Advanced)

Description

Starts the specified program on the HMI device.

The runtime software continues running in the background. Alarms continue to be output and data continues to be updated.

When the given application is exited, the screen which was active during the performance of the system function is displayed on the HMI device.

This system function is used, for example, to edit recipe data records in MS Excel on the HMI device.

Note

If Windows CE is installed on the HMI device, during the configuration it must be checked whether the desired application can be started with this system function.

This system function allows all applications to be started which can be started in the "Execute" dialog of Windows CE.

The application to be started must be installed on the HMI device.

Use in the function list

StartProgram (Program name, Program parameters, Display mode, Wait for program to end)

Use in user-defined functions

StartProgram (Program_name, Program_parameters, Display_mode, Wait_for_program_to_end)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**Program name**

Name and path of the program which is started. Upper and lower case are taken into account in this parameter.

Note

If the path contains spaces, the program can only be started correctly if the path is specified in inverted commas, e.g. "C:\Program Files\START\start.exe".

Program parameters

The parameters you transfer at the start of the program, for example a file that is opened after the start of the program.

The description of the necessary parameters is found in the documentation of the program to be started.

Display mode

Determines how the program window is displayed on the HMI device:

0 (hmiShowNormal) = Normal

1 (hmiShowMinimized) = Minimized

2 (hmiShowMaximized) = Maximized

3 (hmiShowMinimizedAndInactive) = Minimized and inactive

Wait for program to end

Determines whether there is a change back to the project after the called up program has ended:

0 (hmiNo) = No: No change to project

1 (hmiYes) = Yes: Change to project

Note

The "Wait for program to end" parameter is only available for Runtime Advanced and Panels.

1.1.93 StopLogging (Panels, Comfort Panels, RT Advanced)

Description

Stops the logging of process values or alarms in the specified log. The function can also be applied to audit trails.

The "StartLogging" system function is used to resume logging at runtime.

Note

When logging is stopped, a connection between WinCC and the log files or log database still exists. Use the "CloseAllLogs" system function to disconnect this connection.

Use in the function list

StopLogging (Log type, Log)

Use in user-defined functions

StopLogging (Log_type, Log)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Log type

Determines the type of log:

0 (hmiTagArchive) = Data log

1 (hmiAlarmArchive) = Alarm log

2 (hmiAudittrailArchive) = Audit trail

Log

Name of the log that is stopped.

Application example

You are in runtime and want to change the data medium on which the process values are logged.

Notes on configuring

Configure the "StopLogging" and "CloseAllLogs" system functions on the "Close Archive" button.

Configure the "OpenAllLogs" and "StartLogging" system functions on the "Open Archive" button.

As parameter transfer the respective name of the log that is to be stopped and started.

Procedure on HMI device

When the button "Close Archive" is pressed, the specified log is stopped and all open logs are closed. The data medium can be changed. The "Open Archive" button opens all logs and continues logging in the specified log.

1.1.94 StopRuntime (Panels, Comfort Panels, RT Advanced)

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Mode**

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{  
  
//Stop runtime and shutdown  
StopRuntime (hmiStopRuntimeAndOperationSystem);  
  
}
```

The saved return value can be processed in the following code.

1.1.95 TerminatePROFIsafe (Panels, Comfort Panels, RT Advanced)

Description

Disconnects the PROFIsafe connection for fail-safe operation between a KTP Mobile Panel and the PLC.

After execution of the system function "TerminatePROFIsafe", the connector of the KTP Mobile Panel can be removed from the PLC without the system signaling an error.

Use in the function list

TerminatePROFIsafe

Use in user-defined functions

TerminatePROFIsafe

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

--

1.1.96 WinACMPGetVersion (Panels, Comfort Panels)

Description

Reads out the value of the version number of WinAC MP.

Use in the function list

WinACMPGetVersion (Version, Action)

Use in user-defined functions

WinACMPGetVersion (Version, Action)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Version

The tag that contains the value.

Action

Determines whether the version number is read out:

0 (SwitchOff) = Off: Version number is not read out.

1 (SwitchOn) = On: Version number is read out.

1.1.97 WinACMPSetStartAtBoot (Panels, Comfort Panels)

Description

Determines whether or not WinAC MP is started automatically after startup of the HMI device.

Use in the function list

WinACMPSetStartAtBoot (Start at boot)

Use in user-defined functions

WinACMPSetStartAtBoot (Start at boot)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Start at boot

Specifies whether WinAC MP is started automatically.

0 (StartAtBootOff) = Off: WinAC MP is not started on startup of the HMI device.

1 (StartAtBootOn) = On: WinAC MP is started automatically on startup of the HMI device.

1.1.98 WinACSetStartMode (Panels, Comfort Panels)

Description

Sets the operating mode of WinAC MP after startup of the HMI device.

Use in the function list

WinACMPSetStartMode (Start mode)

Use in user-defined functions

WinACSetStartMode (Autostart)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Start mode

Defines whether the Autostart function of WinAC MP is activated.

0 (AutoStartOff) = Off: After startup, WinAC MP remains in the STOP operating mode.

1 (AutoStartOn) = On: After startup, WinAC MP changes to the operating mode it was in before it was closed.

1.2 System functions (RT Professional)

1.2.1 ActivateScreen (RT Professional)

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent area or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent area when the system function is called up, the permanent area maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The following program code activates "Screen_2" with the ActivateScreen function, for example, when you click a button.

```
Sub ActivateScreen_2()  
  
'User-defined code  
' i. e. when pressing a button  
  
ActivateScreen "Screen_2",0
```

1.2.2 ActivateScreenInScreenWindow (RT Professional)

Description

Performs a screen change to the specified screen in a specified screen window.

Use in the function list

ActivateScreenInScreenWindow (Screen name, Screen window, New screen name)

Use in user-defined functions

ActivateScreenInScreenWindow (Screen_name, Screen_window, New_screen_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Screen name

Name of the screen to be displayed in the screen window.

Screen window

Name of the screen window in which the new screen is to be displayed.

New screen name

Name of the new screen to be displayed in the screen window.

Example

The ActivateScreenInScreenWindow function is used in the following program code to activate the "Screen_2" screen when you click a button.

```
{
// User defined code
// i.e. when pressing a button
ActivateScreenInScreenWindow (GetParentScreen(screenName),
GetParentScreenWindow(screenName), "Screen_2");
...
}
```

1.2.3 DecreaseTag (RT Professional)

Description

Subtracts the given value from the tag value.

$X = X - a$

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function for events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual tag value is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

Example

The following program code decrements the value at the varX tag by the value at the value tag. The value entered is saved to the old_value tag and output along with the new varX value.

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Decrease tag
DecreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function DecreaseTag: %i\r\n", old_value, varX);
...
}
```

1.2.4 ExportImportUserAdministration (RT Professional)

Description

Exports all users of the user administration of the currently active project to the given file or imports the users from the given file into the currently active project.

Users, their passwords and rights are saved in the user administration.

All users are overwritten when importing. The imported users are valid immediately.

Note

Exporting and importing of the user administration from V14 is not supported on HMI devices with a Runtime version older than V14.

Use in the function list

ExportImportUserAdministration (File name, Direction)

Use in user-defined functions

ExportImportUserAdministration (File_name, Direction)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter**File name**

Name of the file which contains the passwords or to which the passwords are written. Enter the file location and the file extension (*.txt), for example, "C:\TEMP\Passwords.txt".

Note

If a storage card is used as file location, specify the file location as follows: "\StorageCard \<FileName>".

Direction

Specifies whether passwords are exported or imported:

0 (hmiExport) = Export: Passwords are exported.

1 (hmiImport) = Import: Passwords are imported.

1.2.5 GetParentScreen (RT Professional)

Description

The function determines the name of the parent screen from a specified screen path.

Use in the function list

GetParentScreen (screen)

Use in user-defined functions

GetParentScreen (Screen)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen

The syntax of the transferred "Screen" parameter must correspond to that formed by the graphics system for the screen paths:

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

If you use the object list in the function list to specify the parameter, the screen name is entered instead of the screen path.

Note

The "." character is used to distinguish between screen and screen object names. The ":" character is used for the hierarchy structuring. Therefore, use only the "-" or "_" as a delimiter in the name designations.

Return value

Name of the parent screen.

Principle

A "Screenwindow_1" screen window is in a "Screen_1" screen. In the screen window, the screen becomes "Screen_2" with the "Screenwindow_2" screen window, etc.

Screen path: Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen returns:

- "Screen_2", if the system function is called in the "Screen_3" screen.
- "Screen_1", if the system function is called in the "Screen_2" screen.

GetParentScreenwindow returns:

- "Screenwindow_2", if the system function is called in the "Screen_3" screen.
- "Screenwindow_1", if the system function is called in the "Screen_2" screen.

1.2.6 GetParentScreenWindow (RT Professional)

Description

The function determines the name of the parent screen window from a transferred screen path.

Use in the function list

GetParentScreenWindow (Screen)

Use in user-defined functions

GetParentScreenWindow (Screen)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen

The syntax of the transferred "Screen" parameter must correspond to that formed by the graphics system for the screen paths:

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

If you use the object list in the function list to specify the parameter, the screen name is entered instead of the screen path.

Note

The "." character is used to distinguish between screen and screen object names. The ":" character is used for the hierarchy structuring. Therefore, use only the "-" or "_" as a delimiter in the name designations.

Return value

Name of the parent screen window.

Principle

A "Screenwindow_1" screen window is in a "Screen_1" screen. In the screen window, the screen becomes "Screen_2" with the "Screenwindow_2" screen window, etc.

Screen path: Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen returns:

- "Screen_2", if the system function is called in the "Screen_3" screen.
- "Screen_1", if the system function is called in the "Screen_2" screen.

GetParentScreenwindow returns:

- "Screenwindow_2", if the system function is called in the "Screen_3" screen.
- "Screenwindow_1", if the system function is called in the "Screen_2" screen.

1.2.7 IncreaseTag (RT Professional)

Description

Adds the given value to the value of the tags.

$X = X + a$

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function for events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual tag value is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag

The tag to which the given value is added.

Value

The value that is added.

Example

The following program code increments the value of the varX tag by the value in the value tag. The value entered is saved to the old_value tag and output along with the new varX value.

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Increase tag
IncreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function IncreaseTag: %i\r\n", old_value, varX);
...
}
```

1.2.8 InverseLinearScaling (RT Professional)

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function $X = (Y - b) / a$.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, an auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InvertLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n", varX);
...
}
```

The saved return value can be processed in the following code.

1.2.9 InvertBit (RT Professional)

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the Boolean tag bStatus and outputs the result along with the original bSaved value.

```

'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
  BOOL bStatus = 0;
  BOOL bSaved = bStatus;

  //Invert variable
  invertBit(bStatus);

  //print current and saved value
  printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}

```

1.2.10 InvertBitInTag (RT Professional)

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
...
}
```

1.2.11 LinearScaling (RT Professional)**Description**

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function $Y = (a * X) + b$.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n", Yvar);
...
}
```

The saved return value can be processed in the following code.

1.2.12 LookupText (RT Professional)

Description

Identifies an entry from a text list. The result depends on the value and on the selected runtime language. The result is saved to a tag of data type "String".

Use in the function list

LookupText (Output text, Index, Language, Text list)

Use in user-defined functions

LookupText (Output_text, Index, Language, Text_list)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Output text

The tag to which the result is written.

Index

The tag that defines the value of the list entry.

Language

Defines the runtime language in which the list entry is identified.

- **Runtime language**
Language code as per VBScript reference, for example, "de-DE" for German (Germany) or "en-US" for English (United States of America). The selection depends on the activated runtime languages.
- **Tag**
The tag that contains the language. Enter the runtime language as decimal value of the country ID, for example, 1031 for German - Standard, 1033 for English - USA. Details are available in the VBScript basics, "Locale identifier (LCID) diagram".
- **Integer**
The number that corresponds to the order of runtime languages for language switching. The selection depends on the activated runtime languages, for example, "0" for the language that appears when you first start runtime. You can find more information under the topic of "Languages in Runtime".

Text list

Defines the text list. The list entry is read from the text list.

1.2.13 ResetBit (RT Professional)

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Tag**

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code sets the value of the Boolean tag bStatus to 0 by means of the ResetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BOOL bStatus = 1;
BOOL bSaved = bStatus;

//Reset bit
ResetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

1.2.14 ResetBitInTag (RT Professional)

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit to 0 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
...
}
```

1.2.15 SetBit (RT Professional)**Description**

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the Boolean tag bStatus to 1 by means of the SetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Set bit
SetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

1.2.16 SetBitInTag (RT Professional)

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=2
SetBitInTag(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
...
}
```

1.2.17 SetLanguage

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage (Language)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.
- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).
An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

1.2.18 SetPropertyByConstant (RT Professional)

Description

Specifies the value of an object property as a string.

Use in the function list

SetPropertyByConstant (Screen name, screen object, name of the property, value)

Use in user-defined functions

SetPropertyByConstant

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to change the property of a screen, the parameter "Object" must be empty. For this purpose, use the following syntax, for example:

```
SetPropertyByConstant("<Screen name>", "", "<Property name>", "<Value>")
```

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property is changed.

Name of the property

Name of the property that will be changed.

Value

The value assigned to the property.

Example

The SetPropertyByConstant function is used in the following program code to change an object property: In the "Trends" screen, the "ToolBarButtonClick" property of the "Control_1" object is set to the value 26.

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolBarButtonClick", "26"

'User defined code
...
```



```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolBarButtonClick", "26");

// User defined code
...
}
```

Example: Changing a screen property

The SetPropertyByConstant function is used in the following program code to change a screen property: In the "Trends" screen, the "BackColor" property is set to the value 255.

```
'Programming language: VBS
'Name of the picture: Trends

SetPropertyByConstant "Trends", "Trends", "BackColor", "255"

'User defined code
...


{
//Programming language: C
//Name of the picture: Trends

SetPropertyByConstant ("Trends", "Trends", "BackColor", "255");

// User defined code
...
}
```

Alternatively, use the password ZERO or a space-string instead of the second parameter (object).

1.2.19 SetPropertyByProperty (RT Professional)

Description

Specifies the value of an object property with another object property.

Use in the function list

SetPropertyByProperty (Screen name, object, name of the property, target screen name, target object, target property name)

Use in user-defined functions

SetPropertyByProperty (Screen_name, Screen_object, Property_name, Source_screen_name, Source_screen_object, Source_property_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to specify the property of a screen with another screen property, the parameters "Object" and "Destination object" must be empty. For this purpose, use the following syntax, for example:

SetPropertyByProperty ("<Screen name>", "", "<Property name>", "<Destination_Screen name>", "", "<Destination_Property name>")

Parameters

Screen name

Name of the screen that contains the object.

Object

Name of the object whose property value is transferred to the target object.

Name of the property

Name of the property whose value is transferred to the target object.

Target screen name

Name of the screen that contains the destination object.

Target object

Name of the target object for which a property is changed.

Destination property name

Name of the property that will be changed.

Example

The following program code uses the SetPropertyByProperty function to set the property "ToolBarButtonClick" of the object "Control_1" in the original screen Trend_1": on the corresponding property in the destination screen "Trend_2".

```
'Programming language: VBS
'Name of source picture: Trend_1
'Name of target picture: Trend_2
'Name of the f(t) trend view control: Control_1

SetPropertyByProperty "Trend_1", "Control_1", "ToolBarButtonClick", "Trend_2",
"Control_2", "ToolBarButtonClick"

'User defined code
...

{
//Programming language: C
//Name of source picture: Trend_1
//Name of target picture: Trend_2
//Name of the f(t) trend view control: Control_1

SetPropertyByProperty ("Trend_1", "Control_1", "ToolBarButtonClick", "Trend_2",
"Control_2", "ToolBarButtonClick");

// User defined code
...
}
```

1.2.20 SetPropertyByTag (RT Professional)**Description**

Specifies the value of an object property with a tag value.

Use in the function list

SetPropertyByTag (Screen name, screen object, name of the property, tag name)

Use in user-defined functions

SetPropertyByTag (Screen_name, Screen_object, Property_name, Tag_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to specify the property of a screen by means of a tag value, the "Object" parameter must be empty. For this purpose, use the following syntax, for example:

```
SetPropertyByTag("<Screen name>", "", "<Property name>", "<Tag name>")
```

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object containing the property to be set with the tag value.

Name of the property

Name of the property that is set with the tag value.

Tag name

Name of the tag that contains the value of the property.

Example

The SetPropertyByTag function is used in the following program code to change an object property: When you click on the object, the object name and the screen in which the object is located are transferred. The CaptionText in the screen window contains the value of the HMI_value_1 tag.

```
'Programming language: VBS
SetPropertyByTag screenName, objectName, "CaptionText", "HMI_value_1"

'User defined code
...

{
//Programming language: C
SetPropertyByTag (screenName, objectName, "CaptionText", "HMI_value_1");

// User defined code
...
}
```

Example

The SetPropertyByTag function is used in the following program code to change an object property: In the "Trends" screen, the "ToolBarButtonClick" property of the "Control_1" object is set to the value 26.

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolBarButtonClick", "26"

'User defined code
...

{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolBarButtonClick", "26");

// User defined code
...
}
```

1.2.21 SetPropertyByTagIndirect (RT Professional)

Description

Writes the value of an indirect addressed tag to an object property. The tag transferred as parameter contains the name of a second tag the value of which is read.

Use in the function list

SetPropertyByTagIndirect (Screen name, screen object, name of the property, , tag name)

Use in user-defined functions

SetPropertyByTagIndirect (Screen_name, Screen_object, Property_name, Tag_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to specify the property of a screen by means of a tag, the "Object" parameter must be empty. For this purpose, use the following syntax, for example:

SetPropertyByTagIndirect ("<Screen name>", "", "<Property name>", "<Tag name>")

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property you want to change.

Name of the property

Name of the property to be changed.

Tag name

Name of the tag which, in turn, contains the name of the tag whose value is read.

Example

The SetPropertyByTagIndirect function is used in the following program code to change an object property: .

```
'Programming language: VBS
SetPropertyByTagIndirect GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1"

'User defined code
...

{
//Programming language: C
SetPropertyByTagIndirect (GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1");

// User defined code
...
}
```

1.2.22 SetTag (RT Professional)

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );
    bvalue = GetTagBit("gs_tag_bit");
    printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

1.2.23 SetTagByProperty (RT Professional)

Description

Specifies a tag value with the value of an object property. The change is also logged in the alarm system.

Use in the function list

SetTagIndirectByProperty (tag name, screen name, screen object, name of the property, with or without operator event)

Use in user-defined functions

SetTagByProperty (Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of the tag whose value is specified by the object property.

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property supplies the tag value.

Name of the property

Name of the property that supplies the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code returns the value of the selected text when you click in a combo box.

```
{
char* rt_value;

SetTagByProperty (rt_value, screenName, objectName, "SelectedText",
hmiWithoutOperatorEvent);

...
}
```

1.2.24 SetTagByTagIndirect (RT Professional)

Description

Writes the value of an indirect addressed tag to a tag. The tag transferred as parameter contains the name of a second tag the value of which is read. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByTagIndirect (name of the tag, tag name, with or without operator input alarm)

Use in user-defined functions

SetTagByTagIndirect (Tag_name, Source_tag_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters**Name of the tag**

Name of the tag whose value is to be set.

Tag name

Name of a string tag which contains the name of a tag which supplies the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator input alarm

1 (hmiWithOperatorEvent) = With operator input alarm

Example

The following program code writes the value of the tag Tag4 to the tag Tag1.

```
{
SetTag ("IndirectRead", "Tag4");
SetTagByTagIndirect ("Tag1", "IndirectRead", hmiWithoutOperatorEvent);

...
}
```

1.2.25 SetTagIndirect (RT Professional)**Description**

Writes a value to an indirect addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirect (tag name, , value, with or without operator event)

Use in user-defined functions

SetTagIndirect (Tag_name, Value, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Value

Value which is to be written.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code writes the value "value" to the tag Tag3.

```
{
int value;

SetTag ("IndirectWrite", "Tag3");
SetTagIndirect ("IndirectWrite", "value", hmiWithoutOperatorEvent);
...
}
```

1.2.26 SetTagIndirectByProperty (RT Professional)

Description

Writes the value of an object property to an indirectly addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByProperty (Tag name, Screen name, Screen object, Name of the property, With or without operator event)

Use in user-defined functions

SetTagIndirectByProperty (Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property supplies the value.

Name of the property

Name of the property which supplies the value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without input alarm

1 (hmiWithOperatorEvent) = With operator input alarm

Example

The following program code writes the value of the object property "Background color" to the tag Tag2.

```
{
SetTag ("IndirectWrite", "Tag2");
SetTagIndirectByProperty ("IndirectWrite", screenName, objectName, "BackColor",
hmiWithoutOperatorEvent);
...
}
```

1.2.27 SetTagIndirectByTagIndirect (RT Professional)

Description

Writes the value of an indirectly addressed tag to an indirectly addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The tag transferred as parameter contains the name of a second tag the value of which is read. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByTagIndirect (Tag name, Tag name, With or without operator input alarm)

Use in user-defined functions

SetTagIndirectByTagIndirect (Tag_name, Source_tag_name,
With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters**Tag name**

Name of a string tag which contains the name of a tag whose value is to be changed.

Tag name

Name of the indirect tag that returns the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code writes the value of the tag "Tag4" to the tag "Tag2".

```
{  
SetTag ("IndirectWrite", "Tag2");  
SetTag ("IndirectRead", "Tag4");  
SetTagIndirectByTagIndirect ("IndirectWrite", "IndirectRead");  
...  
}
```

1.2.28 SetTagWithOperatorEvent (RT Professional)**Description**

Specifies the value for a tag. The change is also logged in the alarm system.

Use in the function list

SetTagWithOperatorEvent (tag name, value)

Use in user-defined functions

SetTagWithOperatorEvent (Tag_name, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of the tag whose value is to be set.

Value

Value written to the tag.

Example

The following program code transfers the value from the "value" tag to the "result" tag when you click the corresponding button.

```
'Programming language: VBS
SetTagWithOperatorEvent result, value
...
```

```
//Programming language: C
{
SetTagWithOperatorEvent ("result", "value");
...
}
```

1.2.29 ShowBlockInTIAPortalFromAlarm (RT Professional)

Description

Results in a change from the alarm view to the affected program code of the selected project in the engineering system. Changing from the alarm view to the project in the TIA Portal enables further analysis of the program code, for example, search for a missing contact in the interlock.

You configure this system function at a screen object, for example, a button or a button of the toolbar in an alarm view. When you click the button, the necessary context is stored by the system and the corresponding location is automatically displayed in the program code after switching to the engineering system. You use the Read mode parameter to specify which authorization is required to enable the switch to the engineering system.

If you have configured the system function to a button, the TIA Portal is started in the mode that was specified with the write protection parameter. To enable different levels of security when accessing the system function, configure the buttons with two different authorizations:

- For the first button, configure the authorization for unlimited access by setting the Read mode parameter to FALSE.
- For the second button, configure the authorization so that data cannot be changed by setting the Read mode parameter to TRUE.

Use in the function list

ShowBlockInTiaPortalFromAlarm (write protection, offline mode, TIA Portal project path, screen name of alarm view, name of alarm view, error tag)

Use in user-defined functions

ShowBlockInTIAPortalFromAlarm (readOnly, offlineMode, TiaPortalProjectPath, AlarmScreenName, AlarmViewName, ErrorTag)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional".

Parameters

Write protection

Specifies whether the project is opened in read-only mode.

TRUE (default): TIA Portal is opened in read mode, the user cannot make any changes.

FALSE: TIA Portal is opened with read and right authorization.

Offline mode

Specifies whether the project is opened in offline mode.

TRUE: The TIA Portal does not go to online mode after the block is opened.

FALSE (default): Online mode is started after the block is opened.

TIA Portal project path

Path and file name of the project in the TIA Portal to which you change from the alarm view.

Screen name of the alarm view

Name of the screen that contains the alarm view.

Alarm view name

Object name of the alarm view to which you change from the TIA Portal.

Error tag

If an error occurs, information about the error is provided by the error tag.

1.2.30 ShowLogonDialog (RT Professional)

Description

Opens a dialog on the HMI device with which the user can log on to the HMI device.

Use in the function list

ShowLogonDialog

Use in user-defined functions

ShowLogonDialog

Can be used if the configured device supports user-defined functions. Additional information is available under "AUTOHOTSPOT".

Parameters

--

1.2.31 ShowPLCCodeViewFromAlarm (RT Professional)

Description

Switches from the alarm view with the selected ProDiag alarm or selected GRAPH alarm to the PLC code view with the respective network. The screen that contains the PLC code view is either opened in a root screen or in a screen window.

Use in the function list

ShowPLCCodeViewFromAlarm (alarm screen name, alarm view name, basic screen name, screen window name, screen name of the PLC code view, name of the PLC code view, error tag)

Use in user-defined functions

ShowPLCCodeViewFromAlarm (Screen, Alarmview, Screen, ScreenWindow, Screen, PLCCodeView, Error tag)

Can be used if the configured device supports user-defined functions. For more information, refer to "System functions for Runtime Professional".

Parameters

Alarm screen name

Name of the screen that contains the alarm view.

Alarm view name

Object name of the alarm view from which the selected alarm is transferred.

Basic screen name

Name of the screen which includes a screen window in which the screen can be displayed.

Screen window name

Name of the screen window in which the screen with the PLC code view can be displayed.

Screen name of the PLC code view

Name of the screen that contains the PLC code view.

Name of the PLC code view

Object name of the PLC code view.

Error tag

If an error occurs, information about the error is provided by the error tag.

Error messages

The following errors can be reported back using the error tag:

KOPAPI_E_TPSF_ALGCTRL_NOTSELECTED	Exactly one alarm must be selected
KOPAPI_E_TPSF_ONLYONLINELISTALLOWED	Only select alarms from the alarm list of the alarm view
KOPAPI_E_TPSF_ONALARMWRONGPRODUCERID	No GRAPH/ProDiag alarm selected
KOPAPI_E_TPSF_ONALARMNOSD0JUMPFLAG	It is not possible to jump over this GRAPH/ProDiag alarm. The jump from an alarm in the alarm view to the PLC code view is made possible for the following supervision alarms: • With global supervisions, only for interlock supervisions (Interlock) • With local supervisions, for all basic supervisions at input parameters

1.2.32 StopRuntime (RT Professional)

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{  
  
//Stop runtime and shutdown  
StopRuntime (hmiStopRuntimeAndOperationSystem);  
  
}
```

The saved return value can be processed in the following code.

1.3 VBScript for Windows (Panels, Comfort Panels, RT Advanced, RT Professional)

1.3.1 VBScript for Windows (Panels, Comfort Panels, RT Advanced, RT Professional)

VBScript

If you have worked with Visual Basic or Visual Basic for applications, then VBScript will seem familiar to you. If you do not know Visual Basic and are getting familiar with it, then you will learn all Visual Basic programming languages at once. The step-by-step guides from Microsoft Press are a good introduction to programming.

You will find fundamental information on VBScript language elements on the Microsoft homepage:

<http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx>

Local ID (LCID)

An overview of all language codes can be found on the Microsoft homepage:

<http://msdn.microsoft.com/en-us/global/bb964664>

1.4 VBScript for Windows CE (Panels, Comfort Panels)

1.4.1 VBScript for Windows CE (Panels, Comfort Panels)

VBScript for Windows CE

You can also use the full range of functions of VBScript, except the control elements for access to files, on devices with Windows CE.

On a device with Windows CE the "File" and "FileSystem" control elements and the CreateObject function enable access to files and to the file system.

You will find fundamental information on VBScript language elements on the Microsoft homepage:

<http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx>

Local ID (LCID)

An overview of all language codes can be found on the Microsoft homepage:

<http://msdn.microsoft.com/en-us/global/bb964664>

1.4.2 CreateObject (Panels, Comfort Panels)

Function

This function generates a reference to an automation object.

Syntax

CreateObject (object)

Parameters

Object

A string that contains the ProgID of the object to be generated.

Return value

Outputs a reference to an automation object.

Notes

Use the CreateObject function to generate invisible ActiveX controls in Runtime. CreateObject cannot be used to generate graphic objects such as TreeView Control or ListView Control. The objects generated with CreateObject do not respond to events. To generate objects that respond to events, use the CreateObjectWithEvents function. The following table shows the ProgIDs for ActiveX Controls without events:

Control	ProgID
Microsoft CE File control 6.0	.file
Microsoft CE FileSystem control 6.0	.filesystem
Microsoft CE ImageList control 6.0	CEimageList.imageListctrl

Example

```
Dim f, fwModeAppend
Set f = CreateObject("FileCtl.File")
fwModeAppend=8
f.Open "\Storage Card\testfile.txt", fwModeAppend
f.Close
```

1.4.3 Control element (Panels, Comfort Panels)

Control elements for file access

The "File" and "FileSystem" control elements enable access to files and the file system on a device with Windows CE. The control elements can only be used with the Windows CE Toolkit for Visual Basic 6.0.

Library

FILECTLdtl

DLL

Mscefile.dll

File control element

The file control element supports the following properties:

- Attr
- EOF
- Loc
- LOF
- Seek

The File control element supports the following properties:

- Close (File)
- Get
- Input
- Input
- InputFields
- LineInputString
- LinePrint
- Open
- Put
- WriteFields

FileSystem control element

The FileSystem control element supports the following methods:

- Dir
- FileCopy
- FileDateTime
- FileLen
- GetAttr
- Kill
- MkDir
- MoveFile
- Rmdir
- SetAttr

1.4.4 Properties (Panels, Comfort Panels)

1.4.4.1 Attr (Panels, Comfort Panels)

Function

This property of the File Control returns a number which names the file mode that was used to open the file.

Syntax

file.Attr

Parameters

File

Reference to the File Control.

Return values

The values shown in the following table indicate the access mode. If the return value is 0, the file is closed.

Constant	Value
None	0
fsModeInput	1
fsModeOutput	2
fsModeRandom	4
fsModeAppend	8
fsModeBinary	32

Notes

"Attr" is a read-only property. Use the Open method of the File Control to specify the file mode.

1.4.4.2 EOF (Panels, Comfort Panels)

Function

This property outputs True when the end of a file is reached which was opened for random or sequential input.

Syntax

File.EOF

Parameters**File**

Reference to a File Control.

Notes

Use the EOF property to avoid errors that occur due to reading after the end of a file.

The EOF property outputs False until the end of the file is reached. For files which were opened with fsModeRandom or fsModeBinary, False is output until the last executed Get statement returns no complete data record.

For files that were opened in the fsModeBinary mode, an attempt is made to read the file with the input function until EOF outputs True and an error is generated. If you want to read binary data with input, use the properties LOF and LOC instead of EOF or use Get in conjunction with EOF. For files that were opened with fsModeOutput, EOF always outputs True.

1.4.4.3 Loc (Panels, Comfort Panels)**Function**

This property outputs a number that names the current read/write position.

Syntax

file.Loc

Parameters**File**

Reference to a File Control.

Notes

For files which were opened with fsModeRandom, Loc outputs the last entry that was read or written as number. For files which were opened in all other modes, Loc outputs the position of the last read or written byte.

1.4.4.4 LOF (Panels, Comfort Panels)

Function

This property outputs the size of a file, in bytes as number.

Syntax

file.Loc

Parameters

File

Reference to a File Control.

Notes

The LOF property can be used in combination with the Loc property to ensure that a read operation does not extend beyond the end of a file.

1.4.4.5 Seek (Panels, Comfort Panels)

Function

This property sets the next position that is to be read or written in a file and outputs this.

Syntax

file.Seek [= Position]

Parameters

File

Reference to a File Control.

Position

Numerical address that describes the position in a file.

Notes

The Seek property writes the next file position, by contrast the Loc property describes the current file position. The number output for Seek is always one higher than that for Loc. The only exception to this is when a file was opened new; in this case, both Seek and Loc output one.

An error is output if Seek produces a negative value or zero.

1.4.5 Methods (Panels, Comfort Panels)

1.4.5.1 Close (Panels, Comfort Panels)

Function

This method closes an open File Control.

Syntax

```
file.Close
```

Parameters

File

File name of a File Control.

Output values

None.

Notes

Use the Open method to open a file.

1.4.5.2 Dir (Panels, Comfort Panels)

Function

This method outputs the name of a file or a directory that fits in a specific pattern or contains a specific attribute.

Syntax

```
File.Dir (PathName,[Attribute])
```

Parameters

File

Reference to a FileSystem Control

Pathname

Optional: A character string expression that describes the file name or path.

Attribute

Optional: A numerical expression whose sum describes the file attribute. If this was omitted, all files that are in the corresponding path are output.

The following table shows the parameter settings of the attribute.

Constant	Value	Description
fsAttrNormal	0	Normal
fsAttrReadOnly	1	Read-only
fsAttrHidden	2	Hidden
fsAttrSystem	4	System file
fsAttrVolume	8	Volume Label. If this was defined, all other attributes are ignored.
fsAttrDirectory	16	Directory
fsAttrArchive	32	Log

Output values

String. A file name made up of the path name and the attributes. Dir outputs a zero length string ("") if the path name is not found.

Notes

Dir supports the use of multi-character wildcards (*) and single-character wildcards (?) for the description of files that exist multiple times. When used for the first time, you must specify the path name of the Dir method. If you specify attributes, the path name must also be included.

The Dir method outputs the first file name that matches the path name. If you want to output additional file names that match the path name, execute Dir again without parameters. If no further matching file names can be found, Dir returns a zero-length string (""). If a zero-length string was output, specify the path name by subsequent calling.

1.4.5.3 FileCopy (Panels, Comfort Panels)**Function**

This method copies the already existing file to a new file.

Syntax

Filesystem.FileCopy PathName, NewPathName

Parameters**File system**

Reference to a FileSystem object.

PathName

Character string that contains the path and the file name.

NewPathName

Character string that contains the file name and the path of the new file.

Output values

None.

Notes

FileCopy outputs an error if the new file does not exist.

1.4.5.4 FileDateTime (Panels, Comfort Panels)**Function**

This method outputs a Variant (date) that contains the date and the time at which the file was created or last edited.

Syntax

```
filesystem.FileDateTime(pathname)
```

Parameters**File system**

Reference to a FileSystem Control

Pathname

Required. Character string expression that names the file name. The path name may contain a directory.

Output value

Indicates the date on which the file was last edited.

Notes

If a new file does not exist, FileDateTime outputs an error.

1.4.5.5 FileLen (Panels, Comfort Panels)

Function

This method outputs the value that describes the length of file in bytes.

Syntax

Filesystem.FileLen(pathname)

Parameters

File system

Reference to a FileSystem Control

Pathname

Requires a character string that describes the file name. The path name may contain a directory.

Output values

Indicates how many bytes a file consists of.

Notes

If the specified file is already open when FileLen is called, the value indicates the size of the file before this is opened.

1.4.5.6 Get (Panels, Comfort Panels)

Function

This method reads data from an open file into a tag.

Syntax

file.Get Data, [Recnumber]

Parameters

File

Reference to a File Control.

Data

Required. Variant tag into which the data is read.

Recnumber

Record number at which the read process begins. For files that were opened in binary mode, Recnumber defines the byte position.

Output values

None.

Notes

Data that is read with the Get method is usually written to the file with the Put method. The first data record or the first byte in a file is in position 1, the second in position 2 and so on. If you skip Recnumber, the next data record or the next byte that follows the last Get or Put method (or which was referred to by the Seek method) is read.

The following rules apply to files which were opened in random mode:

- If the length of the read data is shorter than the length defined in length clause of the Open method, Get reads the following data records in the length of the data record limit. The space between the end of a data record and the start of the next data record is filled the content of the file buffer. Because it is not possible to precisely determine the scope of the filled data, it is advisable that the length of the data records matches the length of the data to be read.
- If the data is a numeric type variant, Get reads 2 bytes to determine the VarType of the variant and then reads the data that is written to the tag. If, for example, a variant of type VarType3 reads 6 bytes with Get, 2 bytes identify the variant as VarType 3 (long) and 4 bytes contain long data. The data record length is defined in the length clause. When the open method is used, the data record length must be at least 2 bytes larger than the number of bytes required to save the tag.
- The Get method can be used to read a variant array from the memory. However, it cannot be used to read a scalar variant that contains an array. No objects can be read from the memory with Get.
- If the variant that is to be read is type VarType 8 (string), Get reads 2 bytes and identifies the variant as VarType 8, 2 additional bytes define the length of the string and the data of the string is then read. The data record length of the open method defined by the length clause must be at least 4 bytes greater than the length of the string.
- If the tag to be written is a dynamic array, Get reads the descriptor whose length is 2 plus 8 times the number of dimensions ($2 + 8 * \text{NumberOfDimensions}$). The data record length of the open method defined by the length clause must be greater than or equal to the sum of bytes required to read the array data and the array descriptor.

For files which are opened in binary mode, the length clause of the open method has no effect.
Get reads all tags jointly from the memory without filling between the files.

1.4.5.7 GetAttr (Panels, Comfort Panels)

Function

This method outputs a value which describes the number of attributes of a file or a folder.

Syntax

```
filesystem.GetAttr(path name)
```

Parameters

File system

Reference to a FileSystem Control

Pathname

Requires a character string that describes the file name or folder name. The path name may contain a directory.

Output values

Sum of attribute values. The following table shows the possible values.

Constant	Value	Description
vbNormal	0	Normal
vbReadOnly	1	Read-only
vbHidden	2	Hidden
VbSystem	4	System
VbDirectory	16	Directory
VbArchive	32	The file has changed since the last backup.

Notes

To specify the set attributes, use the And operator to execute a bitwise comparison of the values returned by GetAttr and the values of the attribute selected by you.

1.4.5.8 Input (Panels, Comfort Panels)

Function

This method outputs a string that contains the characters from a file opened in input or binary mode.

Syntax

```
file.Input(Number)
```

Parameters

File

Reference to a File Control

Number

Numerical expression that defines the number of output characters.

Output values

A character string that consists of characters read from a file.

Notes

Data that is read with the Input method is generally written to the file with LinePrint or Put methods. Use this method only for files which were opened in input or binary mode.

In contrast to the LineInputString method, the Input method outputs all read characters, for example commas, carriage returns, line feeds, quotation marks and leading spaces.

Any attempt to read the file with the Input method before the EOF function returns "True" will cause an error in files that were opened for binary access. To avoid this error, use the LOF and Loc functions instead of the EOF function to read binary files with the Input method or use Get in combination with the EOF function.

1.4.5.9 InputB (Panels, Comfort Panels)

Function

This method returns bytes from a file opened in input or binary mode.

Syntax

```
file.InputB(number)
```

Parameters

File

Reference to a File Control.

Number

Each valid numerical expression that describes the number of output bytes.

Output values

An array that contains the bytes read out from the file.

Notes

Files that are read with the InputIB method are usually written with the LinePrint or Put methods. Use this method only for files which were opened in Input or Binary mode.

1.4.5.10 InputFields (Panels, Comfort Panels)

Function

The method reads data from an open sequential file and outputs a one-dimensional variant data field.

Syntax

file.InputFields(Number)

Parameters

File

Reference to a File Control.

Number

Number of fields which are delimited by commas and are to be read from a file.

Output values

An array that contains fields read from the file.

Notes

Data that is read with the InputFields method is usually written with WriteFields. Use this method only in files opened in Binary or Input mode. InputFields reads standard string or numeric data without modifications. The following table shows how InputFields reads other input data:

Data	Content of the assigned tags
Comma delimitation or empty line	Empty
#NULL#	Null
#TRUE# or #FALSE#	True or False
#yyyy-mm-dd hh:mm:ss#	The date and/or the time shown as expression

Double quotation marks ("") with input data are discarded.

If you reach the end of the file while you are adding a data object, the added data is deleted and an error is displayed.

To correctly read data from a file as tags that use InputFields, use the WriteFields method instead of the LinePrint method to write the data to the files. The use of WriteFields ensures that each data field is exactly separated.

1.4.5.11 Kill (Panels, Comfort Panels)

Function

This method deletes files from the hard disk.

Syntax

filesystem.Kill Path name

Parameters

File system

Reference to a FileSystem Control

Pathname

Required. A string which names one or more files that are to be deleted. The path name may contain a folder.

Output values

None.

Notes

The Kill method supports the use of multi-character wildcards (*) and single-character wildcards (?) to identify multiple files.

An error is output if you try to delete an open file using the Kill method.

1.4.5.12 LineInputString (Panels, Comfort Panels)

Function

This method reads a single line from an open sequential file and links it with a string tag.

Syntax

file.LineInputString

Parameters

File

Reference to a File Control.

Output values

None.

Notes

Data that is read with LineInput string is usually written to a file with Line Print. The LineInputString method reads a file character by character until a carriage return sequence (Chr(13)) or carriage-return/line-feed (Chr(13) + Chr(10)) is reached. Carriage return and line feed sequences are more often skipped than appended to the character string.

1.4.5.13 LinePrint (Panels, Comfort Panels)

Function

This method writes a single line to an open sequential file.

Syntax

file.LinePrint output

Parameters**File**

Reference to a File Control

Output

A string that is written to a file.

Output values

None

Notes

Data that is written to a file with LinePrint is usually read out with LineInput string.
A carriage return sequence (Chr(13) + Chr(10)) is appended at the end of the string.

1.4.5.14 Mkdir (Panels, Comfort Panels)**Function**

This method creates a new directory.

Syntax

filesystem.Rmdir Pathname

Parameters**File system**

Reference to a FileSystem Control

Pathname

A character string that contains the directory name.

Output values

None.

Notes

Mkdir returns an error if the directory already exists.

1.4.5.15 MoveFile (Panels, Comfort Panels)

Function

The method renames an already existing file or a directory and all subdirectories.

Syntax

filesystem.MoveFile PathName, NewPathName

Parameters

File system

Reference to a FileSystem Control

PathName

A character string that contains the file name.

NewPathName

A character string that contains the file name to be copied.

Output values

None.

1.4.5.16 Open (Panels, Comfort Panels)

Function

The method opens a file. The following file modes are available: Input (1), Output (2), Random (4), Append (8), or Binary (32).

Syntax

file.Open pathname, mode, [access], [lock], [reclength]

Parameters

File

Reference to a File Control.

Pathname

A string that contains the file name.

Mode

Specifies the file mode: Input (1), Output (2), Random (4), Append (8), or Binary (32).

Access

Not permitted with open file: Read, Write, or ReadWrite [default]. (1, 2, 3)

Lock

Actions on the open file that are blocked by other processes: Shared, LockRead, LockWrite [default] and LockReadWrite. (1, 2, 3, 0).

Reclength

A number in bytes that is lower than 32,767. For files that were opened with random access, this value corresponds to the sentence length. For sequential files, this value is the number of buffered characters.

Output values

None.

Note

The reclength parameter is ignored in Binary mode. If a file was opened in Random mode, you must define a file size that is greater than zero; otherwise, an error is output.

1.4.5.17 Put (Panels, Comfort Panels)**Function**

This method writes data from a tag to a file.

Syntax

file.Put data, [recnumber]

Parameters**Data**

Required. A variant tag which contains the data that is to be written to the memory.

Recnumber

Optional. Variant (long). Data record number (random mode files) or byte number (binary mode files) with which the write process starts.

Output values

None.

Notes

Data that is written with Put is usually read out from a file with Get.

The first data record or the first byte in a file is in position 1, the second in position 2 and so on. If you skip Recnumber, the next data record or the next byte that follows the last Get or Put method, or which was referred to by the Seek method, is read.

The following rules apply to files that were opened in random mode:

- If the length of the data to be written is shorter than the length defined in the length clause of the Mopen method, Put writes the following data records in the length of the data record limit. The space between the end of a data record and the start of the next data record is filled with the content of the file buffer. Because it is not possible to precisely determine the length of the filled data, it is advisable that the length of the data records matches the length of the data to be written. If the length of the data to be written is longer than the length of the Open method defined in the length clause, an error is output.
- If the tag to be written is a numerical type variant, Put first writes 2 bytes to declare the variant as VarType and the tag is then written. If, for example, a VarType 3 variant is written, Put writes 6 bytes, 2 bytes identify the variant as VarType 3 (long) and 4 bytes contain the long data. The data record length of the open method defined in the length clause must be at least 2 bytes greater than the number of bytes required to save the tag.
- The Put method may be used to write a variant array to the memory. However, it cannot be used to write a scalar variant that contains an array. Objects cannot be written with Put to the hard disk.
- If the version that is to be written is type VarType 8 (string), Put writes 2 Bytes and identifies the variant as VarType 8, 2 additional bytes define the length of the string and the data of the string is then written. The data record length of the open method defined by the length clause must be at least 4 bytes greater than the length of the string.
- If the tag to be written is a dynamic array, Put writes the descriptor whose length is 2 plus 8 times the number of dimensions ($2 + 8 * \text{NumberOfDimensions}$). The data record length of the open method defined by the length clause must be greater than or equal to the sum of bytes required to write the array data and the array descriptor.

For files which are opened in binary mode, the length clause of the open method has no effect. Put writes all tags jointly from the memory with filling between the files.

1.4.5.18 Rmdir (Panels, Comfort Panels)

Function

This method deletes an empty directory.

Syntax

`filesystem.Rmdir Pathname`

Parameters

File system

Reference to a FileSystem Control

PathName

A string that includes the path name.

Output values

None.

Notes

The directory must be empty in order to be deleted. A complete path must be specified.

1.4.5.19 SetAttr (Panels, Comfort Panels)

Function

This method set the attribute data of a file.

Syntax

filesystem.SetAttr Pathname, Attribute

Parameters

File system

Reference to a FileSystem Control

Pathname

Requires a character string that contains the file name.

Attributes

Requires a numerical expression that contains the sum of the file attribute. The following table shows the possible values.

Constant	Value	Description
vbNormal	0	Standard (default)
vbReadOnly	1	Read-only
vbHidden	2	Hidden
VbSystem	4	System file
VbArchive	32	The file has been modified since the last backup

Output values

None.

Notes

A runtime error occurs if you try to set the attributes of an open file.

1.4.5.20 WriteFields (Panels, Comfort Panels)

Function

This method writes data to a sequential file.

Syntax

file.WriteFields [data]

Parameters

File

Reference to a File Control.

Data

Variant to be written to a file or variant array of a numerical string or a string expression.

Output values

None.

Notes

Data that is written with WriteFields is usually read from the file with InputFiles.

If you omit data, a blank line is written to the file.

- The full stop is written as decimal delimiter for numerical data.
- Either #TRUE# or #FALSE# are output for Boolean data. The True and False keywords are not compiled, regardless of location.
- The time data is written to the file in the universal date format. If either the date or the time is missing or null, only the existing information is written to the file.
- If the data is empty, nothing is written to the file.
- If the data is null, #NULL# is written to the file.

The WriteFields method adds commas and quotation marks around the strings that are written to the file. Delimiters do not have to be added to the list. WriteFields inserts a line break in the

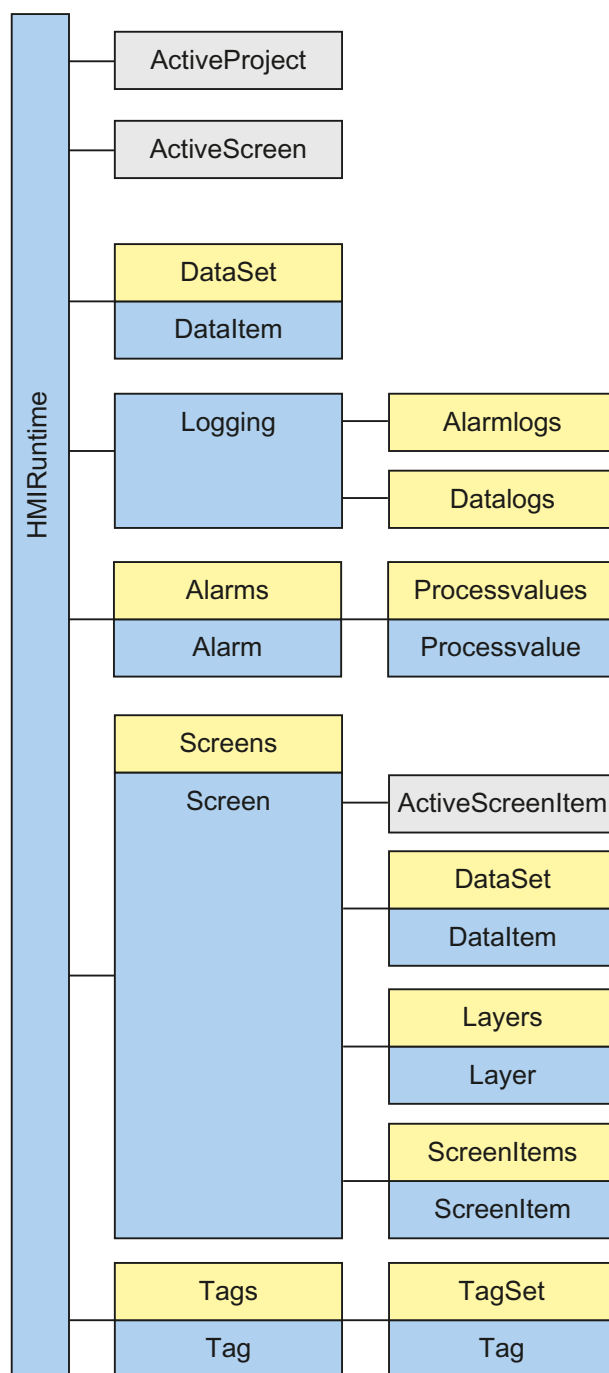
form of a carriage-return/line-feed (Chr(13) + Chr(10))- after it has written the last character of the data to the file.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

1.5.1 VBS object model (RT Professional)

VBS object model in WinCC

The following screen shows the VBS object model in WinCC:



Use the WinCC object model of the graphic Runtime system to access objects and tags in Runtime.

Objects

Objects and lists are provided for access to all the objects in the graphic Runtime systems:

- Display and operating objects
- Screens
- Layers
- Tags

Properties

You can specifically change the display and operating elements and tags in Runtime via the properties of the individual objects. For example, you can enable a button with a click or trigger a color change by changing a tag value.

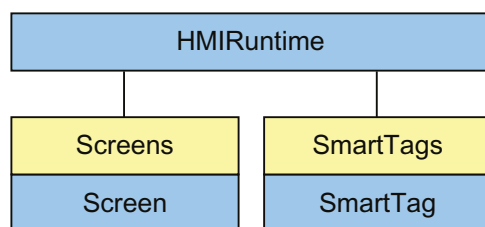
Methods

Methods which are applied to individual objects can be used to read out tag values for further processing or displaying alarms in Runtime.

1.5.2 Objects (Panels, Comfort Panels, RT Advanced)

1.5.2.1 HMIRuntime (Panels, Comfort Panels, RT Advanced)

Description



Portrays the graphic runtime system.

The HMIRuntime object contains properties and methods which return the objects to the main layer, e.g. returns the ActiveScreen property of a screen object.

Application

You use the "HMIRuntime" object, e.g. as follows:

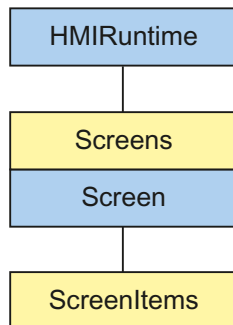
- Read or set the current Runtime language ("Language" property).
- Read name of current base screen or trigger a base screen change by setting a new screen name (property "BaseScreenName")
- Access tags (List "SmartTags")
- End runtime (Method "Stop")
- Output information on sequence tracing output (Method "Trace")
- Access the screens displayed during runtime (List "Screens")

See also

ActiveScreen (Page 497)

1.5.2.2 Screen object (list) (Panels, Comfort Panels, RT Advanced)

Description



A list of screen objects.

The list contains the following two elements:

- The first. Element with the index 0 represents the permanent area.
- The second. Element with the index 1 represents the root screen.

Alternatively, the two elements can be addressed with their their names as follows:

- Permanent area: "Permanent area"
- Root screen: Name of the screen displayed in the root screen

If the named screen is not displayed, an error occurs during access.

Permanent area "Permanent area" is displayed in the Object list and in Auto complete.

Note

The alarm window and the alarm indicator are not contained in the screens list, even if they have the focus in Runtime.

Application

Use the screen property to return the screen list. In the following example, the background color is changed from black to green:

Use the object name as an index.

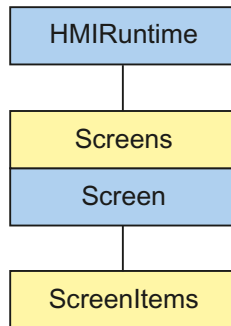
```
'VBS_Example_BackColor  
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

Note

If you perform a screen change, the open references to the screen that is no longer available will become invalid. As a result, it is no longer possible to work with these references.

1.5.2.3 Screen (Panels, Comfort Panels, RT Advanced)

Description



Represents the process screen which is being displayed on the HMI device at the moment or the permanent area in runtime. The screen object is returned to you as a result of accessing the screen list.

The screen object also contains a list of all graphic objects in the screen that can be addressed through the list "ScreenItems".

Application

You can also use the screen object to do the following:

- Read the height and width of a screen (properties "Height" and "Width").
- Change the background color (property "BackColor").

Use the object name as an index.

In the following example, the background color is changed from black to green:

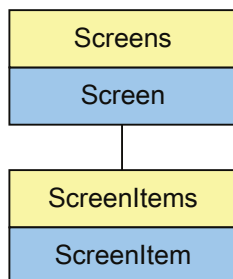
```
'VBS_Example_BackColor  
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

See also

ScreenItem (Page 192)

1.5.2.4 ScreenItem (Panels, Comfort Panels, RT Advanced)

Description



Represents an object in the specified screen. The ScreenItem object is an element of the ScreenItems list.

Application

You can use the ScreenItem object to access the properties of graphic objects within a screen, depending on certain events.

You use the "ScreenItem" object as follows, for example:

- "Visible" property
Switch the visibility of an object on or off.
- "Height" and "Width" properties
Query the width and height of an object.
- "Top" and "Left" properties
Change the position of an object.
- "ObjectName" property
Read the name of a graphic object
- "Parent" property
Set a reference to the parent screen

Use the ScreenItems property to return an object to the screen. Use the object name as an index.

Example

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
'VBS_Example_ScreenItems  
  
Dim objCircle  
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")  
objCircle.BackColor = vbGreen
```

Note

To save memory space in the HMI device, no object names are loaded during transfer of the project. If you still want to transfer object names, call up the Runtime settings for the respective HMI device in WinCC. You can change the setting under "General". The object name is required when the object should be accessed via the object name or for debugging a project.

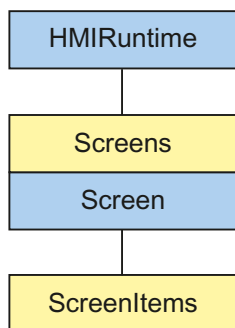
The "ScreenItem" object has different properties depending on its features. The following properties are provided for every "ScreenItem" object:

- Enabled
- Height
- Left
- ObjectName
- Parent
- Top
- Type
- Visible
- Width

If a specific object type is addressed, further properties are added to the standard properties. The additional properties are provided in the descriptions of the individual object types.

1.5.2.5 ScreenItems (Panels, Comfort Panels, RT Advanced)

Description



A list of screen item objects with all screen objects of the given process screen. The list has a parent property. This property provides a reference to the process screen in which the screen object is located.

Application

By means of the "ScreenItems" list you are able

- To edit or output all objects within the list (that is, all objects within a screen)
- To count the objects of a screen (property "Count").
- To work on a particular object in the list (method "Item").

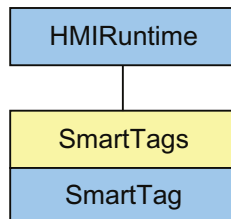
Use the screen items property to return an object from the process screen. Use the object name as an index.

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

1.5.2.6 SmartTags (Panels, Comfort Panels, RT Advanced)

Description



A list of SmartTag objects which represent all of the tags in WinCC Runtime.

Note

The SmartTags list has a limited range of functions. You can only use the tag names to access a SmartTag object. Access via the index or by using "For each" instruction is not supported.

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar  
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Note**Current system behavior for dynamic access to elements of a user data type**

When you compose the tag name of a user data type element dynamically in Runtime, the composed tag name is **not** recognized under the following condition:

- Tag of the data type of the user data type is created.
- Tag is **not** used, e.g. at an I/O field

Example:

You have configured a user data type "Motor" with the elements "RPM" (Int), "On" (Bool) and "Off" (Bool). In the project, you have configured the tag "Motor1" as instance of the user data type. You want to output the values of the elements in only one I/O field in runtime. You have configured an additional I/O field for input of the element name. The value you enter is saved to the internal tag strElementName. With the following script you output the value of an element in the I/O field "IOFieldOutputValue":

```
Dim strDynElementName, objIOFieldOutputValue
Set objIOFieldOutputValue = objscreen.ScreenItems("IOFieldOutput")
'Get element name from tag value
strDynElementName = SmartTags("strElementName").Value
'Create tag name
objIOFieldOutputValue.ProcessValue =
SmartTags("Motor1."+strDynElementName).Value
```

Help:

Configure a separate I/O field for each user data type element, for example, in an additional screen. Interconnect the "process value" with the respective user data type element.

Note**Dynamizing faceplates**

Use the following syntax to access a tag that is connected at a faceplate interface ("Properties"):

```
Smarttags("Properties\int_input1_tag")
```

Application

Use the SmartTags list to return a SmartTag object. Use the tag name to reference the SmartTag object.

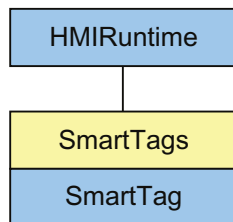
```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

In Runtime Advanced and Panels you address the tag directly using its name. If the tag name corresponds to the VBS name conventions, you do not need to use the SmartTags list. Follow the example below:

```
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(AirPressure)
HMIRuntime.Trace strAirPressure
```

1.5.2.7 SmartTag (Panels, Comfort Panels, RT Advanced)

Description



Represents the value of the specified process tag. The SmartTag object is an element of the SmartTag list.

Application

The SmartTag object provides read and write access to the value of the specified process tag. The SmartTag object does not return an object reference. Use the SmartTags list to return the value of a process tag. Use the tag name as an index.

Note

With the "SmartTags reads values from the cache" setting, values are read from the process image (cache) instead of directly from the controller.

The SmartTag object can also read values directly from the controller. However, you can then expect a substantially higher communication load between the HMI device and the controller.

Example

```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Note

If you want to return the "TypeName" of a SmartTag object data type with the VBS function "TypeName", use the following syntax:

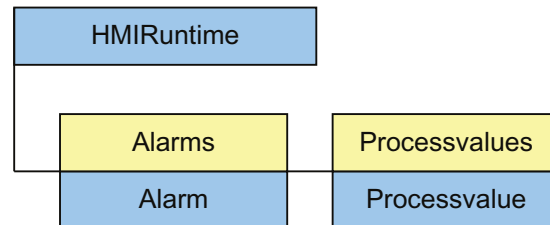
```
TypeName(SmartTags("FillLevel").value)
```

To access the value of an array element, use "SmartTags("<Tag>[Index]")". Set the number of the desired array element for "index", for example, "SmartTags("AirPressure[2]")".

1.5.3 Objects (RT Professional)

1.5.3.1 Alarm (RT Professional)

Description



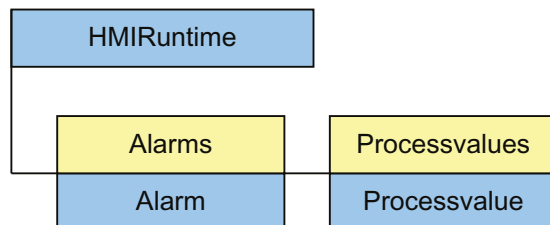
The alarm object is used to access the Alarms object list.

Note

The properties of the alarm object are not automatically updated when the values of the properties change.

1.5.3.2 Alarms (list) (RT Professional)

Description



Use the alarm object to trigger existing messages.

Usage

Using the "Alarms" list you can:

- Access a message in the list (Item method)
- Create a new alarm object (Create method)
- Read the alarm ID of the message (AlarmID attribute)
- Read the status of a message (State property)
- Read the time stamp of the message (Timestamp property)
- Generate an instance of the alarm object (Instance property)
- Read the name of the computer on which the message came (ComputerName property)
- Read or set the name of the user who triggered the message (UserName property)
- Read or set the name of the process value blocks (ProcessValues property)
- Read or set the message commentary (Comment property)
- Read or set the message server prefix (Context property)

Example

In the following example the alarm with the alarm number "1" configured in the "HMI alarms" editor is activated.

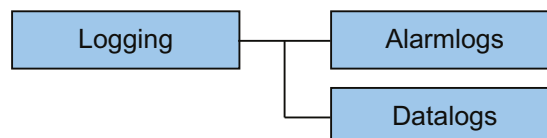
```
'VBS360
Dim MyAlarm
Set MyAlarm = HMIRuntime.Alarms(1)
MyAlarm.State = 5 'hmiAlarmStateCome + hmiAlarmStateComment
MyAlarm.Comment = "MyComment"
MyAlarm.UserName = "Hans-Peter"
MyAlarm.ProcessValues(1) = "Process Value 1"
MyAlarm.ProcessValues(4) = "Process Value 4"
MyAlarm.Create "MyApplication"
```

See also

AlarmID (Page 506)

1.5.3.3 AlarmLogs (list) (RT Professional)

Description



You can use the object to reconnect swapped-out log segments of the alarm log to Runtime or to remove previously swapped-in log segments of the alarm log. The log segments to be swapped in are copied to the "Common logging" folder of the WinCC project. The previously swapped-in log segments are removed from the "Common logging" folder.

You use parameters to control the location from which log segments are to be swapped in. You specify the time period over which the log segments are to be swapped in or removed.

If an error occurs during the operation with log segments, the applied method returns an error alarm.

Usage

- "Restore" method
Previously swapped-out log segments of the alarm log are connected to Runtime.
- "Remove" method
Previously swapped-in log segments of the alarm log are removed from the Runtime project.

Example

In the following example, log segments from the alarm log are swapped in and the return value is output as a trace.

```
'VBS187
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.AlarmLogs.Restore("D:
\Folder","2004-09-14","2004-09-20",-1) & vbNewLine
```

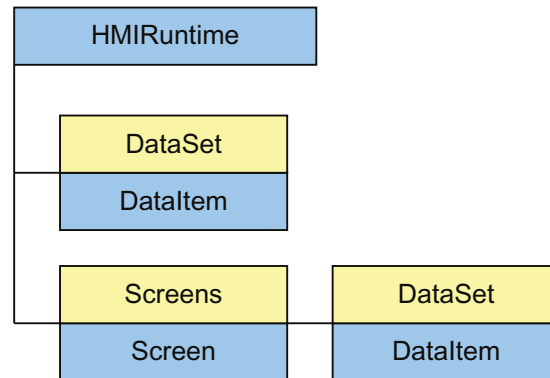
See also

[Restore \(Page 1343\)](#)

[Remove \(Page 1338\)](#)

1.5.3.4 Dataltem (RT Professional)

Description



You can use the Dataltem object to access the contents of the DataSet list. Values or object references are stored in the list as Dataltem.

Access uses the name under which the value was added to the list. Single access using an index is not recommended since the index changes during adding or deleting of values. You can use the index to output the complete contents of the list. The output is in alphabetical order.

Note

For object references, ensure that the objects are capable of multi-threading.

Example

The example shows how the value of 'Motor1' is output as a trace.

```
'VBS163
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
```

The following example enumerates all DataItem objects of the DataSet list. Name and value are output as a trace.

```
'VBS164
Dim data
For Each data In HMIRuntime.DataSet
HMIRuntime.Trace data.Name & ": " & data.Value & vbNewLine
Next
```

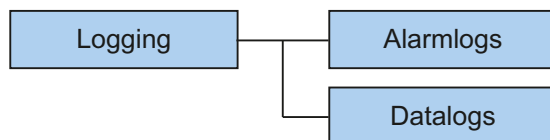
Note

The value may not be output directly for objects.

See also

[Value \(Page 1172\)](#)

[Name \(Page 831\)](#)

1.5.3.5 DataLogs (list) (RT Professional)**Description**

You can use the object to reconnect swapped-out log segments of the data log to Runtime or to delete previously swapped-in log segments of the data log. The log segments to be swapped in are copied to the "Common logging" folder of the WinCC project. The previously swapped-in log segments are removed from the "Common logging" folder.

You use parameters to control the location from which log segments are to be swapped in. You specify the time period over which the log segments are to be swapped in or removed. You also set the type of log ("Fast data log", "Slow data log", "Fast data log and Slow data log").

If an error occurs during the operation with log segments, the applied method returns an error alarm.

Usage

- "Restore" method
Previously swapped out log segments of the data log are connected to Runtime.
- "Remove" method
Previously swapped-in log segments of the data log are removed from the Runtime project.

Example

In the following example, log segments from the Fast data log are swapped in and the return value is output as a trace.

```
'VBS188
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.DataLogs.Restore("D:
\Folder", "2004-09-14", "2004-09-20", -1, 1) & vbNewLine
```

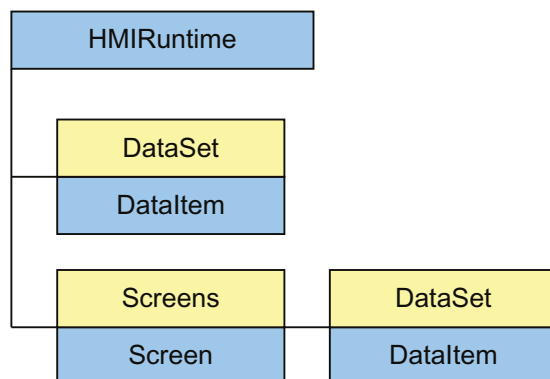
See also

[Restore \(Page 1343\)](#)

[Remove \(Page 1338\)](#)

1.5.3.6 DataSet (list) (RT Professional)

Description



Using the DataSet object, you can exchange data throughout several actions.

A DataSet object is global and defined by the screen object. You can access the data from any VBS action.

You address the screen object according to the screen hierarchy. The DataSet object persists as long as the screen is displayed. The global object persists over the entire runtime time period.

Access uses the Dataltem object.

Note

You cannot include objects with the types Screen, Screens, ScreenItem, ScreenItems, Tag and TagSet in the DataSet list.

The DataSet object does not support any classes.

Application

Use the "DataSet" list as follows:

- Enumerating
Output or process all objects in the list
- "Count" property
Output the number of elements contained

- "Item" method
Work on a particular object in the list
- "Add" method
Add an object to the list
- "Remove" method
Remove a particular object from the list
- "RemoveAll" method
Remove all objects from the list

Access to list elements is made as follows:

```
HMIRuntime.DataSet("Itemname")
```

For a picture-specific list, access is performed as follows:

```
HMIRuntime.Screens("Screenname").DataSet("Itemname")
```

You access the DataSet object of the screen as follows:

```
DataSet("Itemname")
```

If upon access the stated name does not exist in the list, VT_Empty is returned and an Exception is triggered.

Example

The example shows how you check with "Is Nothing" whether the "DataSet" object exists:

```
If HMIRuntime.DataSet("MyDataset") Is Nothing Then
    HMIRuntime.Trace "Requested DataSet-object does not exist"
Else
    HMIRuntime.Trace "Requested DataSet-object exists"
End If
```

The example shows how a value can be entered in the list, read and removed from the list throughout various actions.

```
'VBS162
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")
```

See also

[RemoveAll \(Page 1341\)](#)

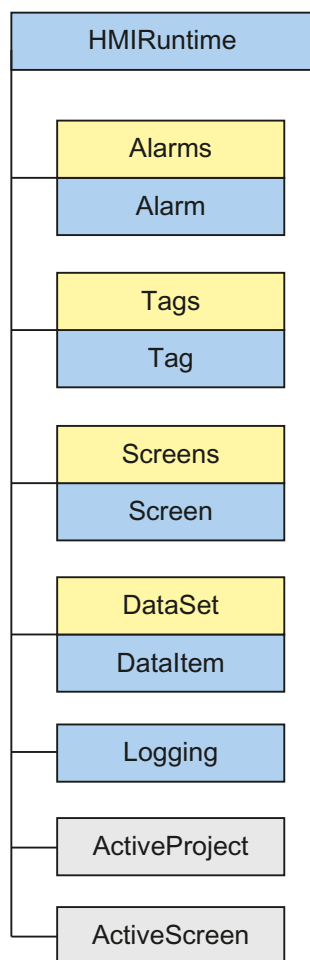
[Remove \(Page 1338\)](#)

Item (Page 1319)

Add (Page 1258)

1.5.3.7 HMIRuntime (RT Professional)

Description



The HMIRuntime object represents the graphic Runtime environment.

Usage

You can use the "HMIRuntime" object as follows:

- "Language" property
Read or set the current Runtime language
- "BaseScreenName" property
Read or set the name of the current root screen
- "ActiveProject" property
Read the path of the active Runtime project
- "Tags" property
Accessing tags
- "DataSet" property
Accessing tags in a list
- "Stop" method
Stop Runtime
- "Trace" method
Display messages in a diagnostics window

Example

The following command closes WinCC Runtime:

```
'VBS3  
HMIRuntime.Stop
```

See also

Trace (Page 1363)
Stop (Page 1363)
Language (Page 763)
Tags (Page 1002)
Logging (Page 798)
DataSet (Page 648)
CurrentContext (Page 643)
MenuToolBarConfig (Page 804)
ActiveScreen (Page 497)
BaseScreenName (Page 548)
SmartTags (Page 957)

1.5.3.8 Item (RT Professional)

Description

The Item object provides a reference to the current object.

Usage

Use the Item object, for example, to address the properties of the object currently selected in the screen.

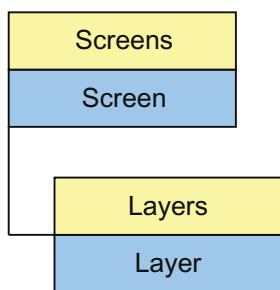
Example

In the following example, you set the background color of the object selected in the screen to red:

```
'VBS195  
Item.BackColor = RGB(255,0,0)
```

1.5.3.9 Layer (RT Professional)

Description



The Layer object returns the result of access to the layers list.

Parent object

Screen in which the screen layer is located

Usage

Depending on certain events, you can use the Layer object to obtain access to the properties of a complete layer to hide or display a layer with operating elements according to the operator authorization.

You can use the "Layer" object as follows:

- "Visible" property
Activate or deactivate visibility of a layer
- "Name" property
Read the name of a layer

Note

The layer property specifies the layer in which the object is located. The layer "0" is output as layer "0".

When accessed, the layers are counted up from 1 in VBS. Therefore, address level "1" with "Layers(2)".

Example

In the following example, Layer 1 is set to "invisible":

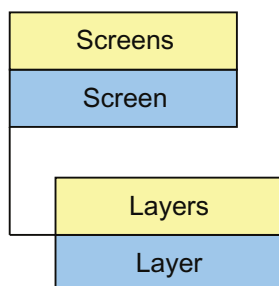
```
'VBS4  
Layers(2).Visible = vbFalse
```

See also

Name (Page 831)

1.5.3.10 Layers (list) (RT Professional)

Description



Use the layers list to access all 32 layers of the graphical Runtime system.

Parent object

Screen in which the screen layer is located

Usage

You use the "Layers" list as follows:

- "_NewEnum" property
Process all layers in the list
- "Count" property
Count all layers contained in the list
- "Item" method
Process a layer from the list

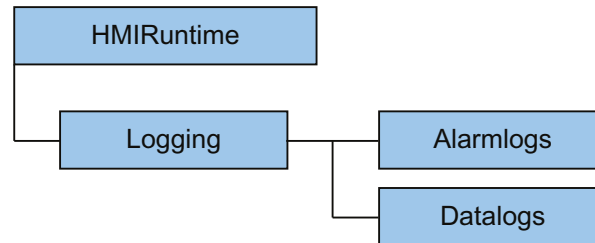
The properties represent default properties and methods of a list and are not described in detail in the WinCC documentation.

See also

Item (Page 1319)

1.5.3.11 Logging (RT Professional)

Description



You can use the object to reconnect swapped-out log segments to Runtime or to remove previously swapped-in log segments. The log segments to be swapped in are copied to the "Common logging" folder of the WinCC project. The previously swapped-in log segments are removed from the "Common logging" folder.

You use parameters to control the location from which log segments are to be swapped in. You specify the time period over which the log segments are to be swapped in or removed.

If an error occurs during the operation with log segments, the applied method returns an error alarm.

Usage

- "Restore" method
Previously swapped out log segments of the alarm log and the data log are connected to Runtime.
- "Remove" method
Previously swapped-in log segments of the alarm log and data log are removed from the Runtime project.

Example

In the following example, log segments from the alarm log and data log are swapped in and the return value is output as a trace.

```
'VBS189
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:
\Folder", "2004-09-14", "2004-09-20", -1) & vbNewLine
```

See also

[Restore \(Page 1343\)](#)

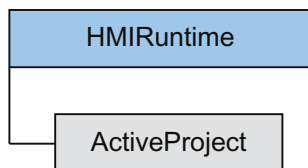
[Remove \(Page 1338\)](#)

[DataLogs \(list\) \(Page 204\)](#)

[AlarmLogs \(list\) \(Page 201\)](#)

1.5.3.12 Project (RT Professional)

Description



Using the object, you can query information from the current Runtime project.

The project object is returned as the result of ActiveProject.

Usage

You can read the following using the "Project" object:

- The path of the current Runtime project ("Path" property)
- The name of the current Runtime project, without path or file extension ("Name" property)

Example

The following example returns name and path of the current Runtime project as a trace:

```
'VBS159
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbNewLine
```

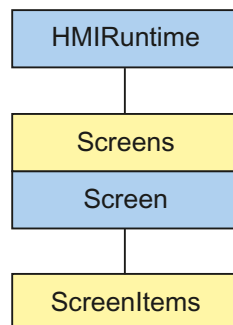
See also

Name (Page 831)

Path (Page 864)

1.5.3.13 Screen (RT Professional)

Description



Represents the screen which is being displayed on the HMI device at the moment or the permanent area in runtime. The screen object is returned as a result of accessing the screen list.

The screen object also contains the following lists:

- You can address all the graphic objects in the addressed screen using the "ScreenItems" list.
- You can address all the layers in the addressed screen using the "Layers" object.

Application

You can use the screen object for the following actions, for example:

- "Width" and "Height" properties
Reading the width and height of a screen
- "BackColor" property
Changing the background color

Use the object name as an index.

Example

In the following example, the background color is changed from black to green:

```
'VBS_Example_BackColor
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

Parent object

Screen window in which the screen object is embedded.

If the screen object is the root screen, the parent object is not defined and set to zero.

Note

If you perform a screen change, the open references to screens that are no longer available will become invalid. As a result, it is no longer possible to work with these references.

Example

In the following example, the width of the first screen is increased by 20 pixels in Runtime:

```
'VBS7
Dim objScreen
Set objScreen = HMIRuntime.Screens(1)
MsgBox "Screen width before changing: " & objScreen.Width
objScreen.Width = objScreen.Width + 20
MsgBox "Screen width after changing: " & objScreen.Width
```


Notes on Cross References

All the screens you address with the standard formulation are automatically compiled by the CrossReference of WinCC and then listed in the screen properties.

```
HMIRuntime.BaseScreenName = "Screenname"
```

If you call screens with different formulations in the code, you make them known by the following section of the CrossReference:

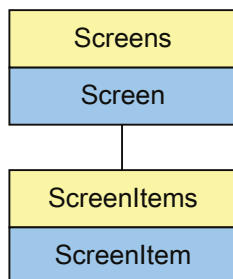
```
' WINCC:SCREENNAME_SECTION_START  
Const ScreenNameInAction = "ScreenName"  
' WINCC:SCREENNAME_SECTION_END  
The section can be inserted in VBS actions as often as required.
```

See also

- [Refresh \(Page 1337\)](#)
- [Activate \(Page 1254\)](#)
- [ObjectSizeDeclutteringEnable \(Page 841\)](#)
- [ObjectSizeDeclutteringMax \(Page 842\)](#)
- [ObjectSizeDeclutteringMin \(Page 842\)](#)
- [LayerDeclutteringEnable \(Page 773\)](#)
- [Layers \(Page 774\)](#)
- [DataSet \(Page 648\)](#)
- [ExtendedZoomingEnable \(Page 686\)](#)
- [AccessPath \(Page 496\)](#)

1.5.3.14 ScreenItem (RT Professional)

Description



Represents an object in the specified screen. The ScreenItem object is an element of the ScreenItems list.

Application

You can use the ScreenItem object to access the properties of graphic objects within a screen, depending on certain events.

You use the "ScreenItem" object as follows, for example:

- "Visible" property
Switch the visibility of an object on or off.
- "Height" and "Width" properties
Query the width and height of an object.
- "Top" and "Left" properties
Change the position of an object.
- "ObjectName" property
Read the name of a graphic object
- "Parent" property
Set a reference to the parent screen

Use the ScreenItems property to return an object to the screen. Use the object name as an index.

Example

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
'VBS_Example_ScreenItems  
  
Dim objCircle  
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")  
objCircle.BackColor = vbGreen
```

Note

To save memory space in the HMI device, no object names are loaded during transfer of the project. If you still want to transfer object names, call up the Runtime settings for the respective HMI device in WinCC. You can change the setting under "General". The object name is required when the object should be accessed via the object name or for debugging a project.

The "ScreenItem" object has different properties depending on its features. The following properties are provided for every "ScreenItem" object:

- Enabled
- Height
- Left
- ObjectName
- Parent
- Top
- Type
- Visible
- Width

If a specific object type is addressed, further properties are added to the standard properties. The additional properties are provided in the descriptions of the individual object types.

See also

Activate (Page 1254)

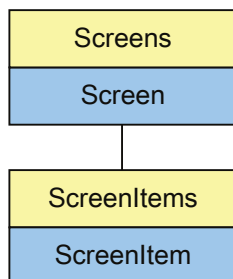
Layer (Page 768)

ScreenItems (Page 904)

Top (Page 1086)

1.5.3.15 ScreenItems (list) (RT Professional)

Description



A list of ScreenItem objects with all screen objects of the specified screen. The list has a "Parent" property. This "Parent" property provides a reference to the screen in which the screen object is located.

Usage

You use the "ScreenItems" list as follows:

- To edit or output all objects within the list (that is, all objects within a screen)
- "Count" property
Count the objects of a screen
- "Item" method
Work on a particular object in the list

Use the screen items property to return an object from the screen. Use the object name as an index.

Example

In the following example, the background color of the "RootScreen" circle in the "myCircle" screen is set to green.

```
Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

Peculiarities of the ScreenItem object

If you embed an external control (ActiveX control or OLE object) in WinCC, the properties of the embedded control may have the same names as the general properties of the ScreenItem object. In this case, the ScreenItem properties take priority.

You can address the properties of the embedded controls, however, via the "object" property. The "object" property is only available in ActiveX controls and OLE objects.

Example:

```
'Control1 is an embedded ActiveX-Control with property "type"
'VBS196
Dim Control
Set Control=ScreenItems("Control1")
Control.object.type

'Control1 is a WinCC-Control
'VBS197
Dim Control
Set Control=ScreenItems("Control1")
Control.type
```

Example

In the following example, you output the name of the objects in the current screen in a message box:

```
Sub OnClick(ByVal Item)
'VBS6
Dim lngAnswer
Dim lngIndex
lngIndex = 1
For lngIndex = 1 To ScreenItems.Count
lngAnswer = MsgBox(ScreenItems(lngIndex).Objectname, vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
End Sub
```

See also

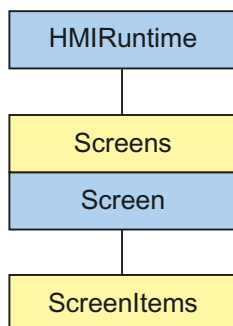
[Item \(Page 1319\)](#)

[ScreenItems \(Page 904\)](#)

[Top \(Page 1086\)](#)

1.5.3.16 Screen object (list) (RT Professional)

Description



Several screens can be opened simultaneously in WinCC Runtime by means of the screen window technique, whereby there is only one main screen. The "Screens" list allows access to all open screens in Runtime using the screen name. The screen list contains all hidden screens.

The access key required in the VBS environment in the `HMIRuntime.Screens(<access key>)` instruction must conform to the following syntax description:

```
[<Root screen name>.]<Screen window name>[:<Screen name>] ...  
.<Screen window name>[:<Screen name>]
```

- The access key represents the screen hierarchy.
- You can omit the screen name at all locations in the key.
- The "AccessPath" property of the "Screen" object corresponds to the full access key.
- The root screen can be addressed via the "" access key.

Examples

The screens are addressed by specifying the hierarchy in the list. You can address the screens with or without using the screen names. In the following example, a "BaseScreenName" root screen is configured with a "ScreenWindow". The screen window contains a "ScreenName" screen.

Addressing using the screen name

```
'VBS8
Set objScreen = HMIRuntime.Screens("BaseScreenName.ScreenWindow:ScreenName")
```

Addressing without using the screen name

```
'VBS9
Set objScreen = HMIRuntime.Screens("ScreenWindow")
```

Referencing the root screen in various ways

```
'VBS10
Set objScreen = HMIRuntime.Screens(1)

'VBS11
Set objScreen = HMIRuntime.Screens("")

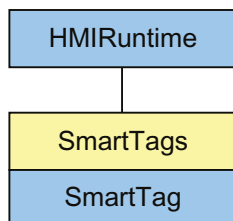
'VBS12
Set objScreen = HMIRuntime.Screens("BaseScreenName")
```

See also

[Screens \(Page 905\)](#)

1.5.3.17 SmartTag (RT Professional)

Description



Represents the value of the specified process tag. The SmartTag object is an element of the SmartTag list.

Application

The SmartTag object provides read and write access to the value of the specified process tag. The SmartTag object does not return an object reference. Use the SmartTags list to return the value of a process tag. Use the tag name as an index.

Note

With the "SmartTags reads values from the cache" setting, values are read from the process image (cache) instead of directly from the controller.

The SmartTag object can also read values directly from the controller. However, you can then expect a substantially higher communication load between the HMI device and the controller.

Example

```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar  
  
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Note

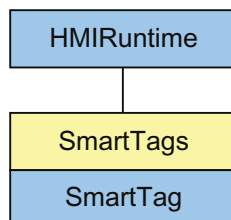
If you want to return the "TypeName" of a SmartTag object data type with the VBS function "TypeName", use the following syntax:

```
TypeName(SmartTags("FillLevel").value)
```

To access the value of an array element, use "SmartTags("<Tag>[Index]")". Set the number of the desired array element for "index", for example, "SmartTags("AirPressure[2]")".

1.5.3.18 SmartTags (RT Professional)

Description



A list of SmartTag objects which represent all of the tags in WinCC Runtime.

Note

The SmartTags list has a limited range of functions. You can only use the tag names to access a SmartTag object. Access via the index or by using "For each" instruction is not supported.

Note

In order to access a tag, which has still not been created in the project, using the SmartTags list, no value is returned. Assignment to a non-existing tag is not executed:

```
Dim intVar  
intVar = SmartTags("FillLevel")
```

"intVar" remains empty when the "FillLevel" tag has not been created.

Note**Current system behavior for dynamic access to elements of a user data type**

When you compose the tag name of a user data type element dynamically in Runtime, the composed tag name is **not** recognized under the following condition:

- Tag of the data type of the user data type is created.
- Tag is **not** used, e.g. at an I/O field

Example:

You have configured a user data type "Motor" with the elements "RPM" (Int), "On" (Bool) and "Off" (Bool). In the project, you have configured the tag "Motor1" as instance of the user data type. You want to output the values of the elements in only one I/O field in runtime. You have configured an additional I/O field for input of the element name. The value you enter is saved to the internal tag strElementName. With the following script you output the value of an element in the I/O field "IOFieldOutputValue":

```
Dim strDynElementName, objIOFieldOutputValue  
Set objIOFieldOutputValue = objscreen.ScreenItems("IOFieldOutput")  
'Get element name from tag value  
strDynElementName = SmartTags("strElementName").Value  
'Create tag name  
objIOFieldOutputValue.ProcessValue =  
SmartTags("Motor1."+strDynElementName).Value
```

Help:

Configure a separate I/O field for each user data type element, for example, in an additional screen. Interconnect the "process value" with the respective user data type element.

Note**Dynamizing faceplates**

Use the following syntax to access a tag that is connected at a faceplate interface ("Properties"):

```
Smarttags("Properties\int_input1_tag")
```

Application

Use the SmartTags list to return a SmartTag object. Use the tag name to reference the SmartTag object.

```
'VBS_Example_SmartTags
'Writes tag value to local tag and returns a user-defined text through the
operating system channel for debug alarms.
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

In Runtime Advanced and Panels you address the tag directly using its name. If the tag name corresponds to the VBS name conventions, you do not need to use the SmartTags list. Follow the example below:

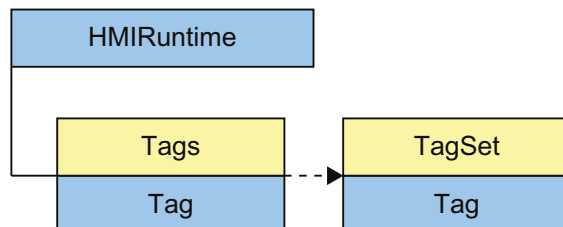
```
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(AirPressure)
HMIRuntime.Trace strAirPressure
```

See also

[SmartTag \(Page 224\)](#)

1.5.3.19 Tag (RT Professional)

Description



A tag object is returned via the "Tags" list. A tag object can be used to address all the properties and methods of a tag.

When creating a tag object, all the properties are initialized with the following values:

- Value = VT_EMPTY
- Name = Tag name
- QualityCode = BAD NON-SPECIFIC
- TimeStamp = 0
- LastError = 0
- ErrorDescription = " "

Note

A summary of possible Quality Codes can be found in WinCC Information System under key word "Communication" > "Diagnostics" or "Communication" > "Quality Codes".

Usage

Use the "Tag" object as follows:

- "Name", "QualityCode", "TimeStamp", "LastError" and "ErrorDescription" properties
Read information on the tag
- "Write" method, "Value" property
Set a value for a tag
- "Read" method, "Value" property
Read a value for a tag

Example

The following example reads the value in the "Tag1" tag:

```
'VBS13
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read()
MsgBox objTag.Value
```

Declaration of tags in WinCC

Always define internal tags in VB script using the "Dim" instruction in order to prevent writing tags wrongly.

When creating a new action, the "Option explicit" instruction is automatically entered in the declaration area and cannot be deleted.

Do not use the "Option explicit" statement in your code as it can lead to Runtime errors.

Example

The following example reads the declaration value in the "lngVar" VB Script tag:

```
'VBS14
Dim lngVar
lngVar = 5
MsgBox lngVar
```

Note

Tag names must not contain any special characters.

When creating a tag, ensure it does not contain a value (Value = VT_EMPTY). Initialize the tags after declaration with the corresponding value.

Notes on Cross References

All the pictures which are addressed with the standard formulation

```
HMIRuntime.Tags("Tagname")
```

are automatically compiled by the CrossReference of WinCC and then listed in the picture properties.

If you address tags with different formulations in the code, you can make them known by the following section of the CrossReference:

```
' ' WINCC:TAGNAME_SECTION_START  
Const TagNameInAction = "TagName"  
' WINCC:TAGNAME_SECTION_END
```

The section can be inserted in VBS actions as often as required.

Note

Composed tag names may not be recognized by the CrossReference.

See also

Name (Page 831)

Value (Page 1172)

ErrorDescription (Page 674)

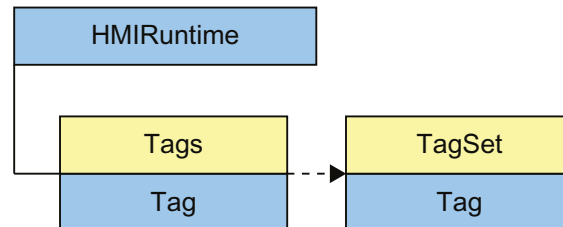
LastError (Page 767)

QualityCode (Page 879)

TimeStamp (Page 1046)

1.5.3.20 Tags (list) (RT Professional)

Description



The "Tags" list enables access to tags in WinCC Runtime. The result of access to the "Tags" list is returned by an object of the type "Tag". The Tag object can be used to access all the tag properties and methods.

Note

"Tags" is a list with a restricted functional scope. The tags in the list cannot be accessed via the index but only by using the tag names. The standard methods `get_Count` and `get_NewEnum` cannot be used in the Tags list.

Application

Tags in the list are accessed via:

```
HMIRuntime.Tags("Tagname")
```

The Tags list is used to declare tags (tag objects) for read and write access. The appropriate HMI tags must exist in order for the write and read access to be executed without error.

You can address HMI tags directly in VBScript and set and read values. If you wish to inquire about additional tag properties, such as quality code or time stamp, or wish to execute error processing, you must address the tag through tags list. The tag object returned enables access to all tag properties and methods.

Using the "CreateTagSet" method, you may generate a TagSet object which allows simultaneous access to several tags.

Example

You use tag names when you set tags.

```
'VBS16
Dim objTag
Set objTag = HMIRuntime.Tags("Tagname")
If you only use the tag name, the "TagPrefix" property is assigned the values from the
current context (the current screen window).
```

1.5.3.21 TagSet (list) (RT Professional)

Description

The "TagSet" object enables simultaneous access to several tags in one call. Simultaneous access demonstrates better performance and lower communication load than single access to multiple tags.

Usage

You can use the TagSet object as follows:

- "Add" method
Add tags to the list
- "Item" method
Access tag objects contained in the list and their properties
- "Write" method
Write all tags of the list
- "Read" method
Read all tags of the list
- "Remove" method
Remove single tags from the list
- "RemoveAll" method
Remove all tags from the list

Tags in the list are accessed via:

```
'VBS169
Dim myTags
myTags = HMIRuntime.Tags.CreateTagSet
myTags("Tagname")
```


In order to have error-free read/write access to tags (tag objects) of the list, the respective tags must exist in WinCC.

If a read/write access error has occurred, the method used will return an error message using the "LastError" and "ErrorDescription" properties.

Example

The following example shows how to generate a TagSet object, how to add tags, and how to write values.

```
'VBS168
'Build a Reference to the TagSet Object
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
'Add Tags to the Collection
group.Add "Motor1"
group.Add "Motor2"
'Set the Values of the Tags
group("Motor1").Value = 3
group("Motor2").Value = 9
'Write the Values to the DataManager
group.Write
```

See also

ErrorDescription (Page 674)

LastError (Page 767)

1.5.4 Object types (Panels, Comfort Panels, RT Advanced, RT Professional)

1.5.4.1 Object types available in VBS (Panels, Comfort Panels, RT Advanced, RT Professional)

Purpose of the tables below

The following tables show the VBS names of the object types and their display name in the "Screens" editor and their availability in the respective runtimes.

Basic objects

VBS name	Display name	Script support	Available in		
			WinCC RT CE (Panels)	WinCC RT Advanced	WinCC RT Professional
Circle	Circle	Yes	Yes	Yes	Yes
CircleSegment	Circle segment	Yes	-	-	Yes

VBS name	Display name	Script support	Available in		
			WinCC RT CE (Panels)	WinCC RT Advanced	WinCC RT Professional
CircularArc	Circular arc	Yes	-	-	Yes
Connector	Connector	Yes	-	-	Yes
Ellipse	Ellipse	Yes	Yes	Yes	Yes
EllipseSegment	Ellipse segment	Yes	-	-	Yes
EllipticalArc	Ellipse arc	Yes	-	-	Yes
GraphicView	Graphic view	Yes	Yes	Yes	Yes
Line	Line	Yes	Yes	Yes	Yes
Polygon	Polygon	Yes	Yes	Yes	Yes
Polyline	Polyline	Yes	Yes	Yes	Yes
Rectangle	Rectangle	Yes	Yes	Yes	Yes
TextField	Text field	Yes	Yes	Yes	Yes
TubeArcObject	Pipe bends	Yes	-	-	Yes
TubeDoubleTeeObject	Double T-piece	Yes	-	-	Yes
TubePolyline	Pipe	Yes	-	-	Yes
TubeTeeObject	T piece	Yes	-	-	Yes

Elements

VBS name	Display name	Script support	Available in		
			WinCC RT CE (Panels)	WinCC RT Advanced	WinCC RT Professional
Bar	Bar	Yes	Yes	Yes	Yes
BatteryView	Charge condition	Yes	Yes	-	-
Button	Button	Yes	Yes	Yes	Yes
CheckBox	Check box	Yes	-	-	Yes
Clock	Clock	Yes	Yes	Yes	Yes
ComboBox	Combo box	Yes	-	-	Yes
DateTimeField	Date/time field	Yes	Yes	Yes	-
DiskSpaceView	Disk space view	Yes	-	-	Yes
Gauge	Gauge	Yes	Yes	Yes	Yes
GraphicIOField	Graphic I/O field	Yes	Yes	Yes	Yes
HelpIndicator	Help indicator	No ¹	Yes	-	-
IOField	I/O field	Yes	Yes	Yes	Yes
MultilineEdit	Editable text field	Yes	-	-	Yes
ListBox	List box	Yes	-	-	Yes
OptionGroup	Option buttons	Yes	-	-	Yes
ProtectedAreaName-View	Effective range name (RFID)	Yes	Yes ²	-	-
RangeLabelView	Effective range name	Yes	Yes ²	-	-
RangeQualityView	Effective range signal	Yes	Yes ²	-	-
RoundButton	Round button	Yes	-	-	Yes

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

VBS name	Display name	Script support	Available in		
			WinCC RT CE (Panels)	WinCC RT Advanced	WinCC RT Professional
Slider	Slider	Yes	Yes	Yes	Yes
Switch	Switch	Yes	Yes	Yes	-
SymbolicIOField	Symbolic I/O field	Yes	Yes	Yes	Yes
SymbolLibrary	Symbol library	Yes	Yes	Yes	Yes
WindowSlider	Scroll bar	Yes	-	-	Yes
WLANQualityView	WLAN reception	Yes	Yes ²	-	-
ZoneLabelView	Zone name	Yes	Yes ²	-	-
ZoneQualityView	Zone signal	Yes	Yes ²	-	-

¹ Only configurable in the global screen

² Mobile Panels

Controls

VBS name	Display name	Script support	Available in		
			WinCC RT CE (Panels)	WinCC RT Advanced	WinCC RT Professional
AlarmControl	Alarm view	Yes	-	-	Yes
AlarmIndicator	Alarm indicator	No ¹	Yes	Yes	-
AlarmView	Alarm view	Yes	Yes	Yes	-
AlarmWindow	Alarm window	No ¹	Yes	Yes	-
ApplicationWindow	Print job/Script diagnostics	Yes	-	-	Yes
CameraControl	Camera view	Yes	Yes	-	-
ChannelDiagnose	Channel diagnostics	Yes	-	-	Yes
FunctionTrendControl	f(x) trend view	Yes	Yes	Yes	Yes
HTMLBrowser	HTML browser	Yes	Yes	Yes	Yes
MediaPlayer	Media Player	Yes	Yes	-	Yes
OnlineTableControl	Table view	Yes	-	-	Yes
OnlineTrendControl	f(t) trend view	Yes	-	-	Yes
PDFview	PDF view	Yes	Yes	Yes	-
PLCCodeViewer	PLC code viewer	Yes	-	-	Yes
RecipeView	Recipe view	Yes	Yes	Yes	-
S7GraphOverview	S7-GRAPH Overview	Yes	-	-	Yes
ScreenWindow	Screen window	Yes	-	-	Yes
SmartClientView	Sm@rtClient view	Yes	Yes	Yes	-
StatusForce	Watch table	Yes	Yes	Yes	-
SysDiagControl	System diagnostics view	Yes	Yes	Yes	Yes
SysDiagWindow	System diagnostics view	No ¹	Yes	Yes	-
TrendRulerControl	Value table	Yes	-	-	Yes

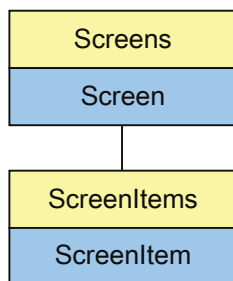
VBS name	Display name	Script support	Available in		
			WinCC RT CE (Panels)	WinCC RT Advanced	WinCC RT Professional
TrendView	Trend view	Yes	Yes	Yes	-
UserArchiveControl	Recipe view	Yes	-	-	Yes
UserView	User view	Yes	Yes	Yes	Yes

¹ Only configurable in the global screen

1.5.4.2 Objects A-I (Panels, Comfort Panels, RT Advanced, RT Professional)

AlarmControl (RT Professional)

Description



Represents the "Alarm view" object. The AlarmControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIAlarmControl

Example

In the following example, the object with the name "Control1" is moved 10 pixels to the right:

```
'VBS54
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 10
```

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-1 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllFilters	-	-	-	-
AllFiltersForHitlist	-	-	-	-
AllServer (Page 513)	RW	-	-	Specifies whether the alarms of all available servers are displayed.
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettings (Page 516)	RW	-	-	Specifies whether the project settings are applied from the "HMI alarms" editor.
ApplyProjectSettings-ForDesignMode	-	-	-	-
AssignedFilters	-	-	-	-
AssignedHitlistFilters	-	-	-	-
AutoCompleteColumns (Page 523)	RW	-	-	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 523)	RW	-	-	Specifies whether empty rows are shown if the control is longer than the number of configured rows.
AutoScroll (Page 525)	RW	-	-	Specifies how the alarm window reacts when new alarms appear.
AutoSelectionColors (Page 525)	RW	-	-	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.
AutoSelectionRectColor (Page 526)	RW	-	-	Specifies whether the selection frame is shown with the color defined by the system.
BackColor (Page 530)	RW	-	-	Specifies the background color of the selected object.
Blocks	-	-	-	-
BorderColor	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness of the selected object.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (Page 591)	RW	-	-	Specifies the text that is displayed in the header of the selected object.
CellCut (Page 595)	RW	-	-	Specifies whether the contents of the cells are abbreviated if the cells are too narrow.
CellSpaceBottom (Page 596)	RW	-	-	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 597)	RW	-	-	Specifies the left indent used in the table cells.
CellSpaceRight (Page 597)	RW	-	-	Specifies the right indent used in the table cells.
CellSpaceTop (Page 598)	RW	-	-	Specifies the top margin of the table cells.
Closeable (Page 603)	RW	-	-	Specifies whether the object can be closed in runtime.
ColumnResize (Page 620)	RW	-	-	Specifies whether you can change the width of the columns.
Columns	-	-	-	-
ColumnScrollbar (Page 621)	RW	-	-	Enables the display of column scroll bars.
ColumnTitleAlignment (Page 631)	RW	-	-	Specifies the type of column title alignment.
ColumnTitles (Page 632)	RW	-	-	Specifies whether the column header is displayed.
ControlDesignMode (Page 638)	RW	-	-	Specifies the style of the control.
DefaultFilterEom	-	-	-	-
DefaultHitListFilterEom	-	-	-	-
DefaultMsgFilterSQL (Page 649)	RW	-	-	Specifies an SQL statement as default for alarm filters.
DefaultSort (Page 650)	RW	-	-	Specifies the type of sorting.
DefaultSort2 (Page 650)	RW	-	-	Specifies the type of sorting.
DefaultSort2Column (Page 651)	RW	-	-	Specifies a column by which the alarm view is sorted initially by default.
DiagnosticsContext (Page 652)	RW	-	-	Specifies the diagnostics context.
DisplayOptions (Page 657)	RW	-	-	Specifies whether alarms whose view was suppressed are displayed.
DoubleClickAction (Page 659)	RW	-	-	Specifies the action to be executed in runtime by double-clicking on a message line.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ExportDelimiter	-	-	-	-
ExportDirectoryChange- able (Page 678)	RW	-	-	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 679)	RW	-	-	Specifies the target directory for the exported run- time data.
ExportFileExtension (Page 679)	RW	-	-	Specifies the file extension of the export file. Only the "csv" file extension is currently supported.
ExportFilename (Page 680)	RW	-	-	Specifies the name of the target file for the exported runtime data.
ExportFilename- Changeable (Page 681)	RW	-	-	Specifies whether the export file name can be changed in runtime.
ExportFormat	-	-	-	-
ExportFormatGuid (Page 682)	RW	-	-	Specifies the assignment of ID number and export provider.
ExportFormatName (Page 683)	RW	-	-	Specifies the export file format. Only the "csv" file format is currently available for the export.
ExportParameters (Page 684)	RW	-	-	Specifies the parameters of the selected format by means of the properties dialog.
ExportSelection (Page 684)	RW	-	-	Specifies which runtime data of the control is expor- ted.
ExportShowDialog (Page 685)	RW	-	-	Specifies whether the data export dialog is shown in runtime.
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Filter	-	-	-	-
Font (Page 709)	RW	-	-	Specifies the font.
GridLineColor (Page 721)	RW	-	-	Specifies the color for the grid lines.
GridLineWidth (Page 723)	RW	-	-	Specifies the width of the separation lines in pixels.
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
Hitlist	-	-	-	-
HitlistColumnAdd (Page 731)	RW	-	-	Transfers the selected alarm text block from the list of available alarm blocks to the list of selected alarm blocks.
HitlistColumnCount (Page 731)	RW	-	-	Specifies the number of alarm text blocks displayed in the hit list in Runtime.
HitlistColumnIndex (Page 732)	RW	-	-	References an alarm text block that is selected for the hit list.
HitlistColumnName (Page 733)	RW	-	-	Shows the name of the alarm text block from the hit list, which is referenced by means of the "HitlistCo- lumnIndex" property.
HitlistColumnRemove (Page 733)	RW	-	-	Cuts the marked alarm text block from the list of se- lected alarm blocks and pastes it into the list of ex- isting alarm blocks.
HitlistColumnRepos (Page 734)	RW	-	-	Changes the sorting order of the alarm blocks.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
HitlistColumnSort (Page 734)	RW	-	-	Specifies the sorting order in the hit list for the alarm text block that is referenced in "HitlistColumnIndex".
HitlistColumnSortIndex (Page 735)	RW	-	-	Defines the sorting order of the alarm text block referenced in "HitlistColumnIndex" in the hit list.
HitlistColumnVisible (Page 736)	RW	-	-	Specifies a list with the selected alarm text blocks of the hit list which are used in the control in Runtime.
HitlistDefaultSort (Page 736)	RW	-	-	Sets the default sorting order in the table columns of the hit list.
HitlistFilter	-	-	-	-
HitlistMaxSourceItems (Page 737)	RW	-	-	Sets the maximum number of data records from the alarm log that are to be used to create the hit list.
HitlistMaxSourceItems- Warn (Page 738)	RW	-	-	Specifies whether a warning should be given if the maximum number of data records that is defined in "HitlistMaxSourceItems" is reached.
HitListRelTime (Page 738)	RW	-	-	Sets a time range for the statistics.
HitListRelTimeFactor (Page 739)	RW	-	-	Specifies the time factor that together with time type "HitlistRelTimeFactorType" determines the time period for which the statistics of the hit list is created.
HitlistRelTimeFactor- Type (Page 739)	RW	-	-	Specifies the time type which, alongside the time factor "HitlistRelativeTimeFactor", determines the time period in which the statistics of the hit list will be created.
HorizontalGridLines (Page 741)	RW	-	-	Specifies whether horizontal separating lines are displayed.
IconSpace (Page 744)	RW	-	-	Specifies the spacing between icons and text in the table cells.
IsActive	-	-	-	-
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate of the object.
LineBackgroundColor	-	-	-	-
LineColor (Page 790)	RW	-	-	Specifies the color of the window separation lines.
LineStyle	-	-	-	-
LineWidth (Page 794)	RW	-	-	Specifies the line thickness of the selected object.
Location	-	-	-	-
LongTermArchiveCon- sistency (Page 799)	RW	-	-	Specifies how alarms are displayed in the alarm view when the "Historical alarm list (long-term)" is configured.
MessageBlockAlign- ment (Page 805)	RW	-	-	Sets the mode of alignment for the table contents of the selected alarm block.
MessageBlockAutoPre- cisions (Page 806)	RW	-	-	Enables automatic setting of the number of decimal places.
MessageBlockCaption (Page 807)	RW	-	-	Specifies the caption of the column header in the alarm view for the selected alarm text block.
MessageBlockCount (Page 807)	RW	-	-	Specifies the number of existing alarm text blocks that are available for the alarm list and hit list.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
MessageBlockDateFormat (Page 808)	RW	-	-	Specifies the date format for displaying the alarms.
MessageBlockExponentialFormat (Page 808)	RW	-	-	Specifies the exponential notation for visualization of the values of a selected alarm block.
MessageBlockFlashOn (Page 809)	RW	-	-	Enables flashing of the selected alarm block in Runtime when an alarm appears.
MessageBlockHideText (Page 810)	RW	-	-	Enables the textual display of the content of the selected alarm block.
MessageBlockHideTitleText (Page 810)	RW	-	-	Specifies textual display of the caption of the selected alarm block.
MessageBlockId (Page 811)	RW	-	-	Specifies the assignment of ID number and alarm text block in the alarm view.
MessageBlockIndex (Page 812)	RW	-	-	References an existing alarm text block.
MessageBlockLeadingZeros (Page 812)	RW	-	-	Enables the display of the selected alarm block with leading zeroes.
MessageBlockLength (Page 813)	RW	-	-	Specifies the length of the selected alarm block based on the number of characters.
MessageBlockName (Page 813)	RW	-	-	Specifies the name for the selected alarm text block.
MessageBlockPrecisions (Page 814)	RW	-	-	Specifies the number of decimal places for the values of the selected alarm block.
MessageBlockSelected (Page 814)	RW	-	-	Existing alarm blocks are blocks that are available for use in the control in Runtime for the alarm list or hit list.
MessageBlockShowDate (Page 815)	RW	-	-	Specifies whether to display both the date and time in the "Time" alarm text block.
MessageBlockShowIcon (Page 816)	RW	-	-	Enables the display of the content of the selected alarm block as icon.
MessageBlockShowTitleIcon (Page 816)	RW	-	-	Specifies textual display of the caption of the selected alarm block.
MessageBlockTextId (Page 817)	RW	-	-	Specifies naming of the selected alarm text block by means of a text ID that was taken from the text library.
MessageBlockTimeFormat (Page 817)	RW	-	-	Specifies the format for the time or period used to display the alarms.
MessageBlockType (Page 818)	RW	-	-	Specifies the number of existing alarm text blocks that are available for the alarm list and hit list.
MessageColumnAdd (Page 819)	RW	-	-	Adds the selected alarm text block from the list of available alarm blocks to the list of selected alarm blocks.
MessageColumnCount (Page 819)	RW	-	-	Specifies the number of selected alarm text blocks displayed in the alarm list in Runtime.
MessageColumnIndex (Page 820)	RW	-	-	References an alarm text block selected for the alarm list.

Properties	RT Professional	RT Advanced	Panel RT	Description
MessageColumnName (Page 821)	RW	-	-	Specifies the name of the alarm text block in the alarm list which is referenced with the "MessageColumnIndex" property.
MessageColumnRemove (Page 821)	RW	-	-	Cuts the marked alarm text block from the list of selected alarm blocks and pastes it into the list of existing alarm blocks.
MessageColumnRepos (Page 822)	RW	-	-	Specifies the order of the alarm text blocks.
MessageColumnSort (Page 822)	RW	-	-	Specifies the sorting order for the alarm text block that is referenced in "MessageColumnIndex".
MessageColumnSortIndex (Page 823)	RW	-	-	Defines the sorting order of the alarm text block referenced in "MessageColumnIndex".
MessageColumnVisible (Page 824)	RW	-	-	Specifies whether alarm text blocks are displayed in the alarm view.
MessageListType (Page 824)	RW	-	-	This setting defines which alarm list contents are displayed when screen opens.
Moveable (Page 829)	RW	-	-	Specifies whether the window can be moved in Runtime.
MsgFilterSQL (Page 830)	RW	-	-	Specifies one or several SQL statements for user-defined selection of the alarms.
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
OperatorAlarms	-	-	-	-
OperatorMessageId (Page 845)	RW	-	-	Specifies the assignment of ID number and trigger event in the alarm view.
OperatorMessageIndex (Page 846)	RW	-	-	References an operator message event.
OperatorMessageName (Page 846)	RW	-	-	Specifies the name which is referenced with the "OperatorMessageIndex" event in message events for operator messages.
OperatorMessageNumber (Page 847)	RW	-	-	Specifies a message number for the operator message of the selected message event if you do <i>not</i> use the operator message of WinCC.
OperatorMessageSelected (Page 847)	RW	-	-	Activates the message events in the list that trigger operator messages.
OperatorMessageSource1 (Page 848)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 1" of the operator message configured here.
OperatorMessageSource10 (Page 854)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 10" of the operator message configured here.
OperatorMessageSource2 (Page 849)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 2" of the operator message configured here.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
OperatorMessage-Source3 (Page 849)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 3" of the operator message configured here.
OperatorMessage-Source4 (Page 850)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 4" of the operator message configured here.
OperatorMessage-Source5 (Page 851)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 5" of the operator message configured here.
OperatorMessage-Source6 (Page 851)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 6" of the operator message configured here.
OperatorMessage-Source7 (Page 852)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 7" of the operator message configured here.
OperatorMessage-Source8 (Page 853)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 8" of the operator message configured here.
OperatorMessage-Source9 (Page 853)	RW	-	-	Specifies a alarm text block of the operated message to be added to "Process value block 9" of the operator message configured here.
OperatorMessageSource1 (Page 855)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource10 (Page 861)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource2 (Page 855)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource3 (Page 856)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource4 (Page 857)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource5 (Page 857)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource6 (Page 858)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource7 (Page 859)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource8 (Page 859)	RW	-	-	Specifies the format of the source content to be transferred.
OperatorMessageSource9 (Page 860)	RW	-	-	Specifies the format of the source content to be transferred.
PageMode (Page 862)	RW	-	-	Enables paging in the historical alarm list (long-term).
PageModeMessage-Number (Page 863)	RW	-	-	Defines the number of alarm shown per page when paging the historical alarm list (long-term).
PrintJob (Page 876)	RW	-	-	Specifies a print job that is created in the "Reports" editor.
RowScrollbar (Page 889)	RW	-	-	Specifies whether row scroll bars are displayed.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
RowTitleAlignment (Page 890)	RW	-	-	Specifies the type of row header alignment.
RowTitles (Page 891)	RW	-	-	Specifies that row headers will be displayed.
RTPersistence (Page 891)	RW	-	-	Specifies how online configurations of WinCC are retained.
RTPersistenceAuthori- zation (Page 892)	RW	-	-	Specifies the authorization required for online con- figuration in runtime.
RTPersistenceType (Page 893)	RW	-	-	Specifies how online configurations of WinCC are retained.
SelectedCellColor (Page 909)	RW	-	-	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 910)	RW	-	-	Specifies the font color of the selected cell.
SelectedRowColor (Page 912)	RW	-	-	Specifies the background color of the selected row.
SelectedRowForeColor (Page 913)	RW	-	-	Specifies the font color of the selected row.
SelectedTitleColor (Page 915)	RW	-	-	Specifies the background color of the selected table header.
SelectedTitleForeColor (Page 916)	RW	-	-	Specifies the font color of the selected table header.
SelectionColoring (Page 918)	RW	-	-	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 920)	RW	-	-	Specifies whether a selection frame is used for the selected cells or rows.
SelectionRectColor (Page 920)	RW	-	-	Specifies the color of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionRectWidth (Page 921)	RW	-	-	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionType (Page 922)	RW	-	-	Specifies the number of lines you can mark.
ServerNames (Page 927)	RW	-	-	Specifies the servers of a distributed system that form the data source for the alarm view.
ShowSortButton (Page 945)	RW	-	-	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 946)	RW	-	-	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 946)	RW	-	-	Specifies whether a sort index is displayed.
ShowTitle (Page 951)	RW	-	-	Specifies the representation of the control's window title.
Size	-	-	-	-
Sizeable (Page 956)	RW	-	-	Specifies that the size of an object can be changed in runtime.
SortSequence (Page 958)	RW	-	-	Specifies how the sorting order can be changed by mouse click.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
StatusbarBackColor (Page 963)	RW	-	-	Specifies the background color for status bars.
StatusbarElementAdd (Page 964)	RW	-	-	Creates a new, user-defined status bar element.
StatusbarElementAuto- Size (Page 965)	RW	-	-	Specifies whether the width of the selected status bar element is set automatically.
StatusbarElement- Count (Page 966)	RW	-	-	Specifies the number of configurable status bar elements.
StatusbarElementIco- nId (Page 967)	RW	-	-	Specifies the ID number and symbol assignment of a status bar element.
StatusbarElementID (Page 967)	RW	-	-	Specifies the ID number of the selected status bar element.
StatusbarElementIndex (Page 968)	RW	-	-	Specifies the reference for a status bar element.
StatusbarElement- Name (Page 969)	RW	-	-	Specifies the object name of the selected status bar element.
StatusbarElementRe- move (Page 970)	RW	-	-	Removes the selected user-defined status bar element.
StatusbarElementRe- name (Page 971)	RW	-	-	Changes the name of the user-defined status bar element that is referenced by the "StatusbarElementIndex" property.
StatusbarElementRe- pos (Page 972)	RW	-	-	Specifies the index of an element in the status bar of the display and operating object.
StatusbarElements	-	-	-	-
StatusbarElementText (Page 973)	RW	-	-	Specifies the text of the selected status bar element.
StatusbarElementTool- tipText (Page 974)	RW	-	-	Specifies the tooltip text for the selected user-defined status bar element.
StatusbarElementUser- Defined (Page 974)	RW	-	-	Specifies whether the configuration engineer has added the status bar element as a new user-defined element.
StatusbarElementVisi- ble (Page 975)	RW	-	-	Specifies whether the referenced element of the status bar element is displayed in Runtime.
StatusbarElementWidth (Page 976)	RW	-	-	Specifies the width of the referenced status bar element in pixels.
StatusbarFont (Page 977)	RW	-	-	Specifies the font of the texts in the status bar.
StatusbarFontColor (Page 978)	RW	-	-	Specifies the color of the texts in the status bar.
StatusbarShowTooltips (Page 979)	RW	-	-	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 980)	RW	-	-	Specifies the default text in the status bar.
StatusbarUseBackCol- or (Page 980)	RW	-	-	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 981)	RW	-	-	Specifies whether the status bar of the control is displayed.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (Page 992)	RW	-	-	Specifies the background color of the table rows in the object.
TableColor2 (Page 992)	RW	-	-	Specifies the second background color of the rows.
TableForeColor (Page 994)	RW	-	-	Specifies the text color used in the table cells of the object.
TableForeColor2 (Page 995)	RW	-	-	Specifies the second text color used in the table cells of the object.
TimeBase (Page 1027)	RW	-	-	Specifies the time zone in which the time values are displayed.
TitleColor (Page 1049)	RW	-	-	Specifies the background color of the table header.
TitleCut (Page 1050)	RW	-	-	Specifies whether the content of the fields in a title bar is to be shortened if the column is too narrow.
TitleDarkShadowColor (Page 1050)	RW	-	-	Specifies the color for the dark side of the 3D shading for column and row headers in the object table.
TitleForeColor (Page 1051)	RW	-	-	Specifies the text color for the table column and row headers of the object.
TitleGridLineColor (Page 1052)	RW	-	-	Specifies the color of the separation lines in the title bar of the table.
TitleLightShadowColor (Page 1053)	RW	-	-	Specifies the color for the light side of the 3D shading for table column and row headers of the object.
TitleSort (Page 1054)	RW	-	-	Specifies how sorting by column header is triggered.
TitleStyle (Page 1055)	RW	-	-	Specifies whether a shading color is used for the table header.
ToolbarAlignment (Page 1063)	RW	-	-	Specifies the position of the toolbar.
ToolbarBackColor (Page 1064)	RW	-	-	Specifies the background color of the toolbar.
ToolbarButtonActive (Page 1065)	RW	-	-	Specifies whether the function linked to the button is activated in runtime.
ToolbarButtonAdd (Page 1066)	RW	-	-	Creates a new user-defined button in the object's toolbar.
ToolbarButtonAuthori- zation (Page 1066)	RW	-	-	Shows the authorization for the selected button function.
ToolbarButtonBe- ginGroup (Page 1067)	RW	-	-	Inserts a leading separator (vertical line) for the selected key function on the toolbar.
ToolbarButtonClick (Page 1068)	RW	-	-	Clicks a toolbar button.
ToolbarButtonCount (Page 1069)	RW	-	-	Specifies the number of buttons in the toolbar.
ToolbarButtonEnabled (Page 1070)	RW	-	-	Specifies whether a user-defined toolbar button can be operated.
ToolbarButtonHotKey (Page 1071)	RW	-	-	Specifies the shortcut for the selected button of the object.
ToolbarButtonID (Page 1071)	RW	-	-	Specifies the key function via your unique ID number.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ToolBarButtonIndex (Page 1072)	RW	-	-	References a button function.
ToolBarButtonLocked (Page 1073)	RW	-	-	Enables/disables display of the locked, pressed state for a user-defined toolbar button.
ToolBarButtonName (Page 1074)	RW	-	-	Specifies the name of the selected user-defined button.
ToolBarButtonRemove (Page 1075)	RW	-	-	Removes the selected user-defined button.
ToolBarButtonRename (Page 1076)	RW	-	-	Specifies the name of the user-defined toolbar button that is being referenced via the "ToolBarButtonIndex" property.
ToolBarButtonRepos (Page 1077)	RW	-	-	Specifies the position of a referenced button in the object's toolbar.
ToolBarButtons	-	-	-	-
ToolBarButtonTooltip- Text (Page 1078)	RW	-	-	Specifies the tooltip text for the user-defined button in the toolbar.
ToolBarButtonUserDe- fined (Page 1079)	RW	-	-	Specifies whether the project engineer has added the toolbar button as new user-defined button.
ToolBarButtonVisible (Page 1080)	RW	-	-	Specifies whether the button is displayed in the toolbar.
ToolBarShowTooltips (Page 1081)	RW	-	-	Specifies whether the tooltips for the key functions are displayed in runtime.
ToolBarUseBackColor (Page 1082)	RW	-	-	Specifies whether the background color for the toolbar is shown.
ToolBarUseHotKeys (Page 1083)	RW	-	-	Specifies whether the hotkeys for the button functions are activated in Runtime.
ToolBarVisible (Page 1084)	RW	-	-	Specifies whether the toolbar of the control is displayed.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate of the selected object.
UseMessageColor (Page 1157)	RW	-	-	Specifies whether the agreed colors of the message classes are displayed.
UseSelectedTitleColor (Page 1159)	RW	-	-	Specifies whether a selection color is used for the headers of selected table cells.
UseTableColor2 (Page 1161)	RW	-	-	Specifies whether a second row color is used for the representation of the table.
VerticalGridLines (Page 1205)	RW	-	-	Specifies whether vertical separation lines are displayed.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-2 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 1260)	Yes	-	-	Executes the "Connect backup" key function of the control.
CopyRows (Page 1261)	Yes	-	-	Executes the "Copy lines" key function of the control.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
DetachDB (Page 1266)	Yes	-	-	Executes the "Disconnect backup" key function of the control.
Export (Page 1267)	Yes	-	-	Executes the "Export log" or "Export data" key function of the control.
GetHitlistColumn (Page 1271)	Yes	-	-	Returns the column object of the alarm view hit list designated by name or index as "ICCAxMessageColumn" type.
GetHitlistColumnCollection (Page 1269)	Yes	-	-	Returns the list of all column objects of the alarm view hit list as "ICCAxCollection" type.
GetMessageBlock (Page 1272)	Yes	-	-	Returns the alarm text block object of the alarm view designated by name or index as "ICCAxMessageBlock" type.
GetMessageBlockCollection (Page 1273)	Yes	-	-	Returns the list of all alarm text block objects of the alarm view as "ICCAxCollection" type.
GetMessageColumn (Page 1274)	Yes	-	-	Returns the column object of the alarm view designated by name or index as "ICCAxMessageColumn" type.
GetMessageColumnCollection (Page 1275)	Yes	-	-	Returns the list of all column objects of the alarm view as "ICCAxCollection" type.
GetOperatorMessage (Page 1276)	Yes	-	-	Returns the operator message object of the alarm view designated by name or index as "ICCAxOperatorMessage" type.
GetOperatorMessageCollection (Page 1277)	Yes	-	-	Returns the list of all operator message objects of the alarm view as "ICCAxCollection" type.
GetRow (Page 1278)	Yes	-	-	Returns the row object defined by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 1280)	Yes	-	-	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetSelectedRow (Page 1286)	Yes	-	-	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 1287)	Yes	-	-	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatusbarElement (Page 1293)	Yes	-	-	Returns the element of the control's status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 1294)	Yes	-	-	Returns the list of all status bar elements of the control as "ICCAxCollection" type.

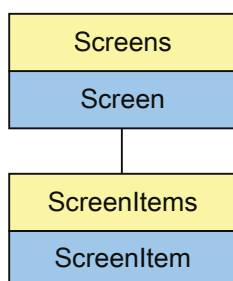
1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
GetToolBarButton (Page 1301)	Yes	-	-	Returns the button function designated by name or index on the control toolbar as "ICCAxToolBarBut- ton" type.
GetToolBarButtonCol- lection (Page 1302)	Yes	-	-	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
HideAlarm (Page 1318)	Yes	-	-	Executes the "Hide alarm" button function of the alarm view.
LockAlarm (Page 1321)	Yes	-	-	Executes the "Disable alarm" button function of the alarm view.
LoopInAlarm (Page 1321)	Yes	-	-	Executes the "Loop in alarm" button function of the alarm view.
MoveToFirstLine (Page 1323)	Yes	-	-	Executes the "First alarm" button function of the alarm view.
MoveToFirstPage (Page 1323)	Yes	-	-	Executes the "First page" button function of the alarm view.
MoveToLastLine (Page 1324)	Yes	-	-	Executes the "Last alarm" button function of the alarm view.
MoveToLastPage (Page 1325)	Yes	-	-	Executes the "Last page" button function of the alarm view.
MoveToNextLine (Page 1326)	Yes	-	-	Executes the "Next alarm" button function of the alarm view.
MoveToNextPage (Page 1326)	Yes	-	-	Executes the "Next page" button function of the alarm view.
MoveToPreviousLine (Page 1327)	Yes	-	-	Executes the "Previous alarm" button function of the alarm view.
MoveToPreviousPage (Page 1328)	Yes	-	-	Executes the "Previous page" button function of the alarm view.
Print (Page 1331)	Yes	-	-	Executes the "Print" button function of the control.
QuitHorn (Page 1332)	Yes	-	-	Executes the "Alarm annunciator acknowledgment" button function of the alarm view.
QuitSelected (Page 1332)	Yes	-	-	Executes the "Single acknowledgment" button func- tion of the alarm view.
QuitVisible (Page 1333)	Yes	-	-	Executes the "Group acknowledgment" button func- tion of the alarm view.
SelectAll (Page 1346)	Yes	-	-	Selects all rows in a table-based control.
SelectRow (Page 1347)	Yes	-	-	Selects a specific row in a table-based control.
ShowComment (Page 1350)	Yes	-	-	Executes the "Comment dialog" button function of the alarm view.
ShowDisplayOptions- Dialog (Page 1351)	Yes	-	-	Executes the "Display options dialog" button func- tion of the alarm view.
ShowEmergencyQuit- Dialog (Page 1351)	Yes	-	-	Executes the "Emergency acknowledgment" button function of the alarm view.
ShowHelp (Page 1352)	Yes	-	-	Executes the "Help" button function of the control.
ShowHideList (Page 1352)	Yes	-	-	Executes the "List of alarm to hide" button function of the alarm view.
ShowHitList (Page 1353)	Yes	-	-	Executes the "Hit list" button function of the alarm view.

Methods	RT Professional	RT Advanced	Panel RT	Description
ShowInfoText (Page 1353)	Yes	-	-	Executes the "About dialog" button function of the alarm view.
ShowLockDialog (Page 1354)	Yes	-	-	Executes the "Lock dialog" button function of the alarm view.
ShowLockList (Page 1354)	Yes	-	-	Executes the "Lock list" button function of the alarm view.
ShowLongTermArchiveList (Page 1355)	Yes	-	-	Executes the "Historical alarm list (long-term)" button function of the alarm view.
ShowMessageList (Page 1355)	Yes	-	-	Executes the "Alarm list" button function of the alarm view.
ShowPropertyDialog (Page 1356)	Yes	-	-	Executes the "Configuration dialog" button function of the control.
ShowSelectionDialog (Page 1358)	Yes	-	-	Executes the "Selection dialog" button function of the alarm view.
ShowShortTermArchiveList (Page 1359)	Yes	-	-	Executes the "Historical alarm list (short-term)" button function of the alarm view.
ShowSortDialog (Page 1360)	Yes	-	-	Executes the "Sorting dialog" button function of the alarm view.
ShowTimebaseDialog (Page 1361)	Yes	-	-	Executes the "Timebase dialog" button function of the alarm view.
UnhideAlarm (Page 1364)	Yes	-	-	Executes the "Unhide alarm" button function of the alarm view.
UnlockAlarm (Page 1364)	Yes	-	-	Executes the "Unlock alarm" button function of the alarm view.
UnselectAll (Page 1365)	Yes	-	-	Removes all selections from the cells of a table-based control.
UnselectRow (Page 1365)	Yes	-	-	Removes the selections from a specific cell of a table-based control.

AlarmView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "Alarm view" object. The AlarmView object is an element of the ScreenItems list.

If you change the settings for this object with a user-defined function, the changed settings are retained even after the screen is called again.

Note

The object "Simple AlarmView" cannot be dynamized with a user-defined function.

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-3 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AlarmAreaHeight	-	-	-	-
AlarmAreaWidth	-	-	-	-
AlarmClasses	-	-	-	-
AlarmLog	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AlarmSource	-	-	-	-
AlarmTextTag	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
BackColor (Page 530)	-	RW	RW	Specifies the background color of the selected object.
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBarElements	-	-	-	-
ButtonBarStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
Columns	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextAckGroup	-	-	-	-
ColumnTextAlarmState	-	-	-	-
ColumnTextAlarmText	-	-	-	-
ColumnTextClassName	-	-	-	-
ColumnTextDate	-	-	-	-
ColumnTextDevice	-	-	-	-
ColumnTextDiagnosable	-	-	-	-
ColumnTextNumber	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ColumnTextTime	-	-	-	-
ColumnWidth	-	-	-	-
ConfiguredAlarmClasses	-	-	-	-
CornerRadius	-	-	-	-
CountOfLinesPerAlarms	-	-	-	-
CountOfVisibleAlarms	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
ES2RT_ButtonPositions	-	-	-	-
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
ES2RT_MessageAreaHeight	-	-	-	-
ES2RT_MessageAreaWidth	-	-	-	-
FilterTag	-	-	-	-
FilterText	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (Page 707)	-	RW	RW	Specifies the color of the focus border when the object has the focus.
FocusWidth (Page 708)	-	RW	RW	Specifies the border width when the object has the focus.
ForeColor	-	-	-	-
GridLineColor (Page 721)	-	RW	RW	Specifies the color for the grid lines.
Height	-	-	-	-
HorizontalScrollingEnabled	-	-	-	-
IsRunningUnderCE	-	-	-	-
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
LineAlarmView	-	-	-	-
Location	-	-	-	-
MessageAreaHeight	-	-	-	-
MessageAreaLeft	-	-	-	-
MessageAreaTop	-	-	-	-
MessageAreaWidth	-	-	-	-
Name	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
PreferredUseOnAck	-	-	-	-
S7Device	-	-	-	-
SecurityForSimpleViewEnabled	-	-	-	-
SelectionBackColor (Page 917)	-	RW	RW	Specifies the background color of the selected cells.
SelectionForeColor (Page 919)	-	RW	RW	Specifies the foreground color of the selected cells.
SeparateLineForAlarmText	-	-	-	-
ShowAcknowledgeButton	-	-	-	-
ShowAlarmsFromDate (Page 929)	-	RW	RW	Specifies that only those message events are displayed that are saved in this tag.
ShowAlarmsToAcknowledge	-	-	-	-
ShowColumnHeaders	-	-	-	-
ShowHelpButton	-	-	-	-
ShowHorizontalGridlines	-	-	-	-
ShowLoopInAlarmButton	-	-	-	-
ShowMilliseconds	-	-	-	-
ShowPendingAlarms	-	-	-	-
Size	-	-	-	-
SortByTimeDirection (Page 957)	-	RW	RW	Specifies whether the last alarm received is shown at the top (ascending sorting order).
SortByTimeEnabled (Page 958)	-	RW	RW	Specifies whether alarm sorting can be altered according to the time.
StyleItem	-	-	-	-
SupportsS7DiagnosticInSimpleView	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (Page 991)	-	RW	RW	Specifies the background color of the table cells.
TableEvenRowBackColor	-	-	-	-
TableFont	-	-	-	-
TableForeColor (Page 994)	-	RW	RW	Specifies the text color used in the table cells of the object.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TableHeaderBackColor (Page 996)	-	RW	RW	Specifies the background color in the header of the table.
TableHeaderBackFill- Style	-	-	-	-
TableHeaderBorder- BackColor	-	-	-	-
TableHeaderBorderCol- or	-	-	-	-
TableHeaderBorder- Width	-	-	-	-
TableHeaderCornerRa- dius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGra- dientColor	-	-	-	-
TableHeaderFirstGra- dientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderForeColor (Page 998)	-	RW	RW	Specifies the text color in the header of the table.
TableHeaderMiddleGra- dientColor	-	-	-	-
TableHeaderPadding- Bottom	-	-	-	-
TableHeaderPaddin- gLeft	-	-	-	-
TableHeaderPadding- Right	-	-	-	-
TableHeaderPadding- Top	-	-	-	-
TableHeaderSecond- GradientColor	-	-	-	-
TableHeaderSecond- GradientOffset	-	-	-	-
ToolbarHeight	-	-	-	-
ToolbarLeft	-	-	-	-
ToolbarTop	-	-	-	-
ToolbarWidth	-	-	-	-
Top (Page 1086)	-	RW	RW	Specifies the value of the Y coordinate.
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGra- dient	-	-	-	-
UseDesignColorSche- ma	-	-	-	-
UseTableHeaderFirst- Gradient	-	-	-	-

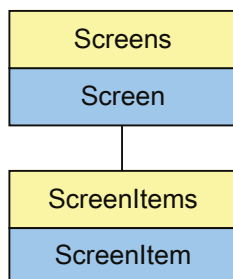
Properties	RT Professional	RT Advanced	Panel RT	Description
UseTableHeaderSecondGradient	-	-	-	-
VerticalScrollBarEnabled	-	-	-	-
VerticalScrollingEnabled	-	-	-	-
ViewType	-	-	-	-
ViewTypeForSaveStream	-	-	-	-
Visible (Page 1208)	-	RW	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-4 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	Yes	Yes	Enables the permanent area or the root screen.

ApplicationWindow (RT Professional)

Description



Represents the "ApplicationWindow" object. The ApplicationWindow object is an element of the ScreenItems list.

Type identifier in VBS

HMIApplicationWindow

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-5 Properties

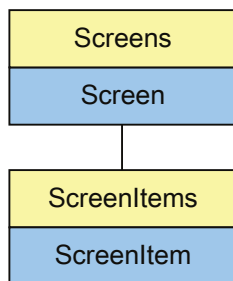
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
BorderEnabled (Page 569)	RW	-	-	Specifies whether the window is displayed with a border in runtime.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
GSCRuntimeAllowed	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
ShowCaption (Page 931)	RW	-	-	Specifies whether the title bar is shown.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Template (Page 1003)	RW	-	-	Specifies the template for displaying window content in the "Print job/Script diagnostics" object.
Top (Page 1086)	-	-	-	-
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.
WindowCloseEnabled (Page 1224)	RW	-	-	Specifies whether a window can be closed in Runtime.
WindowMaximizeEnabled (Page 1224)	RW	-	-	Specifies whether the object can be maximized in Runtime.
WindowMovingEnabled (Page 1225)	RW	-	-	Specifies whether the object can be moved in Runtime.
WindowOnTop (Page 1226)	RW	-	-	Specifies whether the object always remains in the foreground in Runtime.
WindowsContents (Page 1226)	RW	-	-	Specifies the content of the print job or script diagnostics.
WindowSizingEnabled (Page 1227)	RW	-	-	Specifies whether the size can be changed.

Table 1-6 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Bar (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Bar" object. The Bar object is an element of the ScreenItems list.

Type identifier in VBS

HMIBar

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-7 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AlarmLowerLimit (Page 507)	RW	-	-	Specifies the low limit at which the alarm is triggered
AlarmLowerLimitColor (Page 508)	RW	-	-	Specifies the bar color for the "AlarmLowerLimit" limit.
AlarmLowerLimitEna- bled (Page 509)	RW	-	-	Specifies whether the "AlarmLowerLimit" limit is monitored.
AlarmLowerLimitRela- tive (Page 509)	RW	-	-	Specifies whether the low limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.
AlarmUpperLimit (Page 510)	RW	-	-	Specifies the high limit that triggers the interrupt.
AlarmUpperLimitColor (Page 511)	RW	-	-	Specifies the bar color for the "AlarmUpperLimit" limit.
AlarmUpperLimitEna- bled (Page 512)	RW	-	-	Specifies whether the "AlarmUpperLimit" limit is monitored.
AlarmUpperLimitRela- tive (Page 512)	RW	-	-	Specifies whether the high limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
AverageLast15Values (Page 528)	RW	-	-	Specifies whether the average of the last 15 values is to be shown.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background flashes in run-time.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background.
BarBackColor (Page 543)	RW	RW	RW	Specifies the color of the bar background for the selected object.
BarBackFillStyle (Page 544)	RW	-	-	Specifies the fill style for the bar.
BarBackFlashingColor- Off	-	-	-	-
BarBackFlashingColor- On	-	-	-	-
BarBackFlashingEna- bled	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BarBackFlashingRate	-	-	-	-
BarEdgeStyle	-	-	-	-
BarOrientation (Page 548)	RW	-	-	Specifies the bar alignment.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness of the selected object.
BorderWidth3D	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
ColorChangeHysteresis (Page 605)	RW	-	-	Specifies the hysteresis as a percentage of the display value.
ColorChangeHysteresis- sEnabled (Page 605)	RW	-	-	Specifies whether the object is displayed with hysteresis.
CompatibilityMode	-	-	-	-
CornerRadius	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the specified object.
CountDivisions (Page 641)	RW	-	-	Specifies the number of segments into which the bar will be split by means of the large scale tick marks.
CountSubDivisions (Page 641)	-	RW	RW	Specifies the number of scale marks between two main marks.
DeviceStyle	-	-	-	-
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style of the selected object.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	-	-
FlashingColorOff	-	-	-	-
FlashingColorOn	-	-	-	-
FlashingEnabled	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FlashingOnLimitViolation	-	-	-	-
FlashingRate	-	-	-	-
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text of the selected object is shown in bold.
FontName (Page 713)	RW	-	-	Specifies the font of the selected object.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text for the selected object.
ForeColor (Page 716)	RW	RW	RW	Specifies the font color of the text for the selected object.
ForeColorTransparency	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
InnerHeight	-	-	-	-
InnerWidth	-	-	-	-
IntegerDigits (Page 752)	RW	-	-	Specifies the number of digits to the left of the decimal point (0 to 20).
LargeTickLabelingStep (Page 764)	R	-	-	Returns which scale sections are labeled.
LargeTicksBold (Page 764)	RW	-	-	Specifies whether the long tick marks of a scale are shown in bold.
LargeTicksSize (Page 765)	RW	-	-	Specifies the length of the long tick marks of a scale.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the selected object.
Limit4LowerLimit (Page 779)	RW	-	-	Specifies the low limit for "Reserve4".
Limit4LowerLimitColor (Page 780)	RW	-	-	Specifies the color for the "Reserve4" low limit.
Limit4LowerLimitEnabled (Page 781)	RW	-	-	Specifies whether the "Reserve4" low limit is monitored.
Limit4LowerLimitRelative (Page 781)	RW	-	-	Specifies whether the "Reserve4" low limit is evaluated as a percentage or an absolute value.
Limit4UpperLimit (Page 782)	RW	-	-	Specifies the high limit for "Reserve4".
Limit4UpperLimitColor (Page 783)	RW	-	-	Specifies the color for the "Reserve4" high limit.
Limit4UpperLimitEnabled (Page 783)	RW	-	-	Specifies whether the "Reserve4" high limit is monitored.
Limit4UpperLimitRelative (Page 784)	RW	-	-	Specifies whether the "Reserve4" high limit is evaluated as a percentage or an absolute value.
Limit5LowerLimit (Page 785)	RW	-	-	Specifies the low limit for "Reserve5".

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Limit5LowerLimitColor (Page 785)	RW	-	-	Sets the color for the "Reserve5" low limit.
Limit5LowerLimitEna- bled (Page 786)	RW	-	-	Specifies whether the "Reserve5" low limit is moni- tored.
Limit5LowerLimitRela- tive (Page 787)	RW	-	-	Specifies whether the "Reserve5" low limit is evalu- ated as a percentage or an absolute value.
Limit5UpperLimit (Page 787)	RW	-	-	Specifies the high limit for "Reserve5".
Limit5UpperLimitColor (Page 788)	RW	-	-	Specifies the color for the "Reserve5" high limit.
Limit5UpperLimitEna- bled (Page 789)	RW	-	-	Specifies whether the "Reserve5" high limit is moni- tored.
Limit5UpperLimitRela- tive (Page 789)	RW	-	-	Specifies whether the "Reserve5" high limit is eval- uated as a percentage or an absolute value.
LimitRangeCollection	-	-	-	-
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
MaximumValue (Page 803)	RW	RW	RW	Specifies the maximum value of the scale in the se- lected object.
MinimumValue (Page 826)	RW	RW	RW	Specifies the minimum value of the scale in the se- lected object.
Name	-	-	-	-
Precision (Page 875)	RW	-	-	Specifies the number of decimal places (0 to 20).
ProcessValue (Page 877)	RW	RW	RW	Specifies the default for the value to be displayed.
ScaleColor (Page 896)	RW	RW	RW	Specifies the color of the scale of the selected ob- ject.
ScaleGradation (Page 897)	RW	RW	RW	Specifies the distance between two large tick marks of the scale.
ScaleLabelFieldLength	-	-	-	-
ScaleLabelingDouble- Lined	-	-	-	-
ScalePosition (Page 899)	RW	-	-	Specifies the position of the scale of the selected object.
ScaleStart	-	-	-	-
ScalingType (Page 903)	RW	-	-	Specifies the type of bar scaling.
SegmentColoring (Page 907)	RW	RW	RW	Specifies the type of color change to be displayed if the limits are exceeded.
ShowBadTagState (Page 930)	RW	-	-	Defines whether the object is grayed out when a bad quality code or tag status is detected.
ShowLargeTicksOnly (Page 939)	RW	-	-	Specifies whether only large scale marks are shown.
ShowLimitLines	-	-	-	-
ShowLimitMarkers (Page 940)	RW	-	-	Specifies whether the limit values are shown as a scale value.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ShowLimitRanges	-	-	-	-
ShowProcessValue	-	-	-	-
ShowScale (Page 944)	RW	-	-	Specifies whether the values are also shown in a scale.
ShowSignForPositive- Label	-	-	-	-
ShowTickLabels	-	-	-	-
ShowTrendIndicator (Page 954)	RW	-	-	Specifies whether the trend (rising or falling) of the measured value to be monitored is indicated by means of a small arrow.
Size	-	-	-	-
StartValue (Page 962)	RW	-	-	Defines the absolute value of the zero point of the scale indicator.
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToleranceLowerLimit (Page 1057)	RW	-	-	Sets the low limit for tolerance 1.
ToleranceLowerLimit- Color (Page 1058)	RW	-	-	Specifies the color for the low limit "ToleranceLo- werLimit".
ToleranceLowerLimi- tEnabled (Page 1059)	RW	-	-	Specifies whether the "ToleranceLowerLimit" limit is monitored.
ToleranceLowerLimi- tRelative (Page 1059)	RW	-	-	Specifies whether the low limit "ToleranceLowerLi- mit" is evaluated as a percentage or as an absolute value.
ToleranceUpperLimit (Page 1060)	RW	-	-	Sets the high limit for tolerance 1.
ToleranceUpperLimit- Color (Page 1060)	RW	-	-	Specifies the color for the high limit "ToleranceUp- perLimit".
ToleranceUpperLimi- tEnabled (Page 1061)	RW	-	-	Specifies whether the "ToleranceUpperLimit" limit is monitored.
ToleranceUpperLimi- tRelative (Page 1062)	RW	-	-	Specifies whether the high limit "ToleranceUpperLi- mit" is to be evaluated as a percentage or as an absolute value.
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the selec- ted object.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
TrendIndicatorColor (Page 1104)	RW	-	-	Specifies the color for the trend view.
Unit (Page 1146)	RW	-	-	Specifies the unit of measurement.
UseAutoScaling	-	-	-	-
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.

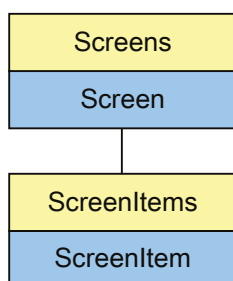
Properties	RT Professional	RT Advanced	Panel RT	Description
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
UseExponentialFormat (Page 1156)	RW	-	-	Specifies whether the numbers are displayed with exponentials (e.g. "1.00e+000").
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
WarningLowerLimit (Page 1213)	RW	-	-	Specifies the low limit for "WarningLowerLimit".
WarningLowerLimitColor (Page 1214)	RW	-	-	Specifies the color for the low limit "WarningLowerLimit".
WarningLowerLimitEnabled (Page 1215)	RW	-	-	Specifies whether the "WarningLowerLimit" limit is monitored.
WarningLowerLimitRelative (Page 1216)	RW	-	-	Specifies whether the low limit "WarningLowerLimit" is evaluated as a percentage or as an absolute value.
WarningUpperLimit (Page 1218)	RW	-	-	Specifies the upper warning limit.
WarningUpperLimitColor (Page 1219)	RW	R	R	Specifies the color for the high limit "WarningUpperLimit".
WarningUpperLimitEnabled (Page 1220)	RW	R	R	Specifies whether the high limit is to be monitored.
WarningUpperLimitRelative (Page 1220)	RW	RW	RW	Specifies whether the high limit "WarningUpperLimit" is to be evaluated as a percentage or as an absolute value.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.
ZeroPoint (Page 1252)	RW	RW	RW	Specifies the position of the zero point as a percentage of the bar height.

Table 1-8 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

BatteryView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "Charge condition" object. The BatteryView object is an element of the ScreenItems list.

Type identifier in VBS

HMIBatteryView

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-9 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	-	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	-	RW	Specifies the value of the X coordinate.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	-	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

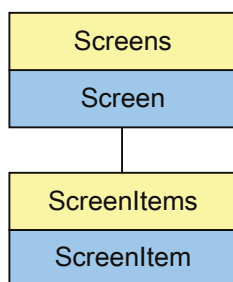
Table 1-10 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

See also

Height (Page 724)

Width (Page 1221)

Button (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "Button" object. The Button object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode of the "button":

Property	"Text" mode	"Text list" mode	"Graphic" mode
TextOff	x	--	--
TextOn	x	--	--

Type identifier in VBS

HMIButton

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-11 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AdaptBorder (Page 502)	RW	-	-	Specifies whether the border of the object will be dynamically adapted to the text size.
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	RW	RW	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	RW	RW	RW	Specifies the background color of the selected object.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern of the selected object.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background of the selected object flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background for the selected object.
BitNumber	-	-	-	-
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line for the selected object.
BorderBrightColor3D (Page 565)	RW	-	-	Specifies the frame color of the following frame parts in case of a 3D display of the selected object: Outer border parts top and bottom; inner border parts bottom and right
BorderColor (Page 566)	RW	-	-	Specifies the line color of the object.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line for the selected object.
BorderShadeColor3D (Page 579)	RW	-	-	Specifies the frame color of the following frame parts in case of a 3D display of the selected object: Inner border parts top and bottom; outer border parts bottom and right
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines for the selected object.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness of the selected object.
BorderWidth3D (Page 584)	RW	-	-	Specifies the width of the inner border for the 3D presentation of the selected object.
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the specified object.
DeviceStyle	-	-	-	-
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style of the selected object.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
ES2RT_StoreAsCheck- Back	-	-	-	-
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern for the selected object.
FirstGradientColor	-	-	-	-
FirstGradientOffset	-	-	-	-
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line for the selected object.
FocusColor (Page 707)	-	RW	RW	Specifies the color for the focus frame of the selected object when it is in focus.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FocusWidth (Page 708)	-	RW	RW	Specifies the border width of the specified object when the object is in focus.
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text of the selected object is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font of the selected object.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text for the selected object.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text of the selected object is underlined.
ForeColor (Page 716)	RW	RW	RW	Specifies the font color of the text for the selected object.
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
HelpText (Page 728)	-	RW	RW	Returns the tooltip that is shown in runtime as user help for the specified object.
HorizontalAlignment (Page 740)	RW	RW	RW	Specifies the horizontal alignment of the text within the selected object.
HorizontalPictureAlign- ment	-	-	-	-
Hotkey	-	-	-	-
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the selected object.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
MiddleGradientColor	-	-	-	-
Mode (Page 828)	RW	-	-	Specifies the field type of the selected object.
Name	-	-	-	-
PictureAlignment (Page 868)	RW	-	-	Specifies the display type for the background screen in the process image.
PictureAreaBottomMar- gin	-	-	-	-
PictureAreaLeftMargin	-	-	-	-
PictureAreaRightMargin	-	-	-	-
PictureAreaTopMargin	-	-	-	-
PictureAutoSizing	-	-	-	-
PictureList	-	-	-	-
PictureOff (Page 870)	RW	-	-	Specifies the picture that is displayed in the "Off" status.
PictureOn (Page 871)	RW	-	-	Specifies the screen to be displayed in the "On" state.

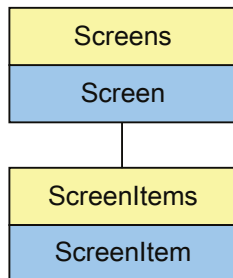
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Pressed (Page 876)	RW	-	-	Specifies whether the selected object is displayed in a pressed state.
ProcessValue (Page 877)	RW	-	-	Specifies the default for the value to be displayed.
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
SecondGradientColor	-	-	-	-
SecondGradientOffset	-	-	-	-
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
StyleItem	-	-	-	-
StyleSettings (Page 986)	RW	-	-	Specifies the style in which the object is displayed.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextAreaBottomMargin	-	-	-	-
TextAreaLeftMargin	-	-	-	-
TextAreaRightMargin	-	-	-	-
TextAreaTopMargin	-	-	-	-
TextList	-	-	-	-
TextOff (Page 1006)	RW	RW	RW	Specifies the text to be displayed in the "Off" state of the selected object.
TextOn (Page 1006)	-	RW	RW	Specifies the text to be displayed in the "On" state of the selected object.
TextOrientation (Page 1007)	RW	-	-	Specifies the text orientation of the selected object.
Toggle (Page 1055)	RW	-	-	Specifies whether the selected object engages after it has been operated in Runtime.
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the selected object.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
TransparentColorPictureOff (Page 1093)	RW	-	-	Specifies the color to be set to "transparent" for the "Off" state of the assigned bitmap object.
TransparentColorPictureOn (Page 1094)	RW	-	-	Specifies the color to be set to "transparent" for the "On" state of the assigned bitmap object.
UseDesignColorScheme (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with the shading defined in the active design.
UseFirstGradient	-	-	-	-
UseSecondGradient	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
UseTransparentColor- PictureOff (Page 1164)	RW	-	-	Specifies if the transparent color defined in the "TransparentColorPictureOff" property is used for the "Off" state.
UseTransparentColor- PictureOn (Page 1165)	RW	-	-	Specifies if the transparent color defined in the "TransparentColorPictureOff" property is used for the "Off" state.
UseTwoHandOperation	-	-	-	-
VerticalAlignment (Page 1204)	RW	RW	RW	Specifies the vertical alignment of the text for the selected object.
VerticalPictureAlign- ment	-	-	-	-
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.
WindowsStyle (Page 1228)	RW	-	-	Specifies whether the object is displayed in the general Windows style.

Table 1-12 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

CameraControl (Panels, Comfort Panels, RT Advanced)**Description**

Represents the "CameraControl" object. The CameraControl object is an element of the ScreenItems list.

Type identifier in VBS

HMICameraControl

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

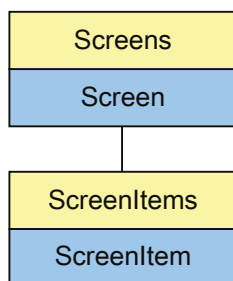
Table 1-13 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CameraUrl	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	-	-	-

Properties	RT Professional	RT Advanced	Panel RT	Description
Layer (Page 768)	-	-	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	-	R	Returns the value of the X coordinate.
Location	-	-	-	-
MaintainAspectRatio	-	-	-	-
MaintainOriginalSize	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	R	Returns the value of the Y coordinate.
UseUdp	-	-	-	-
Visible (Page 1208)	-	-	R	Returns whether the specified object is visible.
Width	-	-	-	-

Table 1-14 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

ChannelDiagnose (RT Professional)**Description**

Represents the "Channel diagnostics display" object. The ChannelDiagnose object is an element of the ScreenItems list.

Type identifier in VBS

HMIChannelDiagnose

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-15 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.

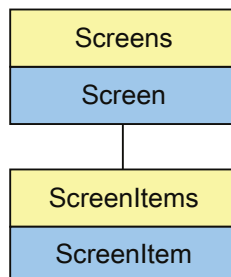
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-16 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

CheckBox (RT Professional)

Description



Represents the "Check box" object. The CheckBox object is an element of the ScreenItems list.

Type identifier in VBS

HMICheckBox

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-17 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AdaptBorder (Page 502)	RW	-	-	Specifies whether the border of the object will be dynamically adapted to the text size.
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
CanBeGrouped	-	-	-	-
CheckMarkAlignment (Page 601)	RW	-	-	Specifies whether the fields are right-aligned.
CheckMarkCount (Page 601)	RW	-	-	Specifies the number of fields.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text is shown underlined.
ForeColor (Page 716)	RW	-	-	Specifies the font color of the text.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalAlignment (Page 740)	RW	-	-	Specifies the horizontal alignment of the text.
Index (Page 745)	RW	-	-	Specifies the index for the selected text field.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
LogOperation (Page 798)	RW	-	-	Specifies whether, after operating this object, an alarm is output to the alarm system.
Name	-	-	-	-

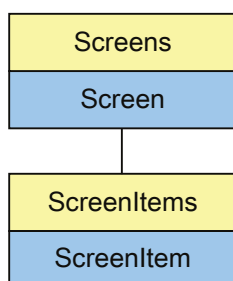
Properties	RT Professional	RT Advanced	Panel RT	Description
ProcessValue (Page 877)	RW	-	-	Specifies the default for the value to be displayed.
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
ShowBadTagState (Page 930)	RW	-	-	Defines whether the object is grayed out when a bad quality code or tag status is detected.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (Page 1004)	RW	-	-	Specifies the label for the text field.
TextHandles	-	-	-	-
TextOrientation (Page 1007)	RW	-	-	Specifies the text orientation.
Texts	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
VerticalAlignment (Page 1204)	RW	-	-	Specifies the vertical alignment of the text in the specified object.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-18 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Circle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Circle" object. The Circle object is an element of the ScreenItems list.

Type identifier in VBS

HMICircle

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-19 Properties

Properties	RT Profes-sional	RT Ad-vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies the operating rights in Runtime.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	RW	RW	Specifies the fill pattern.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	RW	RW	Specifies the line color.

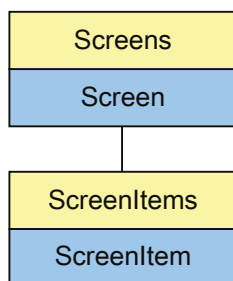
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	RW	RW	Specifies the line thickness.
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DeviceStyle	-	-	-	-
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	RW	RW	Specifies the line style.
Enabled (Page 664)	RW	RW	RW	Specifies whether the object can be operated in runtime.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	RW	RW	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Name	-	-	-	-
Radius (Page 880)	RW	RW	RW	Specifies the radius.
RadiusHeight	-	-	-	-
RadiusWidth	-	-	-	-
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the object is filled.
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	RW	RW	Specifies whether the object is visible.
Width (Page 1221)	RW	RW	RW	Specifies the width of the object in pixels.

Table 1-20 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

CircleSegment (RT Professional)**Description**

Represents the "CircleSegment" object. The CircleSegment object is an element of the ScreenItems list.

Type identifier in VBS

HMICircleSegment

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-21 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 670)	RW	-	-	Specifies the angle by which the end point deviates from the zero position (0°).
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Name	-	-	-	-
Radius (Page 880)	RW	-	-	Specifies the radius.

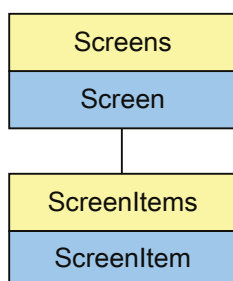
Properties	RT Professional	RT Advanced	Panel RT	Description
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
StartAngle (Page 960)	RW	-	-	Specifies the angle by which the start point deviates from the zero position (0°).
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorScheme (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-22 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

CircularArc (RT Professional)

Description



Represents the "CircularArc" object. The CircularArc object is an element of the ScreenItems list.

Type identifier in VBS

HMICircularArc

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-23 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-

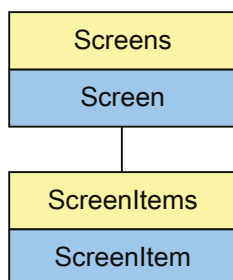
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Color (Page 604)	RW	-	-	Specifies the line color.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 670)	RW	-	-	Specifies the angle by which the end point deviates from the zero position (0°).
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
Radius (Page 880)	RW	-	-	Specifies the radius.
Size	-	-	-	-
StartAngle (Page 960)	RW	-	-	Specifies the angle by which the start point deviates from the zero position (0°).
Style (Page 985)	RW	-	-	Specifies the line style.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-24 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Clock (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Clock" object. The Clock object is an element of the ScreenItems list.

Type identifier in VBS

Clock

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-25 Properties

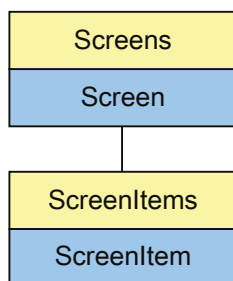
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Analog (Page 514)	RW	-	-	Specifies whether the clock is displayed as an analog clock.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern of the selected object.
BorderColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
DialColor (Page 653)	RW	RW	RW	Specifies the color of the dial for the selected object.
EdgeStyle	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
Flashing (Page 696)	RW	-	-	Specifies whether the selected object flashes in runtime.
Font (Page 709)	RW	-	-	Specifies the font.
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
HourNeedleHeight (Page 743)	RW	RW	RW	Specifies the length of the hour hand in the "Clock" object.
HourNeedleWidth (Page 744)	RW	RW	RW	Specifies the width of the hour hand in the "Clock" object.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the object.
Location	-	-	-	-
LockSquaredExtent (Page 797)	RW	-	-	Specifies whether the size of the clock can be adjusted with the mouse.
MinuteNeedleHeight (Page 827)	RW	RW	RW	Specifies the length of the minute hand in the "Clock" object.
MinuteNeedleWidth (Page 827)	RW	RW	RW	Specifies the width of the minute hand in the "Clock" object.
Name	RW	-	-	Specifies the object name.
NeedleBorderColor (Page 835)	RW	RW	RW	Specifies the line color of the pointer in the "Clock" object.
NeedleColor (Page 836)	RW	RW	RW	Specifies the hand color in the "Clock" object.
NeedleFillStyle (Page 837)	RW	-	-	Specifies whether the pointers are displayed as filled or transparent.
NumberStyle	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
Picture (Page 868)	RW	-	-	Specifies the screen to be displayed in the graphics object in Runtime.
SecondNeedleHeight (Page 906)	RW	RW	RW	Specifies the length of the seconds hand in the "Clock" object.
SecondNeedleWidth (Page 907)	RW	RW	RW	Specifies the width of the seconds hand in the "Clock" object.
ShowFocusRectangle (Page 938)	RW	-	-	Specifies whether the button is given a selection border when it is activated in Runtime.
ShowTicks (Page 950)	RW	RW	RW	Specifies whether the marking lengths are shown in the scale of the selected object.
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TickDistance	-	-	-	-
TicksColor (Page 1011)	RW	RW	RW	Specifies the color of the hour markers on the face of the "Clock" object.
TickStyle	-	-	-	-
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the selected object.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-26 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

ComboBox (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "Check box" object. The ComboBox object is an element of the ScreenItems list.

Type identifier in VBS

HMIComboBox

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-27 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (Page 518)	RW	-	-	Specifies whether the reason for operating this ob- ject is logged.
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
CountVisibleItems (Page 642)	RW	-	-	Specifies how many lines the drop-down list is to contain.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
FillPatternColor	-	-	-	-
FitToLargest	-	-	-	-
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text is shown underlined.
ForeColor (Page 716)	RW	-	-	Specifies the font color of the text.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalAlignment (Page 740)	RW	-	-	Specifies the horizontal alignment of the text.
Index (Page 745)	RW	-	-	Specifies the index for the selected text field.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
LogOperation (Page 798)	RW	-	-	Specifies whether, after operating this object, an alarm is output to the alarm system.
Name	-	-	-	-
SelectedIndex (Page 912)	RW	-	-	Specifies the index of which the associated text is highlighted in the combo box or list box.
SelectedText (Page 914)	RW	-	-	Specifies the text for the entry referenced with "SelectedIndex".
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

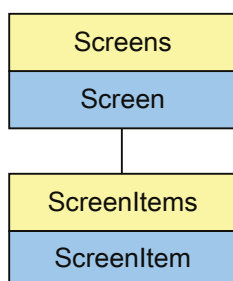
Properties	RT Professional	RT Advanced	Panel RT	Description
Text (Page 1004)	RW	-	-	Specifies the label for the text field.
TextHandles	-	-	-	-
Texts	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-28 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Connector (RT Professional)

Description



Represents the "Connector" object. The Connector object is an element of the ScreenItems list.

Type identifier in VBS

HMIConnector

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-29 Properties

Properties	RT Profes-sional	RT Ad-vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BorderEndStyle (Page 569)	RW	-	-	Specifies the type of line ends.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
Bounds	-	-	-	-

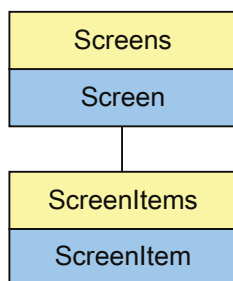
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	-	-	Specifies the line color.
ConnectionType (Page 635)	RW	-	-	Specifies the type of connector.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DeviceStyle	-	-	-	-
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
EndStyle (Page 671)	RW	-	-	Specifies how the line end is displayed.
FirstConnectedObject	-	-	-	-
FirstConnectedObjectIndex (Page 693)	RW	-	-	Specifies the index number of the upper connector point.
FirstConnectedObjectName (Page 694)	RW	-	-	Specifies the name of the object that is docked to the upper connector point.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	-	-	Specifies the height.
LastConnectedObject	-	-	-	-
LastConnectedObjectIndex (Page 766)	RW	-	-	Specifies the index number of the connection point to the last connected object.
LastConnectedObjectName (Page 766)	RW	-	-	Specifies the name of the object that is docked to the lower connector point.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
NoHitTest	-	-	-	-
Points	-	-	-	-
PointsCount	-	-	-	-
Size	-	-	-	-
StartStyle (Page 961)	RW	-	-	Specifies how the line start is displayed.
Style (Page 985)	RW	-	-	Specifies the line style.
SwapFirstWithLastConnection (Page 987)	RW	-	-	Specifies that the first and last connection should be swapped.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-30 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

DateTimeField (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "Date/time field" object. The DateTimeField object is an element of the ScreenItems list.

Type identifier in VBS

HMIDateTimeField

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-31 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	-	RW	RW	Specifies operating rights.
BackColor (Page 530)	-	RW	RW	Specifies the background color.
BackFillStyle	-	-	-	-
BorderBackColor	-	-	-	-
BorderColor (Page 566)	-	RW	RW	Specifies the line color.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderWidth	-	-	-	-
BottomMargin	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
DisplayCentury	-	-	-	-
DisplaySystemTime	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
FillPatternColor	-	-	-	-
FitToLargest	-	-	-	-
Flashing	-	-	-	-
Font	-	-	-	-
ForeColor (Page 716)	-	RW	RW	Specifies the font color of the text.
Format	-	-	-	-
Height	-	-	-	-
HelpText (Page 728)	-	RW	RW	Specifies the tooltip that is shown in Runtime as user help for the specified object.
HorizontalAlignment (Page 740)	-	RW	RW	Specifies the horizontal alignment of the text.
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
LeftMargin	-	-	-	-
LineWrap	-	-	-	-
Location	-	-	-	-
LongDateTimeFormat	-	-	-	-
Mode	-	-	-	-
Name	-	-	-	-
ProcessValue	-	-	-	-
RightMargin	-	-	-	-
ShowDate	-	-	-	-
ShowTime	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextOrientation	-	-	-	-
TimeDisplayMode	-	-	-	-
Top (Page 1086)	-	RW	RW	Specifies the value of the Y coordinate.
TopMargin	-	-	-	-

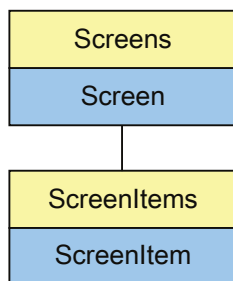
Properties	RT Professional	RT Advanced	Panel RT	Description
UseDesignColorSchema	-	-	-	-
UseTwoHandOperation	-	-	-	-
VerticalAlignment (Page 1204)	-	RW	RW	Specifies the vertical alignment of the text in the specified object.
Visible (Page 1208)	-	RW	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-32 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.

DiskSpaceView (RT Professional)

Description



Represents the "Memory space view" object. The DiskSpaceView object is an element of the ScreenItems list.

Type identifier in VBS

IXDiskSpaceView

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Alarm	-	-	-	-
AlarmColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Drive	-	-	-	-
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
Free	-	-	-	-
FreePercent (Page 719)	R	-	-	Returns the measured values for the free disk space as a percentage.
Height (Page 724)	RW	-	-	Specifies the height.
Interval (Page 753)	R	-	-	Returns the time interval for updating the displayed measured values.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
Location	-	-	-	-
Name (Page 831)	R	-	-	Returns the name.
NormalColor (Page 838)	R	-	-	Returns the color of the used memory space in the memory space view when memory availability is not critical.
Size	-	-	-	-
TabIndex (Page 988)	R	-	-	Returns the position in the order that the object takes when you toggle between the objects with the tab key.
TabIndexAlpha (Page 989)	R	-	-	Returns the position in the order that the object takes when you toggle between the objects with the tab key in Alpha Cursor mode.
Tolerance (Page 1056)	R	-	-	Returns the limit for the memory space view as of which a deviation will be reported.
ToleranceColor (Page 1056)	R	-	-	Returns the colors in which the bar of the memory space view is displayed as soon as the tolerance range is exceeded.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Total (Page 1090)	R	-	-	Returns the storage capacity.

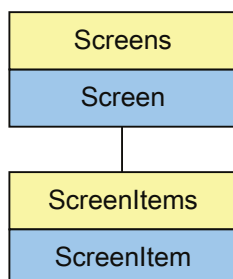
Properties	RT Professional	RT Advanced	Panel RT	Description
Used (Page 1151)	R	-	-	Returns the amount of used memory space.
UsedPercent (Page 1156)	R	-	-	Returns the measured values for the occupied memory space as a percentage.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Warning (Page 1212)	R	-	-	Returns the percentage of occupied memory space as of which a warning is generated.
WarningColor (Page 1213)	R	-	-	Returns the color in which the bar of the memory space view is displayed as soon as the warning range is reached.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-33 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Ellipse (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Ellipse" object. The Ellipse object is an element of the ScreenItems list.

Type identifier in VBS

HMIEllipse

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-34 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	RW	RW	RW	Specifies the background color of the selected object.
BackFillStyle (Page 535)	RW	RW	RW	Specifies the fill pattern of the selected object.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 566)	RW	RW	RW	Specifies the line color of the selected object.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines for the selected object.
BorderWidth (Page 581)	RW	RW	RW	Specifies the line thickness of the selected object.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the specified object.
DeviceStyle	-	-	-	-
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
EdgeStyle (Page 661)	RW	RW	RW	Specifies the line style of the selected object.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line for the selected object.
Height (Page 724)	RW	RW	RW	Specifies the height of the selected object.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the selected object.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (Page 880)	RW	RW	RW	Specifies the secondary axis of the specified object.
RadiusWidth (Page 881)	RW	RW	RW	Specifies the main axis of the specified object.
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the selected object.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.

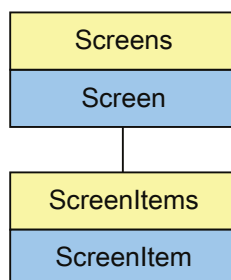
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	RW	RW	Specifies the width of the object in pixels.

Table 1-35 Methods

Method	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

EllipseSegment (RT Professional)

Description



Represents the "EllipseSegment" object. The EllipseSegment object is an element of the ScreenItems list.

Type identifier in VBS

HMIEllipseSegment

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-36 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 670)	RW	-	-	Specifies the angle by which the end point deviates from the zero position (0°).
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

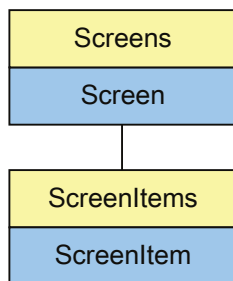
Properties	RT Professional	RT Advanced	Panel RT	Description
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (Page 880)	RW	-	-	Specifies the secondary axis.
RadiusWidth (Page 881)	RW	-	-	Specifies the main axis.
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
StartAngle (Page 960)	RW	-	-	Specifies the angle by which the start point deviates from the zero position (0°).
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-37 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

EllipticalArc (RT Professional)

Description



Represents the "EllipticalArc" object. The EllipticalArc object is an element of the ScreenItems list.

Type identifier in VBS

HMIEllipticalArc

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

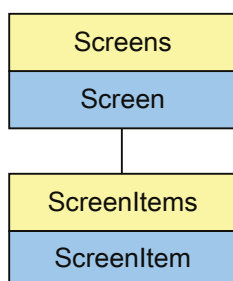
Table 1-38 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	-	-	Specifies the line color.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 670)	RW	-	-	Specifies the angle by which the end point deviates from the zero position (0°).
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (Page 880)	RW	-	-	Specifies the secondary axis.
RadiusWidth (Page 881)	RW	-	-	Specifies the main axis.
Size	-	-	-	-
StartAngle (Page 960)	RW	-	-	Specifies the angle by which the start point deviates from the zero position (0°).
Style (Page 985)	RW	-	-	Specifies the line style.

Properties	RT Professional	RT Advanced	Panel RT	Description
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorScheme (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-39 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

FunctionTrendControl (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "f(x)FunctionTrendView" object. The FunctionTrendControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIFunctionTrendControl

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-40 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettings- ForDesignMode	-	-	-	-
AvailableStatusbarEle- ments	-	-	-	-
AvailableToolbarBut- tons	-	-	-	-
BackColor (Page 530)	RW	-	-	Specifies the background color.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (Page 591)	RW	-	-	Specifies the text to be displayed in the header.
Closeable (Page 603)	RW	-	-	
ConnectTrendWindows (Page 636)	RW	-	-	Specifies whether the configured trend views are connected.
ControlDesignMode (Page 638)	RW	-	-	Specifies the control design.
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (Page 678)	RW	-	-	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 679)	RW	-	-	Specifies the target directory for the exported runtime data.
ExportFileExtension (Page 679)	RW	-	-	Specifies the file extension of the export file.
ExportFilename (Page 680)	RW	-	-	Specifies the name of the target file for the exported runtime data.
ExportFilename-Changeable (Page 681)	RW	-	-	Specifies whether the export file name can be changed in runtime.
ExportFormat	-	-	-	-
ExportFormatGuid (Page 682)	RW	-	-	Specifies the assignment of ID number and export provider.
ExportFormatName (Page 683)	RW	-	-	Specifies the export file format.
ExportParameters (Page 684)	RW	-	-	Specifies the parameters of the selected format by means of the properties dialog.
ExportSelection (Page 684)	RW	-	-	Specifies which runtime data of the control is exported.
ExportShowDialog (Page 685)	RW	-	-	Specifies whether the data export dialog is shown in runtime.
Font (Page 709)	RW	-	-	Specifies the font.
GraphDirection (Page 721)	RW	-	-	Specifies at which border of the trend view the current values are displayed.
Height (Page 724)	RW	-	-	-
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LineColor (Page 790)	RW	-	-	Specifies the color of the window separation lines.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
LoadDataImmediately (Page 796)	RW	-	-	Specifies whether the tag values are downloaded from the logs for the time range to be displayed when you call a screen.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Location	-	-	-	-
Moveable (Page 829)	RW	-	-	Specifies whether the object can be moved in run-time.
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
Online (Page 843)	RW	-	-	Specifies start and stop of the updating.
PrintJob (Page 876)	RW	-	-	Specifies a print job that is created in the "Reports" editor.
RTPersistence (Page 891)	RW	-	-	Specifies whether online configurations are retained after a screen change.
RTPersistenceAutho- rization (Page 892)	RW	-	-	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 893)	RW	-	-	Specifies how online configurations of WinCC are retained.
ShowRuler (Page 942)	RW	-	-	Specifies whether a scale division (help line) is displayed for an axis label of the object "OnlineTrendControl".
ShowRulerInAxis (Page 943)	RW	-	-	Specifies whether rulers are also displayed in the time axes.
ShowScrollbars (Page 944)	RW	-	-	Specifies whether scroll bars are displayed.
ShowTitle (Page 951)	RW	-	-	Specifies the style of the window border and window title for the object.
ShowTrendIcon (Page 953)	RW	-	-	Specifies whether an icon is displayed below the value axes.
Size	-	-	-	-
Sizeable (Page 956)	RW	-	-	Specifies that the size of an object can be changed in runtime.
StatusbarBackColor (Page 963)	RW	-	-	Specifies the background color for status bars.
StatusbarElementAdd (Page 964)	RW	-	-	Creates a new, user-defined status bar element.
StatusbarElementAuto- Size (Page 965)	RW	-	-	Specifies whether the width of a status bar element referenced with "StatusbarElementIndex" is set automatically.
StatusbarElement- Count (Page 966)	RW	-	-	Specifies the number of configured status bar elements.
StatusbarElementIcon- Id (Page 967)	RW	-	-	References a status bar element with its icon ID.
StatusbarElementID (Page 967)	RW	-	-	References a status bar element with its element ID.
StatusbarElementIndex (Page 968)	RW	-	-	References a status bar element.

Properties	RT Professional	RT Advanced	Panel RT	Description
StatusbarElement-Name (Page 969)	RW	-	-	Specifies the name of a status bar element referenced with "StatusbarElementIndex".
StatusbarElementRemove (Page 970)	RW	-	-	Removes the user-defined status bar element referenced with its name.
StatusbarElementRename (Page 971)	RW	-	-	Specifies the new name of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementRepos (Page 972)	RW	-	-	Specifies the position of an element referenced with "StatusbarElementIndex" in the status bar of the object.
StatusbarElements	-	-	-	-
StatusbarElementText (Page 973)	RW	-	-	Specifies the text of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementTooltipText (Page 974)	RW	-	-	Specifies the text for the tooltip of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementUser-Defined (Page 974)	RW	-	-	Specifies whether the configuration engineer has added a status bar element referenced with "StatusbarElementIndex" as a new user-defined element.
StatusbarElementVisible (Page 975)	RW	-	-	Specifies whether the status bar element referenced with "StatusbarElementIndex" is displayed in the object.
StatusbarElementWidth (Page 976)	RW	-	-	Specifies the width in pixels of a status bar element referenced with "StatusbarElementIndex".
StatusbarFont (Page 977)	RW	-	-	Specifies the font of the texts in the status bar.
StatusbarFontColor (Page 978)	RW	-	-	Specifies the color of the texts in the status bar.
StatusbarShowTooltips (Page 979)	RW	-	-	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 980)	RW	-	-	Specifies the default text in the status bar.
StatusbarUseBackColor (Page 980)	RW	-	-	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 981)	RW	-	-	Specifies whether the status bar of the control is displayed.
SupportsUserDefined-ToolbarButtons	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TimeBase (Page 1027)	RW	-	-	Specifies the time zone in which the time values are displayed.
ToolbarAlignment (Page 1063)	RW	-	-	Specifies the position of the toolbar.
ToolbarBackColor (Page 1064)	RW	-	-	Specifies the background color of the toolbar.
ToolbarButtonActive (Page 1065)	RW	-	-	Specifies whether the function linked to the button is activated in runtime.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ToolBarButtonAdd (Page 1066)	RW	-	-	Creates a new user-defined button in the object's toolbar.
ToolBarButtonAuthori- zation (Page 1066)	RW	-	-	Specifies the authorization for the selected key function.
ToolBarButtonBe- ginGroup (Page 1067)	RW	-	-	Inserts a leading separator (vertical line) for the selected key function on the toolbar.
ToolBarButtonClick (Page 1068)	RW	-	-	Clicks a toolbar button.
ToolBarButtonCount (Page 1069)	RW	-	-	Specifies the number of configured buttons in the toolbar.
ToolBarButtonEnabled (Page 1070)	RW	-	-	Specifies whether a user-defined toolbar button can be operated.
ToolBarButtonHotKey (Page 1071)	RW	-	-	Specifies the hot key for the selected object button.
ToolBarButtonID (Page 1071)	RW	-	-	References a button with its ID.
ToolBarButtonIndex (Page 1072)	RW	-	-	References a button.
ToolBarButtonLocked (Page 1073)	RW	-	-	Specifies whether the locked, pressed state of a user-defined button referenced with "ToolBarButtonIndex" is displayed.
ToolBarButtonName (Page 1074)	RW	-	-	Specifies the name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRemove (Page 1075)	RW	-	-	Removes a user-defined button referenced with its name.
ToolBarButtonRename (Page 1076)	RW	-	-	Specifies the new name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRepos (Page 1077)	RW	-	-	Specifies the position of a button referenced with "ToolBarButtonIndex" in the toolbar of the object.
ToolBarButtons	-	-	-	-
ToolBarButtonTooltip- Text (Page 1078)	RW	-	-	Specifies the tooltip text for the user-defined button in the toolbar.
ToolBarButtonUserDe- fined (Page 1079)	RW	-	-	Specifies whether the project engineer has added the toolbar button as new user-defined button.
ToolBarButtonVisible (Page 1080)	RW	-	-	Specifies whether the button referenced with "ToolBarButtonIndex" is displayed in the toolbar.
ToolBarShowTooltips (Page 1081)	RW	-	-	Specifies whether the tooltips for the key functions are displayed in runtime.
ToolBarUseBackColor (Page 1082)	RW	-	-	Specifies whether the background color for the toolbar is shown.
ToolBarUseHotKeys (Page 1083)	RW	-	-	Specifies whether the hotkeys are activated for buttons in the toolbar.
ToolBarVisible (Page 1084)	RW	-	-	Specifies whether the toolbar of the control is displayed.
Top (Page 1086)	-	R	R	Returns the value of the Y coordinate.
TrendActualize (Page 1095)	RW	-	-	Specifies whether the selected trend is updated.

Properties	RT Professional	RT Advanced	Panel RT	Description
TrendAdd (Page 1095)	RW	-	-	Creates a new trend.
TrendBeginTime (Page 1099)	RW	-	-	Defines the start time of the time range for data transfer to the selected trend.
TrendColor (Page 1099)	RW	-	-	Specifies the color of a referenced trend in the trend view.
TrendCount (Page 1100)	RW	-	-	Specifies the number of configured trends.
TrendEndTime (Page 1101)	RW	-	-	Defines the end of the time range for data connections of a selected trend.
TrendExtendedColor-Set (Page 1101)	RW	-	-	Specifies whether the point and fill colors of the trend are shown.
TrendFill (Page 1102)	RW	-	-	Specifies whether the area below the trend is shown filled.
TrendFillColor (Page 1103)	RW	-	-	Defines the fill color of the trend.
TrendIndex (Page 1104)	RW	-	-	References a configured trend.
TrendLabel (Page 1105)	RW	-	-	Defines the label text for a trend referenced with "TrendIndex".
TrendLineStyle (Page 1106)	RW	-	-	Specifies which line type is used for display of the trend.
TrendLineType (Page 1106)	RW	-	-	Specifies how the trend is displayed.
TrendLineWidth (Page 1107)	RW	-	-	Specifies the line thickness of the selected trend in pixels.
TrendLowerLimit (Page 1108)	RW	-	-	Specifies the low limit of a tag that is displayed in the object as a trend.
TrendLowerLimitColor (Page 1108)	RW	-	-	Specifies the color that marks trend values which are below the value of "TrendLowerLimit".
TrendLowerLimitColoring (Page 1109)	RW	-	-	Specifies whether "TrendLowerLimitColor" is used to identify tag values which are less than the value of "TrendLowerLimit".
TrendMeasurePoints (Page 1110)	RW	-	-	Specifies the number of measurement points for visualization of the selected trend.
TrendName (Page 1110)	RW	-	-	Specifies the name of a trend referenced with "TrendIndex".
TrendPointColor (Page 1111)	RW	-	-	Specifies the color of the dots on the reference trend.
TrendPointStyle (Page 1112)	RW	-	-	Specifies how the trend points are displayed.
TrendPointWidth (Page 1112)	RW	-	-	Specifies the point width in pixels.
TrendProvider (Page 1113)	RW	-	-	Specifies the data source for a selected trend.
TrendProviderCLSID (Page 1114)	RW	-	-	Specifies the provider CLSID of the data for a trend.
TrendRangeType (Page 1114)	RW	-	-	Specifies the time range during which the selected trend is supplied with data.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TrendRemove (Page 1115)	RW	-	-	Removes a trend referenced with its name.
TrendRename (Page 1116)	RW	-	-	Specifies the new name of a trend referenced with "TrendIndex".
TrendRepos (Page 1116)	RW	-	-	Specifies the position of a trend referenced with "TrendIndex" in the trend window of the object.
Trends	-	-	-	-
TrendSelectTagNameX (Page 1118)	RW	-	-	Specifies that the dialog for selecting the tag name of the data source of the x-axis is displayed initially in runtime.
TrendSelectTagNameY (Page 1118)	RW	-	-	Specifies that the dialog for selecting the tag name for the data source of the Y axis is displayed initially in runtime.
TrendTagNameX (Page 1119)	RW	-	-	Specifies the name of the connected HMI tag or column for the X-axis.
TrendTagNameY (Page 1120)	RW	-	-	Specifies the name of the connected HMI tag or column for the Y-axis.
TrendTimeRangeBase (Page 1121)	RW	-	-	Specifies the time unit for calculating the time range.
TrendTimeRangeFac- tor (Page 1122)	RW	-	-	Specifies the factor for determining the time interval for which the referenced trend is displayed.
TrendTrendWindow (Page 1122)	RW	-	-	Specifies the trend window in which the selected trend is shown.
TrendUncertainColor (Page 1123)	RW	-	-	Specifies the color for values of uncertain status.
TrendUncertainColor- ing (Page 1124)	RW	-	-	Specifies that values of uncertain status are to be highlighted.
TrendUpperLimit (Page 1124)	RW	-	-	Specifies the high limit of a tag that is displayed in the specified object as a trend.
TrendUpperLimitColor (Page 1125)	RW	-	-	Specifies the color that marks trend values which are above the value of "TrendUpperLimit".
TrendUpperLimitColor- ing (Page 1126)	RW	-	-	Specifies whether the selection frame is shown with the color defined by the system.
TrendVisible (Page 1128)	RW	-	-	Specifies whether the trend referenced with "TrendIndex" is displayed in the object.
TrendWindowAdd (Page 1128)	RW	-	-	Creates a new trend view.
TrendWindowCoarse- Grid (Page 1129)	RW	-	-	Specifies if the grid lines for the main scale are displayed.
TrendWindowCoarse- GridColor (Page 1130)	RW	-	-	Specifies the color of the main grid for the referenced diagram in the object.
TrendWindowCount (Page 1130)	RW	-	-	Specifies the number of configured trend diagrams in the trend view.
TrendWindowFineGrid (Page 1131)	RW	-	-	Specifies whether grid lines are displayed for the secondary scaling.
TrendWindowFineGrid- Color (Page 1132)	RW	-	-	Specifies the color of the secondary grid for the referenced diagram in the object.

Properties	RT Professional	RT Advanced	Panel RT	Description
TrendWindowForegroundTrendGrid (Page 1132)	RW	-	-	Specifies whether only the grid lines for the foreground trend are displayed in the selected trend window.
TrendWindowGridInTrendColor (Page 1133)	RW	-	-	Specifies whether the grid lines for the main scaling are displayed in the trend color.
TrendWindowHorizontalGrid (Page 1134)	RW	-	-	Specifies whether the horizontal grid lines are shown.
TrendWindowIndex (Page 1134)	RW	-	-	References a trend view.
TrendWindowName (Page 1135)	RW	-	-	Specifies the name of a trend view referenced with "TrendWindowIndex".
TrendWindowRemove (Page 1136)	RW	-	-	Removes a trend view referenced with its name.
TrendWindowRename (Page 1136)	RW	-	-	Specifies the new name of a trend view referenced with "TrendWindowIndex".
TrendWindowRepos (Page 1137)	RW	-	-	Specifies the position of a trend view referenced with "TrendWindowIndex".
TrendWindowRulerColor (Page 1137)	RW	-	-	Specifies the color of the ruler.
TrendWindowRulerLayer (Page 1138)	RW	-	-	Specifies the display layer of the ruler in the trend view.
TrendWindowRulerStyle (Page 1139)	RW	-	-	Specifies the appearance of the ruler.
TrendWindowRulerWidth (Page 1140)	RW	-	-	Specifies the thickness of the ruler in pixels.
TrendWindows	-	-	-	-
TrendWindowSpacePortion (Page 1140)	RW	-	-	Specifies the range proportion of the referenced diagram in the diagram area of the object.
TrendWindowVerticalGrid (Page 1143)	RW	-	-	Specifies whether the vertical grid lines are displayed.
TrendWindowVisible (Page 1144)	RW	-	-	Specifies whether the trend view referenced with "TrendWindowIndex" is displayed.
TrendXAxis (Page 1144)	RW	-	-	Defines the x-axis to be used for the referenced trend.
TrendYAxis (Page 1145)	RW	-	-	Defines the y-axis to be used for the referenced trend.
UseTrendNameAsLabel (Page 1165)	RW	-	-	Specifies whether the "Name" or "Label" property is used as a designation for the trend in runtime.
Visible (Page 1208)	-	R	R	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.
XAxes	-	-	-	-
XAxisAdd (Page 1229)	RW	-	-	Creates a new X-axis.
XAxisAlignment (Page 1230)	RW	-	-	Specifies the alignment of the X-axis referenced with "XAxisIndex".
XAxisAutoPrecisions (Page 1230)	RW	-	-	Specifies whether the number of decimal places for an X-axis referenced with "XAxisIndex" is set automatically.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
XAxisAutoRange (Page 1231)	RW	-	-	Specifies whether the value range of an X-axis referenced with "XAxisIndex" is calculated automatically.
XAxisBeginValue (Page 1232)	RW	-	-	Specifies the lower end of the value range of an X-axis referenced with "XAxisIndex".
XAxisColor (Page 1232)	RW	-	-	Specifies the color of an X-axis referenced with "XAxisIndex".
XAxisCount (Page 1233)	RW	-	-	Specifies the number of configured X-axes.
XAxisEndValue (Page 1233)	RW	-	-	Specifies the upper end of the value range of an X-axis referenced with "XAxisIndex".
XAxisExponentialFormat (Page 1234)	RW	-	-	Specifies whether the values of an X-axis referenced with "XAxisIndex" are displayed in exponential notation.
XAxisIndex (Page 1235)	RW	-	-	References an X-axis.
XAxisInTrendColor (Page 1235)	RW	-	-	Specifies whether the color of the axis referenced with "XAxisIndex" corresponds to the trend color.
XAxisLabel (Page 1236)	RW	-	-	Specifies the label text for an X-axis referenced with "XAxisIndex".
XAxisName (Page 1236)	RW	-	-	Specifies the name of an X-axis referenced with "XAxisIndex".
XAxisPrecisions (Page 1237)	RW	-	-	Specifies the number of decimal places to be displayed for the values of an X-axis referenced with "XAxisIndex".
XAxisRemove (Page 1238)	RW	-	-	Removes an X-axis referenced with its name.
XAxisRename (Page 1238)	RW	-	-	Specifies the new name of an X-axis referenced with "XAxisIndex".
XAxisRepos (Page 1239)	RW	-	-	For multiple X-axes, specifies the position of an X-axis referenced with "XAxisIndex".
XAxisScalingType (Page 1239)	RW	-	-	Specifies the type of scaling for the X-axis referenced with "XAxisIndex".
XAxisTrendWindow (Page 1240)	RW	-	-	Specifies the trend diagram in which the axis referenced with "XAxisIndex" is displayed.
XAxisVisible (Page 1240)	RW	-	-	Specifies whether the X-axis referenced with "XAxisIndex" is displayed in the object.
YAxes	-	-	-	-
YAxisAdd (Page 1241)	RW	-	-	Creates a new Y-axis.
YAxisAlignment (Page 1242)	RW	-	-	Specifies the alignment of the Y-axis referenced with "YAxisIndex".
YAxisAutoPrecisions (Page 1242)	RW	-	-	Specifies whether the number of decimal places is specified automatically for a Y-axis referenced with "YAxisIndex".
YAxisAutoRange (Page 1243)	RW	-	-	Specifies whether the value range of a Y-axis referenced with "YAxisIndex" is calculated automatically.
YAxisBeginValue (Page 1243)	RW	-	-	Specifies the lower end of the value range of a Y-axis referenced with "YAxisIndex".

Properties	RT Professional	RT Advanced	Panel RT	Description
YAxisColor (Page 1244)	RW	-	-	Specifies the color of a Y-axis referenced with "YAxisIndex".
YAxisCount (Page 1245)	RW	-	-	Specifies the number of configured Y-axes.
YAxisEndValue (Page 1245)	RW	-	-	Specifies the upper end of the value range of a Y-axis referenced with "YAxisIndex".
YAxisExponentialFormat (Page 1246)	RW	-	-	Specifies whether the values of a Y-axis referenced with "YAxisIndex" are displayed in exponential notation.
YAxisIndex (Page 1246)	RW	-	-	References a Y-axis.
YAxisInTrendColor (Page 1247)	RW	-	-	Specifies whether the color of the axis referenced with "YAxisIndex" corresponds to the trend color.
YAxisLabel (Page 1248)	RW	-	-	Specifies the label text for a Y-axis referenced with "YAxisIndex".
YAxisName (Page 1248)	RW	-	-	Specifies the name of a Y-axis referenced with "YAxisIndex".
YAxisPrecisions (Page 1249)	RW	-	-	Specifies the number of decimal places to be displayed for the values of a Y-axis referenced with "YAxisIndex".
YAxisRemove (Page 1249)	RW	-	-	Removes a Y-axis referenced with its name.
YAxisRename (Page 1250)	RW	-	-	Specifies the new name of a Y-axis referenced with "YAxisIndex".
YAxisRepos (Page 1250)	RW	R	R	For multiple Y-axes, specifies the position of a Y-axis referenced with "YAxisIndex".
YAxisScalingType (Page 1251)	RW	R	R	Specifies the type of scaling for the Y-axis referenced with "YAxisIndex".
YAxisTrendWindow (Page 1251)	RW	R	R	Specifies the trend diagram in which the axis referenced with "YAxisIndex" is displayed.
YAxisVisible (Page 1252)	RW	R	R	Specifies whether the Y-axis referenced with "YAxisIndex" is displayed in the object.

Table 1-41 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 1260)	Yes	-	-	Executes the "Connect backup" key function of the control.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
DetachDB (Page 1266)	Yes	-	-	Executes the "Disconnect backup" key function of the control.

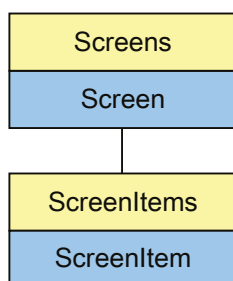
1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Export (Page 1267)	Yes	-	-	Executes the "Export log" or "Export data" key function of the control.
GetStatusbarElement (Page 1293)	Yes	-	-	Returns the element of the control status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 1294)	Yes	-	-	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetToolBarButton (Page 1301)	Yes	-	-	Returns the control toolbar button function designated by name or index as "ICCAxToolBarButton" type.
GetToolBarButtonCollection (Page 1302)	Yes	-	-	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
GetTrend (Page 1304)	Yes	-	-	Returns the f(t) or f(x) trend view trend designated by name or index as "ICCAxTrend" or "ICCAxFunctionTrend" type.
GetTrendCollection (Page 1305)	Yes	-	-	Returns the list of all trends of the f(t) or f(x) trend view as "ICCAxCollection" type.
GetTrendWindow (Page 1306)	Yes	-	-	Returns the trend window object of the f(t) or f(x) trend view designated by name or index as "ICCAxTrendWindow" type.
GetTrendWindowCollection (Page 1307)	Yes	-	-	Returns the list of all trend window objects of the f(t) or f(x) trend views as "ICCAxCollection" type.
GetXAxis (Page 1313)	Yes	-	-	Returns the f(x) trend view X-axis object that is specified by name or index as "ICCAxValueAxis" type.
GetXAxisCollection (Page 1314)	Yes	-	-	Returns a list of all f(x) trend view X-axis objects as "ICCAxCollection" type.
GetYAxis (Page 1316)	Yes	-	-	Returns the f(x) trend view Y-axis object that is specified by name or index as "ICCAxValueAxis" type.
GetYAxisCollection (Page 1317)	Yes	-	-	Returns the list of all f(x) trend view Y-axis objects as "ICCAxCollection" type.
MoveAxis (Page 1322)	Yes	-	-	Executes the "Move axes area" button function of the f(t) and f(x) trend views.
NextTrend (Page 1329)	Yes	-	-	Executes the "Next trend" button function of the f(t) and f(x) trend views.
OneToOneView (Page 1329)	Yes	-	-	Executes the "Original view" button function of the f(t) and f(x) trend views.
PreviousTrend (Page 1331)	Yes	-	-	Executes the "Previous trend" button function of the f(t) and f(x) trend views.
Print (Page 1331)	Yes	-	-	Executes the "Print" button function of the control.
ShowHelp (Page 1352)	Yes	-	-	Executes the "Help" button function of the control.
ShowPropertyDialog (Page 1356)	Yes	-	-	Executes the "Configuration dialog" button function of the control.
ShowTagSelection (Page 1360)	Yes	-	-	Executes the "Select data connection" button function of the control.
ShowTimeSelection (Page 1361)	Yes	-	-	Executes the "Select time range" button function of the control.
ShowTrendSelection (Page 1362)	Yes	-	-	Executes the "Select trends" button function of the f(t) and f(x) trend views.

Methods	RT Professional	RT Advanced	Panel RT	Description
StartStopUpdate (Page 1362)	Yes	-	-	Executes the "Start" or "Stop" button function of the control.
ZoomArea (Page 1370)	Yes	-	-	Executes the "Zoom area" button function of the f(t) and f(x) trend views.
ZoomInOut (Page 1370)	Yes	-	-	Executes the "Zoom +/-" button function of the f(t) and f(x) trend views.
ZoomInOutX (Page 1372)	Yes	-	-	Executes the "Zoom X axis +/-" button function of the f(x) trend view.
ZoomInOutY (Page 1372)	Yes	-	-	Executes the "Zoom Y axis +/-" button function of the f(x) trend view.
ZoomMove	Yes	-	-	Executes the "Move trend area" button function of the f(t) and f(x) trend views.

Gauge (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Gauge" object. The Gauge object is an element of the ScreenItems list.

Type identifier in VBS

HMIGauge

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-42 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AngleMax (Page 515)	RW	RW	RW	Specifies the angle of the scale end of the "Gauge" object.
AngleMin (Page 515)	RW	RW	RW	Specifies the angle of the scale start of the "Gauge" object.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BackPicture (Page 542)	RW	-	-	Specifies the background graphic.
BackStyle (Page 542)	RW	-	-	Specifies the background style.
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderInnerStyle3D (Page 576)	RW	-	-	Specifies the appearance of the inner part of the object border.
BorderOuterStyle3D (Page 577)	RW	-	-	Specifies the appearance of the outer part of the object border.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
BorderWidth3D (Page 584)	RW	-	-	Specifies the width of the inner border for 3D display.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionColor (Page 593)	-	RW	RW	Specifies the color of the text to be displayed in the title bar.
CaptionFont (Page 593)	RW	-	-	Specifies the character set for the caption.
CaptionText (Page 594)	RW	RW	RW	Specifies the text to be displayed in the header.
CaptionTop (Page 595)	RW	-	-	Specifies the distance between the instrument labels and the upper edge of the object.
CenterColor (Page 599)	RW	RW	RW	Specifies the color of the center point.
CenterSize (Page 599)	RW	-	-	Specifies the diameter of the scale center point.
CompatibilityMode	-	-	-	-
CornerRadius	-	-	-	-
DangerRangeColor (Page 644)	RW	RW	RW	Specifies the color of the danger range on the scale of the "Gauge" object.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
DangerRangeStart (Page 645)	RW	RW	RW	Specifies at which scale value the danger range of the "Gauge" object will start.
DangerRangeVisible (Page 646)	RW	RW	RW	Specifies whether the danger range in the scale of the "Gauge" object is to be displayed.
DeviceStyle	-	-	-	-
DialColor (Page 653)	RW	RW	RW	Specifies the color of the dial for the selected object.
DialFillStyle (Page 654)	RW	-	-	Specifies the type of background.
DialPicture (Page 654)	RW	-	-	Specifies a graphic for the dial.
DialSize (Page 655)	RW	-	-	Specifies the diameter of the scale in relation to the smaller value of the geometry attributes "Width" and "Height".
EdgeStyle (Page 661)	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
Flashing (Page 696)	RW	-	-	Specifies whether the selected object flashes in runtime.
Gradation (Page 720)	RW	RW	RW	Specifies the value difference between two main marks of the "Gauge" object.
InnerDialColor	-	-	-	-
InnerDialInnerDistance	-	-	-	-
InnerDialOuterDistane	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LimitRangeCollection	-	-	-	-
Location	-	-	-	-
LockSquaredExtent (Page 797)	RW	-	-	Specifies whether the size of the clock can be adjusted with the mouse.
MaximumValue (Page 803)	RW	RW	RW	Specifies the maximum value of the scale in the selected object.
MinimumValue (Page 826)	RW	RW	RW	Specifies the minimum value of the scale in the selected object.
Name (Page 831)	RW	-	-	Specifies the object name.
NeedleHeight	-	-	-	-
NormalRangeColor (Page 838)	RW	RW	RW	Specifies the color of the normal range on the scale of the "Gauge" object.
NormalRangeVisible (Page 839)	RW	RW	RW	Specifies whether the normal range in the scale of the "Gauge" object is displayed.
PointerColor (Page 873)	RW	RW	RW	Specifies the pointer color of the "Gauge" object.
ProcessValue (Page 877)	RW	RW	RW	Specifies the default for the value to be displayed.
ScaleLabelColor (Page 898)	RW	RW	RW	Specifies the color of the label for the scale marks of the "Gauge" object.
ScaleLabelFont (Page 899)	RW	-	-	Specifies the font for the scale label.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

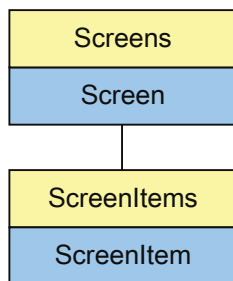
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ScaleTickColor (Page 900)	RW	RW	RW	Specifies the color of the scale gradation of the "Gauge" object.
ScaleTickLabelPosition (Page 901)	RW	-	-	Specifies the diameter of the imaginary circle on which the scale tick label is location.
ScaleTickLength (Page 901)	RW	-	-	Specifies the length of the main scale ticks.
ScaleTickPosition (Page 902)	RW	-	-	Specifies the diameter of the assumed circle on which the scale divisions are located.
ShowDecimalPoint (Page 932)	RW	-	-	Specifies whether the scale is labeled with decimal figures (decimal point and one decimal place) or with whole integers.
ShowInnerDial	-	-	-	-
ShowLimitRanges	-	-	-	-
ShowPeakValuePointer (Page 941)	RW	RW	RW	Specifies whether a slave pointer will be used for the selected object.
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UnitColor (Page 1146)	RW	RW	RW	Specifies the text color for the unit of measurement.
UnitFont (Page 1147)	RW	-	-	Specified the font of the unit of measurement.
UnitText (Page 1147)	RW	RW	RW	Specifies the text for the unit of measurement.
UnitTop (Page 1148)	RW	-	-	Specifies the distance from the unit of measurement to the upper edge of the object.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
WarningRangeColor (Page 1216)	RW	RW	RW	Specifies the color of the warning range on the scale of the "Gauge" object.
WarningRangeStart (Page 1217)	RW	RW	RW	Specifies at which scale value the warning range of the "Gauge" object will start.
WarningRangeVisible (Page 1218)	RW	RW	RW	Specifies whether the warning range in the scale of the "Gauge" object is to be displayed.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-43 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

GraphicIOField (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Graphic I/O field" object. The GraphicIOField object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode:

Property	"Input"	"Output"	"Input/Output"	"Two states"
BorderColor	--	--	--	x
Enabled	x	--	x	--
FocusColor	x	--	x	--
FocusWidth	x	--	x	--
HelpText	x	--	x	--
TransparentColor	x	x	x	--
UseTransparentColor	x	x	x	--

Type identifier in VBS

HMIGraphicIOField

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-44 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AboveUpperLimitColor	-	-	-	-
AdaptPicture	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	RW	RW	Specifies operating rights.
AutoSizing	-	-	-	-
BackColor (Page 530)	-	RW	RW	Specifies the background color.
BackFillStyle	-	-	-	-
BelowLowerLimitColor	-	-	-	-
BitNumber	-	-	-	-
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate	-	-	-	-
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingOnLimitViola- tion	-	-	-	-
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
FlashTransparentColor (Page 705)	RW	-	-	Specifies the color of the bitmap object of a flashing graphic that is set to "transparent".
FocusColor (Page 707)	-	RW	RW	Specifies the color of the focus border when the ob- ject has the focus.
FocusWidth (Page 708)	-	RW	RW	Specifies the border width when the object has the focus.
Height (Page 724)	RW	-	-	Specifies the height.
HelpText (Page 728)	-	RW	RW	Specifies the tooltip that is shown in Runtime as user help for the specified object.
IsImageMiddleAligned	-	-	-	-
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an ob- ject.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Mode	-	-	-	-
Name	-	-	-	-
OnValue	-	-	-	-
PictureList	-	-	-	-
PictureOff (Page 870)	RW	-	-	Specifies the picture that is displayed in the "Off" status.
PictureOn (Page 871)	RW	-	-	Specifies the graphic that is displayed in the "On" status.
PictureRotation	-	-	-	-
ProcessValue (Page 877)	RW	-	-	Specifies the default for the value to be displayed.
ScrollBarOrientation	-	-	-	-
ShowScrollBar	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.

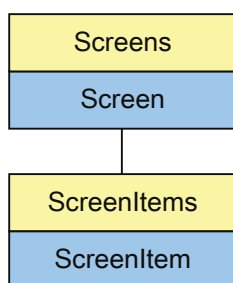
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TransparentColor (Page 1092)	RW	RW	RW	Specifies which color of the allocated graphic (*.bmp, *.dib) is to be set to "transparent".
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
UseFlashTransparent- Color (Page 1157)	RW	-	-	Specifies whether the color of the bitmap object of a flashing graphic is set to "transparent".
UseTransparentColor (Page 1163)	RW	RW	RW	Specifies whether the color defined with the "TransparentColor" property is shown as transparent.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-45 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

GraphicView (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Graphic view" object. The GraphicView object is an element of the ScreenItems list.

Type identifier in VBS

HMIGraphicView

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-46 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AdaptPicture	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
AutoSizing	-	-	-	-
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	RW	RW	Specifies whether the object can be operated in run-time.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
FitToLargest	-	-	-	-
Flashing	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
IsImageMiddleAligned	-	-	-	-
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Name	-	-	-	-
Picture (Page 868)	RW	-	-	Specifies the screen from the WinCC project graphics to be displayed in the object in runtime.
PictureRotation	-	-	-	-
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the object is filled.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
TransparentColor (Page 1092)	RW	RW	RW	Specifies which color of the allocated graphic (*.bmp, *.dib) is to be set to "transparent".
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.

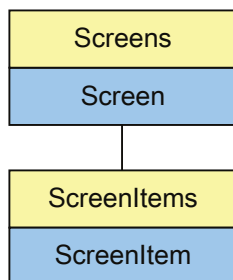
Properties	RT Professional	RT Advanced	Panel RT	Description
UseTransparentColor (Page 1163)	RW	RW	RW	Specifies whether the color defined with the "TransparentColor" property is shown as transparent.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-47 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

HTMLBrowser (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the object "HTML Browser". The HTMLBrowser object is an element of the ScreenItems-list.

Type identifier in VBS

HMIHTMLBrowser

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-48 Properties

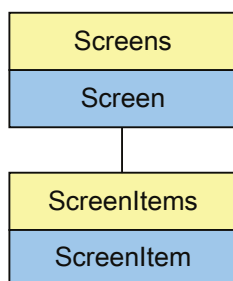
Properties	RT Professional	RT Advanced	Panel RT	Description
Address (Page 504)	RW	RW	RW	Specifies the web address that will be opened in the HTML browser.
AddressEnabled	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
BrowserTypeUsed	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
Flashing	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
ShowStatusBar (Page 948)	RW	-	-	Specifies whether the status bar is shown.
ShowToolBar	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	RW	RW	Specifies the width of the object in pixels.

Table 1-49 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
SetHtml (Page 1348)	Yes	-	-	Writes a HTML code in the display range of the HTML browser.

See also

SetHTML (Page 1348)

IOField (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "I/O field" object. The IOField object is an element of the ScreenItems list.

Type identifier in VBS

HMIIField

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-50 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AboveUpperLimitColor (Page 494)	RW	-	-	Sets the color of the specified object for the "High limit violated" event.
AcceptOnExit (Page 495)	RW	-	-	Specifies whether the input field is confirmed automatically when it is left.
AcceptOnFull (Page 495)	RW	-	-	Specifies whether the input field will be left and confirmed automatically when the defined number of values has been entered.
AdaptBorder (Page 502)	RW	-	-	Specifies whether the border of the object will be dynamically adapted to the text size.
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (Page 518)	RW	-	-	Specifies whether the reason for operating this object is logged.
Authorization (Page 521)	RW	RW	RW	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	RW	RW	RW	Specifies the background color of the selected object.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern of the selected object.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background of the selected object flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background for the selected object.
BelowLowerLimitColor (Page 549)	RW	-	-	Specifies the color for the "Low limit violated" event.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line for the selected object.
BorderColor	-	-	-	-
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".

Properties	RT Professional	RT Advanced	Panel RT	Description
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines for the selected object.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness of the selected object.
BorderWidth3D	-	-	-	-
BottomMargin	-	-	-	-
CanBeGrouped	-	-	-	-
ClearOnError (Page 602)	RW	-	-	Specifies whether an invalid input in this object is deleted automatically.
ClearOnFocus (Page 602)	RW	-	-	Specifies whether the field entry is deleted as soon as the I/O field is activated.
CornerRadius	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the specified object.
CursorControl (Page 643)	RW	-	-	Specifies whether the mouse cursor jumps to the next field of the TAB sequence after leaving a field.
DataFormat (Page 646)	RW	-	-	Specifies the display format of an I/O field.
DeviceStyle	-	-	-	-
EdgeStyle (Page 661)	RW	RW	RW	Specifies the line style of the selected object.
EditOnFocus (Page 663)	RW	-	-	Specifies whether data input is immediately possible if the input field is selected using the <Tab> key.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
FieldLength	-	-	-	-
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern for the selected object.
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingOnLimitViolation	-	-	-	-
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line for the selected object.
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text of the selected object is shown in bold.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FontItalic (Page 712)	RW	-	-	Specifies whether the text of the selected object is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font of the selected object.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text for the selected object.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text of the selected object is underlined.
ForeColor (Page 716)	RW	RW	RW	Specifies the font color of the text for the selected object.
FormatPattern (Page 718)	RW	-	-	Specifies the format of the output value.
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
HelpText (Page 728)	-	RW	RW	Specifies the tooltip that is shown in Runtime as user help for the specified object.
HiddenInput (Page 729)	RW	-	-	Specifies whether the input value or a * for each character is shown during the input.
HorizontalAlignment (Page 740)	RW	RW	RW	Specifies the horizontal alignment of the text within the selected object.
InputValue (Page 751)	RW	-	-	Specifies the input value.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the selected object.
LeftMargin	-	-	-	-
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
LineWrap	-	-	-	-
Location	-	-	-	-
LogOperation (Page 798)	RW	-	-	Specifies whether, after operating this object, an alarm is output to the alarm system.
LowerLimit (Page 799)	RW	-	-	Specifies the low limit for input values.
Mode (Page 828)	RW	-	-	Specifies the field type of the selected object.
Name	-	-	-	-
ProcessValue (Page 877)	RW	RW	RW	Specifies the default for the value to be displayed.
RightMargin	-	-	-	-
ShiftDecimalPoint	-	-	-	-
ShowBadTagState (Page 930)	RW	-	-	Defines whether the object is grayed out when a bad quality code or tag status is detected.
ShowLeadingZeros	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TextOrientation (Page 1007)	RW	-	-	Specifies the text orientation of the selected object.
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the object.
TopMargin	-	-	-	-
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
Unit (Page 1146)	RW	-	-	Specifies the unit of measurement.
UpperLimit (Page 1149)	RW	-	-	Specifies the high limit for input values.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with the shading defined in the active design.
UseTagLimitColors (Page 1163)	RW	-	-	Specifies whether the violation of the configured high and low limits for the object are highlighted in color.
UseTwoHandOperation (Page 1166)	-	R	R	Returns whether two-hand operation is enabled.
VerticalAlignment (Page 1204)	RW	RW	RW	Specifies the vertical alignment of the text in the specified object.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-51 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

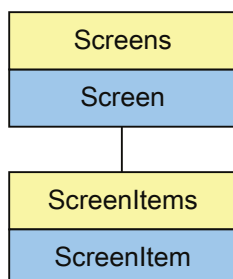
See also

BorderColor (Page 566)

1.5.4.3 Objects K-Z (Panels, Comfort Panels, RT Advanced, RT Professional)

Line (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Line" object. The Line object is an element of the ScreenItems list.

Type identifier in VBS

HMILine

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-52 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ActualPointIndex	-	-	-	-
ActualPointLeft	-	-	-	-
ActualPointTop	-	-	-	-
AllTagTypesAllowed	-	-	-	-

Properties	RT Professional	RT Advanced	Panel RT	Description
Authorization (Page 521)	RW	-	-	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	RW	RW	RW	Specifies the background color of the selected object.
BorderEndStyle (Page 569)	RW	-	-	Specifies the type of line ends for the selected object.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines for the selected object.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	RW	RW	Specifies the line color of the selected object.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the selected object.
DeviceStyle	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
EndLeft	-	-	-	-
EndStyle (Page 671)	RW	RW	RW	Specifies how the line end of the selected object is displayed.
EndTop	-	-	-	-
FillStyle (Page 691)	-	RW	RW	Specifies the fill pattern of the selected object.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line for the selected object.
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the selected object.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line end.
LineWidth (Page 794)	RW	RW	RW	Specifies the line thickness of the selected object.
Location	-	-	-	-
Name	-	-	-	-
Points	-	-	-	-
PointsCount	-	-	-	-
RotationAngle (Page 886)	RW	-	-	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 887)	RW	-	-	Specifies the X coordinate of the pivot point for the object in runtime.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

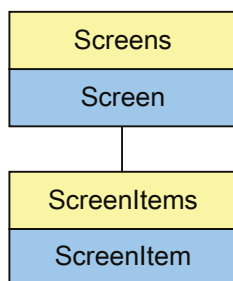
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
RotationCenterTop (Page 888)	RW	-	-	Specifies the Y coordinate of the pivot point for the object in runtime.
Size	-	-	-	-
StartLeft	-	-	-	-
StartStyle (Page 961)	RW	RW	RW	Specifies how the line start of the selected object is displayed.
StartTop	-	-	-	-
Style (Page 985)	RW	RW	RW	Specifies the line style of the selected object.
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the selected object.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-53 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Listbox (RT Professional)

Description



Represents the "Listbox" object. The Listbox object is an element of the ScreenItems list.

Type identifier in VBS

HMIListBox

Application

In the following example, the object with the name "ListBox1" is moved 10 pixels to the right:

```
'VBS21
Dim objListBox
Set objListBox = ScreenItems("ListBox1")
objListBox.Left = objListBox.Left + 10
```

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (Page 518)	RW	-	-	Specifies whether the reason for operating this object is logged.
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
CountVisibleItems (Page 642)	RW	-	-	Specifies how many lines the drop-down list is to contain.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
FillPatternColor	-	-	-	-
FitToLargest	-	-	-	-
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text is shown underlined.
ForeColor (Page 716)	RW	-	-	Specifies the font color of the text.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalAlignment (Page 740)	RW	-	-	Specifies the horizontal alignment of the text.
Index (Page 745)	RW	-	-	Specifies the index for the selected text field.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
LogOperation (Page 798)	RW	-	-	Specifies whether, after operating this object, an alarm is output to the alarm system.
Name	-	-	-	-

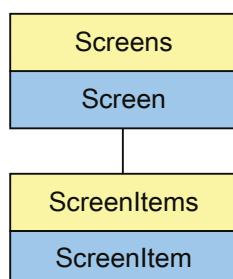
Properties	RT Professional	RT Advanced	Panel RT	Description
SelectedIndex (Page 912)	RW	-	-	Specifies the index of which the associated text is highlighted in the combo box or list box.
SelectedText (Page 914)	RW	-	-	Specifies the text for the entry referenced with "SelectedIndex".
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (Page 1004)	RW	-	-	Specifies the label for the text field.
TextHandles	-	-	-	-
Texts	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorScheme (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-54 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

MediaPlayer (RT Professional)

Description



Represents the "Media Player" object. The MediaPlayer object is an element of the ScreenItems list.

Type identifier in VBS

HMIMediaPlayer

Application

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```

'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
  
```

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-55 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AspectRatio (Page 519)	RW	-	-	Specifies whether the aspect ratio of the media player is maintained when the size is changed.
AutoStart	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
FileName (Page 687)	RW	-	-	Specifies the name of the file to be loaded.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
PictureSizeMode (Page 872)	RW	-	-	Specifies the size adjustment between the media player and the content to be displayed.
PlayCount	-	-	-	-
PlayEndless (Page 872)	RW	-	-	Specifies that the media file is to play endlessly.
PopupMenuEnabled	-	-	-	-
ShowControls (Page 931)	RW	-	-	Specifies that a toolbar is displayed.
ShowFeatureBackward (Page 933)	RW	-	-	Specifies that the "Backward" button is to be displayed in runtime.
ShowFeatureForward (Page 934)	RW	-	-	Specifies that the "Forward" button is displayed in runtime.
ShowFeature-FullScreen (Page 935)	RW	-	-	Specifies that the Media Player can be displayed as a full screen.
ShowFeatureFullVolume	-	-	-	-
ShowFeaturePause (Page 935)	RW	-	-	Specifies that the "Pause" button is displayed in runtime.
ShowFeaturePlay (Page 936)	RW	-	-	Specifies that the "Play" button is displayed in runtime.
ShowFeatureStop (Page 936)	RW	-	-	Specifies that the "Stop" button is displayed in runtime.
ShowStatusBar	-	-	-	-
ShowTracker (Page 952)	RW	-	-	Specifies that the tracker is displayed.
Size	-	-	-	-
StepSeconds (Page 984)	RW	-	-	Specifies the step interval in seconds after the "Forward" or "Backward" key is pressed.

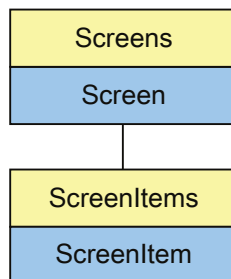
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-56 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

MultiLineEdit (RT Professional)

Description



Represents the "Editable text field" object. The MultiLineEdit object is an element of the ScreenItems list.

Type identifier in VBS

HMIMultiLineEdit

Application

In the following example, the object with the name "MultiLineEdit1" is moved 10 pixels to the right:

```
'VBS21
Dim objMultiLineEdit
Set objMultiLineEdit = ScreenItems("MultiLineEdit1")
objMultiLineEdit.Left = objMultiLineEdit.Left + 10
```

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-57 Properties

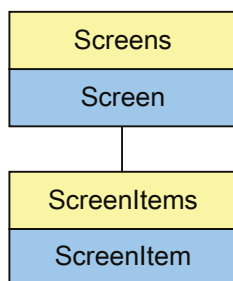
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text is shown in italics.
FontName	-	-	-	-
FontSize (Page 715)	RW	-	-	Specifies the font size of the text.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
FontUnderline (Page 715)	RW	-	-	Specifies whether the text is shown underlined.
ForeColor (Page 716)	RW	-	-	Specifies the font color of the text.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalAlignment (Page 740)	RW	-	-	Specifies the horizontal alignment of the text.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (Page 1004)	RW	-	-	Specifies the label for the text field.
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorScheme (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-58 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
GetSelectionText	Yes	-	-	-
SetSelection	Yes	-	-	-
SetSelectionText	Yes	-	-	-

OnlineTableControl (RT Professional)**Description**

Represents the "Table view" object. The OnlineTableControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIOnlineTableControl

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettings- ForDesignMode	-	-	-	-
AutoCompleteColumns (Page 523)	RW	-	-	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 523)	RW	-	-	Specifies whether empty rows are shown if the control is longer than the number of configured rows.
AutoSelectionColors (Page 525)	RW	-	-	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
AutoSelectionRectColor (Page 526)	RW	-	-	Specifies whether the selection frame is shown with the color defined by the system.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption	-	-	-	-
CellCut (Page 595)	RW	-	-	Specifies whether the contents of the cells are abbreviated if the cells are too narrow.
CellSpaceBottom (Page 596)	RW	-	-	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 597)	RW	-	-	Specifies the left indent used in the table cells.
CellSpaceRight (Page 597)	RW	-	-	Specifies the right indent used in the table cells.
CellSpaceTop (Page 598)	RW	-	-	Specifies the top margin of the table cells.
Closeable (Page 603)	RW	-	-	Specifies whether the object can be closed in runtime.
ColumnResize (Page 620)	RW	-	-	Specifies whether you can change the width of the columns.
ColumnScrollbar (Page 621)	RW	-	-	Specifies when the horizontal scroll bar is to be displayed.
ColumnTitleAlignment (Page 631)	RW	-	-	Specifies the alignment of the title in a column referenced with "ColumnIndex".
ColumnTitles (Page 632)	RW	-	-	Specifies whether the column header is displayed.
ControlDesignMode (Page 638)	RW	-	-	Specifies the control design.
Enabled (Page 664)	-	-	-	-
EnableEdit (Page 668)	RW	-	-	Specifies whether the data shown can be edited in runtime.
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (Page 678)	RW	-	-	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 679)	RW	-	-	Specifies the target directory for the exported runtime data.
ExportFileExtension (Page 679)	RW	-	-	Specifies the file extension of the export file.
ExportFilename (Page 680)	RW	-	-	Specifies the name of the target file for the exported runtime data.
ExportFilename-Changeable (Page 681)	RW	-	-	Specifies whether the export file name can be changed in runtime.
ExportFormat	-	-	-	-

Properties	RT Professional	RT Advanced	Panel RT	Description
ExportFormatGuid (Page 682)	RW	-	-	Specifies the assignment of ID number and export provider.
ExportFormatName (Page 683)	RW	-	-	Specifies the export file format.
ExportParameters (Page 684)	RW	-	-	Specifies the parameters of the selected format by means of the properties dialog.
ExportSelection (Page 684)	RW	-	-	Specifies which runtime data of the control is exported.
ExportShowDialog (Page 685)	RW	-	-	Specifies whether the data export dialog is shown in runtime.
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Font (Page 709)	RW	-	-	Specifies the font.
GridLineColor (Page 721)	RW	-	-	Specifies the color for the grid lines.
GridLineWidth (Page 723)	RW	-	-	Specifies the width of the separation lines in pixels.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalGridLines (Page 741)	RW	-	-	Specifies whether horizontal separating lines are displayed.
IconSpace (Page 744)	RW	-	-	Specifies the spacing between icons and text in the table cells.
Layer	-	-	-	-
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineBackgroundColor	-	-	-	-
LineColor (Page 790)	RW	-	-	Specifies the color of the window separation lines.
LineStyle	-	-	-	-
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
LoadDataImmediately (Page 796)	RW	-	-	Specifies whether the tag values are downloaded from the logs for the time range to be displayed when you call a screen.
Location	-	-	-	-
MaximumNumberOfTimeColumns	-	-	-	-
MaximumNumberOfValueColumns	-	-	-	-
MinimumNumberOfTimeColumns	-	-	-	-
MinimumNumberOfValueColumns	-	-	-	-
Moveable (Page 829)	RW	-	-	Specifies whether the object can be moved in runtime.
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
OcxStateForEs2Rt	-	-	-	-
Online (Page 843)	RW	-	-	Specifies start and stop of the updating.
PrintJob (Page 876)	RW	-	-	Specifies a print job that is created in the "Reports" editor.
RowScrollbar (Page 889)	RW	-	-	Specifies when the vertical scroll bar is displayed.
RowTitleAlignment (Page 890)	RW	-	-	Specifies the type of row header alignment.
RowTitles (Page 891)	RW	-	-	Specifies whether numbered column headers are displayed.
RTPersistence (Page 891)	RW	-	-	Specifies whether online configurations are retained after a screen change.
RTPersistenceAutho- rization (Page 892)	RW	-	-	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 893)	RW	-	-	Specifies how online configurations of WinCC are retained.
SelectedCellColor (Page 909)	RW	-	-	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 910)	RW	-	-	Specifies the font color of the selected cell.
SelectedRowColor (Page 912)	RW	-	-	Specifies the background color of the selected row.
SelectedRowForeColor (Page 913)	RW	-	-	Specifies the font color of the selected row.
SelectedTitleColor (Page 915)	RW	-	-	Specifies the background color of the selected table header.
SelectedTitleForeColor (Page 916)	RW	-	-	Specifies the font color of the selected table header.
SelectionColoring (Page 918)	RW	-	-	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 920)	RW	-	-	Enables the use of a selection border for selected cells or rows.
SelectionRectColor (Page 920)	RW	-	-	Specifies the color of the selection rectangle in the alarm window.
SelectionRectWidth (Page 921)	RW	-	-	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionType (Page 922)	RW	-	-	Specifies the number of lines you can mark.
ShowSortButton (Page 945)	RW	-	-	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 946)	RW	-	-	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 946)	RW	-	-	Specifies whether a sort index is displayed.
ShowTitle (Page 951)	RW	-	-	Specifies the style of the window border and window title for the object.
Size	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Sizeable (Page 956)	RW	-	-	Specifies that the size of an object can be changed in runtime.
SortSequence (Page 958)	RW	-	-	Specifies how the sorting order is changed if the operator clicks on the column header in runtime.
StatusbarBackColor (Page 963)	RW	-	-	Specifies the background color for status bars.
StatusbarElementAdd (Page 964)	RW	-	-	Creates a new, user-defined status bar element.
StatusbarElementAuto-Size (Page 965)	RW	-	-	Specifies whether the width of a status bar element referenced with "StatusbarElementIndex" is set automatically.
StatusbarElement-Count (Page 966)	RW	-	-	Specifies the number of configured status bar elements.
StatusbarElementIconId (Page 967)	RW	-	-	References a status bar element with its icon ID.
StatusbarElementID (Page 967)	RW	-	-	References a status bar element with its element ID.
StatusbarElementIndex (Page 968)	RW	-	-	References a status bar element.
StatusbarElement-Name (Page 969)	RW	-	-	Specifies the name of a status bar element referenced with "StatusbarElementIndex".
StatusbarElementRemove (Page 970)	RW	-	-	Removes the user-defined status bar element referenced with its name.
StatusbarElementRename (Page 971)	RW	-	-	Specifies the new name of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementRepos (Page 972)	RW	-	-	Specifies the position of an element referenced with "StatusbarElementIndex" in the status bar of the object.
StatusbarElements	-	-	-	-
StatusbarElementText (Page 973)	RW	-	-	Specifies the text of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementTooltipText (Page 974)	RW	-	-	Specifies the text for the tooltip of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementUser-Defined (Page 974)	RW	-	-	Specifies whether the configuration engineer has added a status bar element referenced with "StatusbarElementIndex" as a new user-defined element.
StatusbarElementVisible (Page 975)	RW	-	-	Specifies whether the status bar element referenced with "StatusbarElementIndex" is displayed in the object.
StatusbarElementWidth (Page 976)	RW	-	-	Specifies the width in pixels of a status bar element referenced with "StatusbarElementIndex".
StatusbarFont (Page 977)	RW	-	-	Specifies the font of the texts in the status bar.
StatusbarFontColor (Page 978)	RW	-	-	Specifies the color of the texts in the status bar.
StatusbarShowTooltips (Page 979)	RW	-	-	Specifies whether tooltips are displayed for the elements of the status bar in runtime.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
StatusbarText (Page 980)	RW	-	-	Specifies the default text in the status bar.
StatusbarUseBackCol- or (Page 980)	RW	-	-	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 981)	RW	-	-	Specifies whether the status bar of the control is displayed.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (Page 992)	RW	-	-	Specifies the background color of the table rows in the object.
TableColor2 (Page 992)	RW	-	-	Specifies the second background color of the table rows in the object.
TableForeColor (Page 994)	RW	-	-	Specifies the text color used in the table cells of the object.
TableForeColor2 (Page 995)	RW	-	-	Specifies the second text color used in the table cells of the object.
TimeBase (Page 1027)	RW	-	-	Specifies the time zone in which the time values are displayed.
TimeColumnActualize (Page 1028)	RW	-	-	Specifies whether the values of the time column referenced with "TimeColumnIndex" are updated.
TimeColumnAdd (Page 1029)	RW	-	-	Creates a new time column.
TimeColumnAlignment (Page 1029)	RW	-	-	Specifies how the text in the time column referenced with "TimeColumnIndex" is aligned.
TimeColumnBackColor (Page 1030)	RW	-	-	Specifies the background color of the time column referenced with "TimeColumnIndex".
TimeColumnBeginTime (Page 1031)	RW	-	-	Specifies the start of the time range for the time column referenced with "TimeColumnIndex".
TimeColumnCaption (Page 1031)	RW	-	-	Specifies the label of the time column referenced with "TimeColumnIndex".
TimeColumnCount (Page 1032)	RW	-	-	Specifies the number of configured time columns.
TimeColumnDateFormat (Page 1032)	RW	-	-	Specifies the date format for displaying the time column referenced with "TimeColumnIndex".
TimeColumnEndTime (Page 1033)	RW	-	-	Defines the end of the time range of for a time column referenced with "TimeColumnIndex".
TimeColumnForeColor (Page 1033)	RW	-	-	Specifies the font color of the time column referenced with "TimeColumnIndex".
TimeColumnHideText (Page 1034)	RW	-	-	Specifies whether the content of the time column referenced with "TimeColumnIndex" is displayed as text.
TimeColumnHideTitle- Text (Page 1035)	RW	-	-	Specifies whether the header of the time column referenced with "TimeColumnIndex" is displayed as text.
TimeColumnIndex (Page 1035)	RW	-	-	References a configured time column.

Properties	RT Professional	RT Advanced	Panel RT	Description
TimeColumnLength (Page 1036)	RW	-	-	Specifies the width of the time column referenced with "TimeColumnIndex".
TimeColumnMeasurePoints (Page 1036)	RW	-	-	Specifies the number of measuring points to be displayed in the time column referenced with "TimeColumnIndex".
TimeColumnName (Page 1037)	RW	-	-	Specifies the name of a time column referenced with "TimeColumnIndex".
TimeColumnRangeType (Page 1037)	RW	-	-	Specifies the time range for a time column referenced with "TimeColumnIndex".
TimeColumnRemove (Page 1038)	RW	-	-	Removes a time column referenced with its name.
TimeColumnRename (Page 1038)	RW	-	-	Specifies the new name of a time column referenced with "TimeColumnIndex".
TimeColumnRepos (Page 1039)	RW	-	-	For multiple time columns with corresponding value columns, specifies the position of a time column referenced with "TimeColumnIndex".
TimeColumns	-	-	-	-
TimeColumnShowDate (Page 1040)	RW	-	-	Specifies whether the time column referenced with "TimeColumnIndex" is displayed with date and time.
TimeColumnShowIcon (Page 1041)	RW	-	-	Specifies whether the content of the time column referenced with "TimeColumnIndex" is displayed as an icon.
TimeColumnShowTitleIcon (Page 1041)	RW	-	-	Specifies whether the header of the time column referenced with "TimeColumnIndex" is displayed as an icon.
TimeColumnSort (Page 1042)	RW	-	-	Specifies how a time column referenced with "TimeColumnIndex" is sorted.
TimeColumnSortIndex (Page 1042)	RW	-	-	Specifies the sorting order of the time column referenced in "TimeColumnIndex".
TimeColumnTimeFormat (Page 1043)	RW	-	-	Specifies the time format for displaying the time column referenced with "TimeColumnIndex".
TimeColumnTimeRangeBase (Page 1043)	RW	-	-	Specifies the time unit for calculating the time range displayed in the time column referenced with "TimeColumnIndex".
TimeColumnTimeRangeFactor (Page 1044)	RW	-	-	Specifies the factor for calculating the time range.
TimeColumnUseValueColumnColors (Page 1045)	RW	-	-	Specifies whether the time column referenced with "TimeColumnIndex" is displayed in the colors of the value column.
TimeColumnVisible (Page 1045)	RW	-	-	Specifies whether the time column referenced with "TimeColumnIndex" is displayed in the table view.
TimeStepBase (Page 1047)	RW	-	-	Specifies the time unit for the accuracy of the time stamp that is displayed in the table.
TimeStepFactor (Page 1048)	RW	-	-	Specifies the time stamp accuracy with the "TimeStepBase" time unit.
TitleColor (Page 1049)	RW	-	-	Specifies the background color of the table header.
TitleCut (Page 1050)	RW	-	-	Specifies whether the content of the fields in a title bar is to be shortened if the column is too narrow.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TitleDarkShadowColor (Page 1050)	RW	-	-	Specifies the color for the dark side of the 3D shading for column and row headers in the object table.
TitleForeColor (Page 1051)	RW	-	-	Specifies the text color for the table column and row headers of the object.
TitleGridLineColor (Page 1052)	RW	-	-	Specifies the color of the separation lines in the title bar of the table.
TitleLightShadowColor (Page 1053)	RW	-	-	Specifies the color for the light side of the 3D shading for table column and row headers of the object.
TitleSort (Page 1054)	RW	-	-	Specifies how sorting by column header is triggered.
TitleStyle (Page 1055)	RW	-	-	Specifies whether to use a shading color for the text in the column headers.
ToolBarAlignment (Page 1063)	RW	-	-	Specifies the position of the toolbar.
ToolBarBackColor (Page 1064)	RW	-	-	Specifies the background color of the toolbar.
ToolBarButtonActive (Page 1065)	RW	-	-	Specifies whether the function linked to the button is activated in runtime.
ToolBarButtonAdd (Page 1066)	RW	-	-	Creates a new user-defined button in the object's toolbar.
ToolBarButtonAuthori- zation (Page 1066)	RW	-	-	Specifies the authorization for the selected key function.
ToolBarButtonBe- ginGroup (Page 1067)	RW	-	-	Inserts a leading separator (vertical line) for the selected key function on the toolbar.
ToolBarButtonClick (Page 1068)	RW	-	-	Clicks a toolbar button.
ToolBarButtonCount (Page 1069)	RW	-	-	Specifies the number of configured buttons in the toolbar.
ToolBarButtonEnabled (Page 1070)	RW	-	-	Specifies whether a user-defined toolbar button can be operated.
ToolBarButtonHotKey (Page 1071)	RW	-	-	Specifies the hot key for the selected object button.
ToolBarButtonID (Page 1071)	RW	-	-	References a button with its ID.
ToolBarButtonIndex (Page 1072)	RW	-	-	References a button.
ToolBarButtonLocked (Page 1073)	RW	-	-	Specifies whether the locked, pressed state of a user-defined button referenced with "ToolBarButtonIndex" is displayed.
ToolBarButtonName (Page 1074)	RW	-	-	Specifies the name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRemove (Page 1075)	RW	-	-	Removes a user-defined button referenced with its name.
ToolBarButtonRename (Page 1076)	RW	-	-	Specifies the new name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRepos (Page 1077)	RW	-	-	Specifies the position of a button referenced with "ToolBarButtonIndex" in the toolbar of the object.
ToolBarButtons	-	-	-	-

Properties	RT Professional	RT Advanced	Panel RT	Description
ToolBarButtonTooltipText (Page 1078)	RW	-	-	Specifies the tooltip text for the user-defined button in the toolbar.
ToolBarButtonUserDefined (Page 1079)	RW	-	-	Specifies whether the project engineer has added the toolbar button as new user-defined button.
ToolBarButtonVisible (Page 1080)	RW	-	-	Specifies whether the button referenced with "ToolBarButtonIndex" is displayed in the toolbar.
ToolBarShowTooltips (Page 1081)	RW	-	-	Specifies whether the tooltips for the key functions are displayed in runtime.
ToolBarUseBackColor (Page 1082)	RW	-	-	Specifies whether the background color for the toolbar is shown.
ToolBarUseHotKeys (Page 1083)	RW	-	-	Specifies whether the hotkeys are activated for buttons in the toolbar.
ToolBarVisible (Page 1084)	RW	-	-	Specifies whether the toolbar of the control is displayed.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
UseColumnBackColor (Page 1150)	RW	-	-	Specifies the settings for the background color of the columns.
UseColumnForeColor (Page 1150)	RW	-	-	Specifies the settings for the font color of the columns.
UseSelectedTitleColor (Page 1159)	RW	-	-	Specifies whether a selection color is used for the headers of selected table cells.
UseTableColor2 (Page 1161)	RW	-	-	Specifies whether a second row color is used for the representation of the table.
ValueColumnAdd (Page 1187)	RW	-	-	Creates a new value column.
ValueColumnAlignment (Page 1187)	RW	-	-	Specifies how the text in the value column referenced with "ValueColumnIndex" is aligned.
ValueColumnAutoPrecisions (Page 1188)	RW	-	-	Specifies whether the number of decimal places for a value axis referenced with "ValueColumnIndex" is set automatically.
ValueColumnBackColor (Page 1189)	RW	-	-	Specifies the background color of the value column referenced with "ValueColumnIndex".
ValueColumnCaption (Page 1189)	RW	-	-	Specifies the name of the value column referenced with "ValueColumnIndex".
ValueColumnCount (Page 1190)	RW	-	-	Specifies the number of configured value columns.
ValueColumnExponentialFormat (Page 1190)	RW	-	-	Specifies whether the values of a value column referenced with "ValueColumnIndex" are displayed in exponential notation.
ValueColumnForeColor (Page 1191)	RW	-	-	Specifies the font color of the value column referenced with "ValueColumnIndex".
ValueColumnHideText (Page 1192)	RW	-	-	Specifies whether the text in the value column referenced with "ValueColumnIndex" is hidden.
ValueColumnHideTitleText (Page 1192)	RW	-	-	Specifies whether the text in the header of the value column referenced with "ValueColumnIndex" is hidden.
ValueColumnIndex (Page 1193)	RW	-	-	References a value column.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

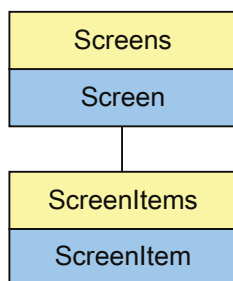
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ValueColumnLength (Page 1193)	RW	-	-	Specifies the width in characters of the value column referenced with "ValueColumnIndex".
ValueColumnName (Page 1194)	RW	-	-	Specifies the name of a value column referenced with "ValueColumnIndex".
ValueColumnPreci- sions (Page 1194)	RW	-	-	Specifies the number of decimal places for values in the value column referenced with "ValueColumnIndex".
ValueColumnProvider (Page 1195)	RW	-	-	Specifies the data source for the value column referenced with "ValueColumnIndex".
ValueColumnProvi- derCLSID (Page 1196)	RW	-	-	Specifies the provider CLSID of the data for a value column referenced with "ValueColumnIndex".
ValueColumnRemove (Page 1196)	RW	-	-	Removes a value column referenced with its name.
ValueColumnRename (Page 1197)	RW	-	-	Specifies the new name of a value column referenced with "ValueColumnIndex".
ValueColumnRepos (Page 1197)	RW	-	-	Specifies the position of a value column referenced with "ValueColumnIndex".
ValueColumns	-	-	-	-
ValueColumnSelect- TagName (Page 1198)	RW	-	-	Specifies that the dialog for selecting the tag name of the data source for the value column referenced with "ValueColumnIndex" is displayed in Runtime initial.
ValueColumnShowIcon (Page 1199)	RW	-	-	Specifies whether an icon is displayed in the value column referenced with "ValueColumnIndex".
ValueColumnShowTi- tleIcon (Page 1199)	RW	-	-	Specifies whether an icon is displayed in the header of the value column referenced with "ValueColumnIndex".
ValueColumnSort (Page 1200)	RW	-	-	Specifies the type of sorting for the value column referenced with "ValueColumnIndex" .
ValueColumnSortIndex (Page 1200)	RW	-	-	Specifies the sorting order.
ValueColumnTagName (Page 1201)	RW	-	-	Specifies the names of the tags whose values are displayed in the value column referenced with "ValueColumnIndex".
ValueColumnTimeCol- umn (Page 1201)	RW	-	-	Specifies the corresponding time column.
ValueColumnVisible (Page 1202)	RW	-	-	Specifies whether the value column referenced with "ValueColumnIndex" is displayed in the object.
VerticalGridLines (Page 1205)	RW	-	-	Specifies whether vertical separation lines are displayed.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-59 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 1260)	Yes	-	-	Executes the "Connect backup" key function of the control.
CalculateStatistic (Page 1261)	Yes	-	-	Executes the "Calculate statistics" key function of the f(t) trend view and the table view.
CopyRows (Page 1261)	Yes	-	-	Executes the "Copy lines" key function of the control.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
DetachDB (Page 1266)	Yes	-	-	Executes the "Disconnect backup" key function of the control.
Edit (Page 1266)	Yes	-	-	Executes the "Edit" key function of the table view.
Export (Page 1267)	Yes	-	-	Executes the "Export log" or "Export data" key function of the control.
GetOperatorMessage (Page 1276)	Yes	-	-	Returns the operator message object of the alarm view designated by name or index as "ICCAxOperatorMessage" type.
GetOperatorMessageCollection	Yes	-	-	-
GetRow (Page 1278)	Yes	-	-	Returns the row object designated by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 1280)	Yes	-	-	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetSelectedRow (Page 1286)	Yes	-	-	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 1287)	Yes	-	-	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatusbarElement (Page 1293)	Yes	-	-	Returns the element of the control status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 1294)	Yes	-	-	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetTimeColumn (Page 1299)	Yes	-	-	Returns the time column object of the table view designated by name or index as "ICCAxTimeColumn" type.
GetTimeColumnCollection (Page 1300)	Yes	-	-	Returns the list of all time column objects of the table view as "ICCAxCollection" type.
GetToolBarButton (Page 1301)	Yes	-	-	Returns the control toolbar button function designated by name or index as "ICCAxToolBarButton" type.
GetToolBarButtonCollection (Page 1302)	Yes	-	-	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
GetValueColumn (Page 1311)	Yes	-	-	Returns the value column object of the f(t) trend view designated by name or index as "ICCAxValueColumn" type.
GetValueColumnCollection (Page 1312)	Yes	-	-	Returns the list of all value column objects of the f(t) trend view as "ICCAxCollection" type.
MoveToFirst (Page 1322)	Yes	-	-	Executes the "First line" button function of the control.
MoveToLast (Page 1324)	Yes	-	-	Executes the "Last data record" button function of the control.
MoveToNext (Page 1325)	Yes	-	-	Executes the "Next data record" button function of the control.
MoveToPrevious (Page 1327)	Yes	-	-	Executes the "Previous data record" button function of the control.
NextColumn (Page 1328)	Yes	-	-	Executes the "Next column" button function of the table view.
PreviousColumn (Page 1330)	Yes	-	-	Executes the "Previous column" button function of the table view.
Print (Page 1331)	Yes	-	-	Executes the "Print" button function of the control.
SelectAll (Page 1346)	Yes	-	-	Selects all rows in a table-based control.
SelectRow (Page 1347)	Yes	-	-	Selects a specific row in a table-based control.
SelectStatisticArea (Page 1346)	Yes	-	-	Executes the "Set statistics range" button function of the table view.
ShowColumnSelection (Page 1350)	Yes	-	-	Executes the "Select columns" button function of the table view.
ShowHelp (Page 1352)	Yes	-	-	Executes the "Help" button function of the control.
ShowInsertValueDialog	Yes	-	-	-
ShowPropertyDialog (Page 1356)	Yes	-	-	Executes the "Configuration dialog" button function of the control.
ShowTagSelection (Page 1360)	Yes	-	-	Executes the "Select data connection" button function of the control.
ShowTimeSelection (Page 1361)	Yes	-	-	Executes the "Select time range" button function of the control.
StartStopUpdate (Page 1362)	Yes	-	-	Executes the "Start" or "Stop" button function of the control.
UnselectAll (Page 1365)	Yes	-	-	Removes all selections from the cells of a table-based control.
UnselectRow (Page 1365)	Yes	-	-	Removes the selections from a specific cell of a table-based control.

OnlineTrendControl (RT Professional)**Description**

Represents the "f(t) TrendView" object. The OnlineTrendControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIOnlineTrendControl

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-60 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettings- ForDesignMode	-	-	-	-
BackColor (Page 530)	RW	-	-	Specifies the background color.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
CanBeGrouped	-	-	-	-
Caption (Page 591)	RW	-	-	Specifies the text to be displayed in the header.
Closeable (Page 603)	RW	-	-	Specifies whether the object can be closed in run-time.
ConnectTrendWindows (Page 636)	RW	-	-	Specifies whether the configured trend views are connected.
ControlDesignMode (Page 638)	RW	-	-	Specifies the control design.
Enabled	-	-	-	-
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (Page 678)	RW	-	-	Specifies whether the directory for data export can be changed in runtime
ExportDirectoryname (Page 679)	RW	-	-	Specifies the target directory for the exported run-time data.
ExportFileExtension (Page 679)	RW	-	-	Specifies the file extension of the export file.
ExportFilename (Page 680)	RW	-	-	Specifies the name of the target file for the exported runtime data.
ExportFilename-Changeable (Page 681)	RW	-	-	Specifies whether the export file name can be changed in runtime.
ExportFormat	-	-	-	-
ExportFormatGuid (Page 682)	RW	-	-	Specifies the assignment of ID number and export provider.
ExportFormatName (Page 683)	RW	-	-	Specifies the export file format.
ExportParameters (Page 684)	RW	-	-	Specifies the parameters of the selected format by means of the properties dialog.
ExportSelection (Page 684)	RW	-	-	Specifies which runtime data of the control is exported.
ExportShowDialog (Page 685)	RW	-	-	Specifies whether the data export dialog is shown in runtime.
Font (Page 709)	RW	-	-	Specifies the font.
GraphDirection (Page 721)	RW	-	-	Specifies at which border of the trend view the current values are displayed.
Height (Page 724)	RW	-	-	Specifies the height.
Layer	-	-	-	-
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineColor (Page 790)	RW	-	-	Specifies the color of the window separation lines.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
LoadDataImmediately (Page 796)	RW	-	-	Specifies whether the tag values are downloaded from the logs for the time range to be displayed when you call a screen.
Location	-	-	-	-
MaximumNumberOfTimeAxes	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
MaximumNumberOfValueAxes	-	-	-	-
MinimumNumberOfTimeAxes	-	-	-	-
MinimumNumberOfValueAxes	-	-	-	-
Moveable (Page 829)	RW	-	-	Specifies whether the object can be moved in runtime.
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
Online (Page 843)	RW	-	-	Specifies start and stop of the updating.
PercentageAxis (Page 866)	RW	-	-	Specifies that an additional axis with percentage scaling is shown in the trend view.
PercentageAxisAlignment (Page 866)	RW	-	-	Specifies the alignment of the percentage axis.
PercentageAxisColor (Page 867)	RW	-	-	Specifies the font and line color of the percentage axis.
PrintJob (Page 876)	RW	-	-	Specifies a print job that is created in the "Reports" editor.
RTPersistence (Page 891)	RW	-	-	Specifies whether online configurations are retained after a screen change.
RTPersistenceAuthorization (Page 892)	RW	-	-	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 893)	RW	-	-	Specifies how online configurations of WinCC are retained.
ShowRuler (Page 942)	RW	-	-	Specifies whether a scale division (help line) is displayed for the axis label.
ShowRulerInAxis (Page 943)	RW	-	-	Specifies whether rulers are also displayed in the time axes.
ShowScrollbars (Page 944)	RW	-	-	Specifies whether scroll bars are displayed.
ShowStatisticRuler (Page 947)	RW	-	-	Specifies whether the lines defining the statistics range are displayed.
ShowTitle (Page 951)	RW	-	-	Specifies the style of the window border and window title for the object.
ShowTrendIcon (Page 953)	RW	-	-	Specifies whether an icon is displayed below the value axes.
Size	-	-	-	-
Sizeable (Page 956)	RW	-	-	Specifies that the size of an object can be changed in runtime.
StatusbarBackColor (Page 963)	RW	-	-	Specifies the background color for status bars.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
StatusbarElementAdd (Page 964)	RW	-	-	Creates a new, user-defined status bar element.
StatusbarElementAuto- Size (Page 965)	RW	-	-	Specifies whether the width of the selected status bar element is set automatically.
StatusbarElement- Count (Page 966)	RW	-	-	Specifies the number of configured status bar elements.
StatusbarElementIconId (Page 967)	RW	-	-	Specifies the ID number and symbol assignment of a status bar element.
StatusbarElementID (Page 967)	RW	-	-	References a status bar element with its element ID.
StatusbarElementIndex (Page 968)	RW	-	-	References a status bar element.
StatusbarElement- Name (Page 969)	RW	-	-	Specifies the name of a status bar element referenced with "StatusbarElementIndex".
StatusbarElementRe- move (Page 970)	RW	-	-	Removes the user-defined status bar element referenced with its name.
StatusbarElementRe- name (Page 971)	RW	-	-	Specifies the new name of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementRe- pos (Page 972)	RW	-	-	Specifies the position of an element referenced with "StatusbarElementIndex" in the status bar of the object.
StatusbarElements	-	-	-	-
StatusbarElementText (Page 973)	RW	-	-	Specifies the text of the selected status bar element.
StatusbarElementTool- tipText (Page 974)	RW	-	-	Specifies the tooltip text for the selected user-defined status bar element.
StatusbarElementUser- Defined (Page 974)	RW	-	-	Specifies whether the configuration engineer has added the status bar element as a new user-defined element.
StatusbarElementVisi- ble (Page 975)	RW	-	-	Specifies whether the status bar element referenced with "StatusbarElementIndex" is displayed in the object.
StatusbarElementWidth (Page 976)	RW	-	-	Specifies the width of the referenced status bar element in pixels.
StatusbarFont (Page 977)	RW	-	-	Specifies the font of the texts in the status bar.
StatusbarFontColor (Page 978)	RW	-	-	Specifies the color of the texts in the status bar.
StatusbarShowTooltips (Page 979)	RW	-	-	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 980)	RW	-	-	Specifies the default text in the status bar.
StatusbarUseBackCol- or (Page 980)	RW	-	-	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 981)	RW	-	-	Specifies whether the status bar of the control is displayed.
TabIndex	-	-	-	-

Properties	RT Professional	RT Advanced	Panel RT	Description
TabIndexAlpha	-	-	-	-
TimeAxes	-	-	-	-
TimeAxisAdd (Page 1013)	RW	-	-	Creates a new time axis. The newly created time axis is automatically referenced with "TimeAxisIndex".
TimeAxisAlignment (Page 1013)	RW	-	-	Specifies the alignment of the time axis.
TimeAxisBeginTime (Page 1014)	RW	-	-	Specifies the start time for the display of the selected trend.
TimeAxisColor (Page 1015)	RW	-	-	Specifies the color of the time axis.
TimeAxisCount (Page 1015)	RW	-	-	Specifies the number of configured time axes.
TimeAxisDateFormat (Page 1016)	RW	-	-	Defines the date format for displaying the selected time axis.
TimeAxisEndTime (Page 1017)	RW	-	-	Specifies the end time for the display of the selected trend.
TimeAxisIndex (Page 1017)	RW	-	-	References a time axis.
TimeAxisInTrendColor (Page 1018)	RW	-	-	Specifies whether the color of the axis referenced with "TimeAxisIndex" corresponds to the trend color.
TimeAxisLabel (Page 1018)	RW	-	-	Specifies the label text for a time axis referenced with "TimeAxisIndex".
TimeAxisMeasure-Points (Page 1019)	RW	-	-	Specifies the number of measurement points to be displayed for the selected time axis.
TimeAxisName (Page 1020)	RW	-	-	Specifies the name of a time axis referenced with "TimeAxisIndex".
TimeAxisOnline (Page 1020)	RW	-	-	Specifies that the time axis is updated continuously.
TimeAxisRangeType (Page 1021)	RW	-	-	Defines the time range setting for the selected time axis.
TimeAxisRemove (Page 1021)	RW	-	-	Removes a time axis referenced with its name.
TimeAxisRename (Page 1022)	RW	-	-	Specifies the new name of a time axis referenced with "TimeAxisIndex".
TimeAxisRepos (Page 1023)	RW	-	-	Specifies the position of a time axis referenced with "TimeAxisIndex" in the trend diagram of the object.
TimeAxisShowDate (Page 1023)	RW	-	-	Specifies whether the selected time axis is shown with date and time.
TimeAxisTimeFormat (Page 1024)	RW	-	-	Specifies the format of the information along the time axis for the selected trend.
TimeAxisTimeRange-Base (Page 1025)	RW	-	-	Specifies the time unit that defines the time range together with the time factor "TimeAxisTimeRange-Factor".
TimeAxisTimeRange-Factor (Page 1025)	RW	-	-	Specifies the time factor that defines the time period together with the time unit "TimeAxisTimeRange-Base".

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TimeAxisTrendWindow (Page 1026)	RW	-	-	Specifies the trend diagram in which the axis referenced with "TimeAxisIndex" is displayed.
TimeAxisVisible (Page 1026)	RW	-	-	Specifies whether the time axis referenced with "TimeAxisIndex" is displayed in the object.
TimeBase (Page 1027)	RW	-	-	Specifies the time zone in which the time values are displayed.
ToolBarAlignment (Page 1063)	RW	-	-	Specifies the position of the toolbar.
ToolBarBackColor (Page 1064)	RW	-	-	Specifies the background color of the toolbar.
ToolBarButtonActive (Page 1065)	RW	-	-	Specifies whether the function linked to the button is activated in runtime.
ToolBarButtonAdd (Page 1066)	RW	-	-	Creates a new user-defined button in the object's toolbar.
ToolBarButtonAuthori- zation (Page 1066)	RW	-	-	Specifies the authorization for the selected key function.
ToolBarButtonBe- ginGroup (Page 1067)	RW	-	-	Inserts a leading separator (vertical line) for the selected key function on the toolbar.
ToolBarButtonClick (Page 1068)	RW	-	-	Clicks a toolbar button.
ToolBarButtonCount (Page 1069)	RW	-	-	Specifies the number of configured buttons in the toolbar.
ToolBarButtonEnabled (Page 1070)	RW	-	-	Specifies whether a user-defined toolbar button can be operated.
ToolBarButtonHotKey (Page 1071)	RW	-	-	Specifies the shortcut for the selected button of the object.
ToolBarButtonID (Page 1071)	RW	-	-	References a button with its ID.
ToolBarButtonIndex (Page 1072)	RW	-	-	References a button.
ToolBarButtonLocked (Page 1073)	RW	-	-	Specifies whether the locked, pressed state of a user-defined button referenced with "ToolBarButtonIndex" is displayed.
ToolBarButtonName (Page 1074)	RW	-	-	Specifies the name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRemove (Page 1075)	RW	-	-	Removes a user-defined button referenced with its name.
ToolBarButtonRename (Page 1076)	RW	-	-	Specifies the new name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRepos (Page 1077)	RW	-	-	Specifies the position of a button referenced with "ToolBarButtonIndex" in the toolbar of the object.
ToolBarButtons	-	-	-	-
ToolBarButtonTooltip- Text (Page 1078)	RW	-	-	Specifies the tooltip text for the user-defined button in the toolbar.
ToolBarButtonUserDe- fined (Page 1079)	RW	-	-	Specifies whether the project engineer has added the toolbar button as new user-defined button.

Properties	RT Professional	RT Advanced	Panel RT	Description
ToolBarButtonVisible (Page 1080)	RW	-	-	Specifies whether the button referenced with "ToolBarButtonIndex" is displayed in the toolbar.
ToolBarShowTooltips (Page 1081)	RW	-	-	Specifies whether the tooltips for the key functions are displayed in runtime.
ToolBarUseBackColor (Page 1082)	RW	-	-	Specifies whether the background color for the toolbar is shown.
ToolBarUseHotKeys (Page 1083)	RW	-	-	Specifies whether the hotkeys are activated for buttons in the toolbar.
ToolBarVisible (Page 1084)	RW	-	-	Specifies whether the toolbar of the control is displayed.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
TrendAdd (Page 1095)	RW	-	-	Creates a new trend.
TrendAutoRangeBegin-TagName (Page 1096)	RW	-	-	Specifies a tag that defines the start value of the trend data area.
TrendAutoRangeBegin-Value (Page 1096)	RW	-	-	Specifies the start value of the trend data area.
TrendAutoRangeEnd-TagName (Page 1097)	RW	-	-	Specifies a tag that defines the end value of the trend data area.
TrendAutoRangeEnd-Value (Page 1098)	RW	-	-	Specifies the end value of the trend data area.
TrendAutoRange-Source (Page 1098)	RW	-	-	Specifies how the automatic data range of the trend data is determined.
TrendColor (Page 1099)	RW	-	-	Specifies the color of a referenced trend in the trend view.
TrendCount (Page 1100)	RW	-	-	Specifies the number of configured trends.
TrendExtendedColor-Set (Page 1101)	RW	-	-	Specifies whether the point and fill colors of the trend are shown.
TrendFill (Page 1102)	RW	-	-	Specifies whether the area below the trend is shown filled.
TrendFillColor (Page 1103)	RW	-	-	Defines the fill color of the trend.
TrendIndex (Page 1104)	RW	-	-	References a configured trend.
TrendLabel (Page 1105)	RW	-	-	Defines the label text for a trend referenced with "TrendIndex".
TrendLineStyle (Page 1106)	RW	-	-	Specifies which line type is used for display of the trend.
TrendLineType (Page 1106)	RW	-	-	Specifies how the trend is displayed.
TrendLineWidth (Page 1107)	RW	-	-	Specifies the line thickness of the selected trend in pixels.
TrendLowerLimit (Page 1108)	RW	-	-	Specifies the low limit of a tag that is displayed in the object as a trend.
TrendLowerLimitColor (Page 1108)	RW	-	-	Specifies the color that marks trend values which are below the value of "TrendLowerLimit".

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TrendLowerLimitColoring (Page 1109)	RW	-	-	Specifies whether "TrendLowerLimitColor" is used to identify tag values which are less than the value of "TrendLowerLimit".
TrendName (Page 1110)	RW	-	-	Specifies the name of a trend referenced with "TrendIndex".
TrendPointColor (Page 1111)	RW	-	-	Specifies the color of the dots on the reference trend.
TrendPointStyle (Page 1112)	RW	-	-	Specifies how the trend points are displayed.
TrendPointWidth (Page 1112)	RW	-	-	Specifies the point width in pixels.
TrendProvider (Page 1113)	RW	-	-	Specifies the data source for a selected trend.
TrendProviderCLSID (Page 1114)	RW	-	-	Specifies the provider CLSID of the data for a trend.
TrendRemove (Page 1115)	RW	-	-	Removes a trend referenced with its name.
TrendRename (Page 1116)	RW	-	-	Specifies the new name of a trend referenced with "TrendIndex".
TrendRepos (Page 1116)	RW	-	-	Specifies the position of a trend referenced with "TrendIndex" in the trend window of the object.
Trends	-	-	-	-
TrendSelectTagName (Page 1117)	RW	-	-	Specifies that the dialog for selecting the tag name for the data source of the value axis is displayed initially in runtime.
TrendTagName (Page 1119)	RW	-	-	Specifies the name of the tag that supplies the value axis with data.
TrendTimeAxis (Page 1121)	RW	-	-	Specifies the time axis to be used for the selected trend.
TrendTrendWindow (Page 1122)	RW	-	-	Specifies the trend window in which the selected trend is shown.
TrendUncertainColor (Page 1123)	RW	-	-	Specifies the color for values of uncertain status.
TrendUncertainColoring (Page 1124)	RW	-	-	Specifies that values of uncertain status are to be highlighted.
TrendUpperLimit (Page 1124)	RW	-	-	Specifies the high limit of a tag that is displayed in the specified object as a trend.
TrendUpperLimitColor (Page 1125)	RW	-	-	Specifies the color that marks trend values which are above the value of "TrendUpperLimit".
TrendUpperLimitColoring (Page 1126)	RW	-	-	Specifies whether the selection frame is shown with the color defined by the system.
TrendValueAlign	-	-	-	-
TrendValueAxis (Page 1127)	RW	-	-	Specifies the value axis that is used for the trend selected.
TrendValueUnit (Page 1127)	RW	-	-	For the trend type "Display values", specifies a unit for the values that is then attached to the values to be displayed, e.g. "%" or "°C".

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TrendVisible (Page 1128)	RW	-	-	Specifies whether the trend referenced with ""TrendIndex" is displayed in the object.
TrendWindowAdd (Page 1128)	RW	-	-	Creates a new trend view.
TrendWindowCoarse- Grid (Page 1129)	RW	-	-	Specifies if the grid lines for the main scale are displayed.
TrendWindowCoarse- GridColor (Page 1130)	RW	-	-	Specifies the color of the main grid for the referenced diagram in the object.
TrendWindowCount (Page 1130)	RW	-	-	Specifies the number of configured trend diagrams in the trend view.
TrendWindowFineGrid (Page 1131)	RW	-	-	Specifies whether grid lines are displayed for the secondary scaling.
TrendWindowFineGrid- Color (Page 1132)	RW	-	-	Specifies the color of the secondary grid for the referenced diagram in the object.
TrendWindowFore- groundTrendGrid (Page 1132)	RW	-	-	Specifies whether only the grid lines for the foreground trend are displayed in the selected trend window.
TrendWindowGridIn- TrendColor (Page 1133)	RW	-	-	Specifies whether the grid lines for the main scaling are displayed in the trend color.
TrendWindowHorizon- talGrid (Page 1134)	RW	-	-	Specifies whether the horizontal grid lines are shown.
TrendWindowIndex (Page 1134)	RW	-	-	References a trend view. To access the properties of a trend view, you need to set "TrendWindowIndex".
TrendWindowName (Page 1135)	RW	-	-	Specifies the name of a trend view referenced with "TrendWindowIndex".
TrendWindowRemove (Page 1136)	RW	-	-	Removes a trend view referenced with its name.
TrendWindowRename (Page 1136)	RW	-	-	Specifies the new name of a trend view referenced with "TrendWindowIndex".
TrendWindowRepos (Page 1137)	RW	-	-	Specifies the position of a trend view referenced with "TrendWindowIndex".
TrendWindowRulerCol- or (Page 1137)	RW	-	-	Specifies the color of the ruler.
TrendWindowRulerLay- er (Page 1138)	RW	-	-	Specifies the display layer of the ruler in the trend view.
TrendWindowRuler- Style (Page 1139)	RW	-	-	Specifies the appearance of the ruler.
TrendWindowRuler- Width (Page 1140)	RW	-	-	Specifies the thickness of the ruler in pixels.
TrendWindows	-	-	-	-
TrendWindowSpace- Portion (Page 1140)	RW	-	-	Specifies the range proportion of the referenced diagram in the diagram area of the object.
TrendWindowStatisti- cRulerColor (Page 1141)	RW	-	-	Specifies the color of the statistics ruler.
TrendWindowStatisti- cRulerStyle (Page 1142)	RW	-	-	Defines the graphical configurability of the lines for specifying the statistics area in the trend window.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TrendWindowStatisticRulerWidth (Page 1143)	RW	-	-	Specifies the line thickness of the statistics ruler.
TrendWindowVerticalGrid (Page 1143)	RW	-	-	Specifies whether the vertical grid lines are displayed.
TrendWindowVisible (Page 1144)	RW	-	-	Specifies whether the trend view referenced with "TrendWindowIndex" is displayed.
UseTrendNameAsLabel (Page 1165)	RW	-	-	Specifies whether the "Name" or "Label" property is used as a designation for the trend in runtime.
ValueAxes	-	-	-	-
ValueAxisAdd (Page 1175)	RW	-	-	Creates a new value axis.
ValueAxisAlignment (Page 1175)	RW	-	-	Specifies the alignment of the value axis.
ValueAxisAutoPrecisions (Page 1176)	RW	-	-	Specifies whether the number of decimal places for a value axis referenced with "ValueAxisIndex" is set automatically.
ValueAxisAutoRange (Page 1177)	RW	-	-	Specifies whether the value range of a value axis referenced with "ValueAxisIndex" is calculated automatically.
ValueAxisBeginValue (Page 1177)	RW	-	-	Specifies the lower end of the value range of a value axis referenced with "ValueAxisIndex".
ValueAxisColor (Page 1178)	RW	-	-	Specifies the color of a value axis.
ValueAxisCount (Page 1179)	RW	-	-	Specifies the number of configured value axes.
ValueAxisEndValue (Page 1179)	RW	-	-	Specifies the upper end of the value range of a value axis referenced with "ValueAxisIndex".
ValueAxisExponentialFormat (Page 1180)	RW	-	-	Specifies whether the values of a value axis referenced with "ValueAxisIndex" are displayed in exponential notation.
ValueAxisIndex (Page 1180)	RW	-	-	References a value axis.
ValueAxisInTrendColor (Page 1181)	RW	-	-	Specifies whether the color of the axis referenced with "ValueAxisIndex" corresponds to the trend color.
ValueAxisLabel (Page 1182)	RW	-	-	Defines the label text for a value axis referenced with "ValueAxisIndex".
ValueAxisName (Page 1182)	RW	-	-	Specifies the name of a value axis referenced with "ValueAxisIndex".
ValueAxisPrecisions (Page 1183)	RW	-	-	Specifies the number of displayed decimal places.
ValueAxisRemove (Page 1183)	RW	-	-	Removes a value axis referenced with its name.
ValueAxisRename (Page 1184)	RW	-	-	Specifies the new name of a value axis referenced with "ValueAxisIndex".
ValueAxisRepos (Page 1184)	RW	-	-	Specifies the position of a value axis referenced with "ValueAxisIndex".

Properties	RT Professional	RT Advanced	Panel RT	Description
ValueAxisScalingType (Page 1185)	RW	-	-	Specifies the type of scaling for the value axis referenced with "ValueAxisIndex".
ValueAxisTrendWindow (Page 1186)	RW	-	-	Specifies the trend diagram in which the axis referenced with "ValueAxisIndex" is displayed.
ValueAxisVisible (Page 1186)	RW	-	-	Specifies whether the value axis referenced with "ValueAxisIndex" is displayed in the object.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-61 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
AttachDB (Page 1260)	Yes	-	-	Executes the "Connect backup" key function of the control.
CalculateStatistic (Page 1261)	Yes	-	-	Executes the "Calculate statistics" key function of the f(t) trend view and the table view.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
DetachDB (Page 1266)	Yes	-	-	Executes the "Disconnect backup" key function of the control.
Export (Page 1267)	Yes	-	-	Executes the "Export log" or "Export data" key function of the control.
GetRulerData (Page 1285)	Yes	-	-	Returns the value of the called trend at the ruler position.
GetStatusbarElement (Page 1293)	Yes	-	-	Returns the element of the control status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 1300)	Yes	-	-	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetTimeAxis (Page 1296)	Yes	-	-	Returns the f(t) trend view time axis object that is specified by name or index as "ICCAxTimeAxis" type.
GetTimeAxisCollection (Page 1297)	Yes	-	-	Returns a list of all f(t) trend view time axis objects as "ICCAxCollection" type.
GetToolBarButton (Page 1301)	Yes	-	-	Returns the control toolbar button function designated by name or index as "ICCAxToolBarButton" type.
GetToolBarButtonCollection (Page 1302)	Yes	-	-	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
GetTrend (Page 1304)	Yes	-	-	Returns the f(t) or f(x) trend view trend designated by name or index as "ICCAxTrend" or "ICCAxFuctionTrend" type.

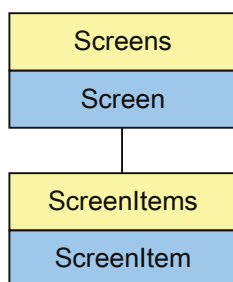
1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
GetTrendCollection (Page 1305)	Yes	-	-	Returns the list of all trends of the f(t) or f(x) trend view as "ICCAxCollection" type.
GetTrendWindow (Page 1306)	Yes	-	-	Returns the trend window object of the f(t) or f(x) trend view designated by name or index as "ICCAx-TrendWindow" type.
GetTrendWindowCol- lection (Page 1307)	Yes	-	-	Returns the list of all trend window objects of the f(t) or f(x) trend views as "ICCAxCollection" type.
GetValueAxis (Page 1308)	Yes	-	-	Returns the f(t) trend view value axis object that is specified by name or index as "ICCAxValueAxis" type.
GetValueAxisCollection (Page 1309)	Yes	-	-	Returns a list of all f(t) trend view value axis objects as "ICCAxCollection" type.
MoveAxis (Page 1322)	Yes	-	-	Executes the "Move axes area" button function of the f(t) and f(x) trend views.
MoveRuler	Yes	-	-	
MoveToFirst (Page 1322)	Yes	-	-	Executes the "First line" button function of the control.
MoveToLast (Page 1324)	Yes	-	-	Executes the "Last data record" button function of the control.
MoveToNext (Page 1325)	Yes	-	-	Executes the "Next data record" button function of the control.
MoveToPrevious (Page 1327)	Yes	-	-	Executes the "Previous data record" button function of the control.
NextTrend (Page 1329)	Yes	-	-	Executes the "Next trend" button function of the f(t) and f(x) trend views.
OneToOneView (Page 1329)	Yes	-	-	Executes the "Original view" button function of the f(t) and f(x) trend views.
PreviousTrend (Page 1331)	Yes	-	-	Executes the "Previous trend" button function of the f(t) and f(x) trend views.
Print (Page 1331)	Yes	-	-	Executes the "Print" button function of the control.
ShowHelp (Page 1352)	Yes	-	-	Executes the "Help" button function of the control.
ShowPercentageAxis (Page 1356)	Yes	-	-	Executes the "Relative axis" button function of the f(t) trend view.
ShowPropertyDialog (Page 1356)	Yes	-	-	Executes the "Configuration dialog" button function of the control.
ShowTagSelection (Page 1360)	Yes	-	-	Executes the "Select data connection" button function of the control.
ShowTimeSelection (Page 1361)	Yes	-	-	Executes the "Select time range" button function of the control.
ZoomArea (Page 1370)	Yes	-	-	Executes the "Zoom area" button function of the f(t) and f(x) trend views.
ZoomInOut (Page 1370)	Yes	-	-	Executes the "Zoom +/-" button function of the f(t) and f(x) trend views.
ZoomInOutTime (Page 1371)	Yes	-	-	Executes the "Zoom time axis +/-" button function of the f(t) trend view.

Methods	RT Professional	RT Advanced	Panel RT	Description
ZoomInOutValues (Page 1371)	Yes	-	-	Executes the "Zoom value axis +/-" button function of the f(t) trend view.
ZoomMove (Page 1373)	Yes	-	-	Executes the "Move trend area" button function of the f(t) and f(x) trend views.

OptionGroup (RT Professional)

Description



Represents the "Option button" object. The OptionGroup object is an element of the ScreenItems list.

Type identifier in VBS

HMIOptionGroup

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-62 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AdaptBorder (Page 502)	RW	-	-	Specifies whether the border of the object will be dynamically adapted to the text size.
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle	-	-	-	-
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CheckMarkAlignment (Page 601)	RW	-	-	Specifies whether the fields are right-aligned.
CheckMarkCount (Page 601)	RW	-	-	Specifies the number of fields.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.

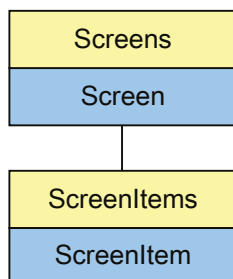
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text is shown underlined.
ForeColor (Page 716)	RW	-	-	Specifies the font color of the text.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalAlignment (Page 740)	RW	-	-	Specifies the horizontal alignment of the text.
Index (Page 745)	RW	-	-	Specifies the index for the selected text field.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
LogOperation (Page 798)	RW	-	-	Specifies whether, after operating this object, an alarm is output to the alarm system.
Name	-	-	-	-
ProcessValue (Page 877)	RW	-	-	Specifies the default for the value to be displayed.
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
ShowBadTagState (Page 930)	RW	-	-	Defines whether the object is grayed out when a bad quality code or tag status is detected.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Text (Page 1004)	RW	-	-	Specifies the label for the text field.
TextHandles	-	-	-	-
TextOrientation (Page 1007)	RW	-	-	Specifies the text orientation.
Texts	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
VerticalAlignment (Page 1204)	RW	-	-	Specifies the vertical alignment of the text in the specified object.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-63 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

PDFview (Panels, Comfort Panels, RT Advanced)**Description**

Represents the "PDF view" object. The PDFview object is an element of the ScreenItems list.

Type identifier in VBS

HMIPDFview

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-64 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
FileName	-	-	-	-

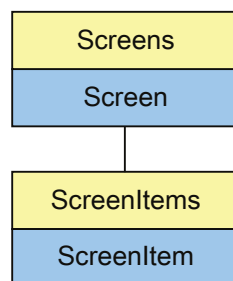
Properties	RT Professional	RT Advanced	Panel RT	Description
Height	-	-	-	Specifies the height.
Layer	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	RW	RW	Specifies the value of the Y coordinate.
Visible	-	-	-	-
Width	-	-	-	Specifies the width of the object in pixels.

Table 1-65 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	Yes	Yes	Enables the permanent area or the root screen.

PLCCodeViewer (RT Professional)

Description



Displays the "PLC Code Viewer" object. The PLCCodeViewer object is an element of the ScreenItems list.

Type identifier in VBS

HMIPLCCodeViewer

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-66 Properties

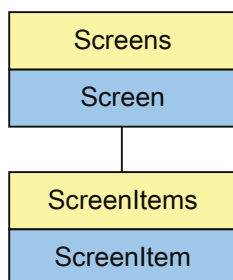
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
NavigateTo (Page 834)	RW	RW	RW	Specifies the module to be jumped to.
NumberOfVisibleLines (Page 840)	RW	-	-	Specifies the number of visible lines in the PLC Code Viewer.
PathHeaderFont (Page 864)	RW	-	-	Specifies the header font.
Size	-	-	-	-
SymbolTableFont (Page 987)	RW	-	-	Specifies the font of the symbol table in the PLC Code Viewer.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolbarButtons	-	-	-	-
Top (Page 1086)	-	RW	RW	Specifies the value of the Y coordinate.
TransitionHeaderFont (Page 1090)	RW	-	-	Specifies the font in the information area of this PLC Code Viewer.
Visible (Page 1208)	-	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-67 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Polygon (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Polygon" object. The Polygon object is an element of the ScreenItems list.

Type identifier in VBS

HMIPolygon

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-68 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
ActualPointIndex (Page 500)	RW	-	-	Specifies the number of the current corner point.
ActualPointLeft (Page 500)	RW	-	-	Specifies the X coordinate of the current corner point with reference to the screen origin.
ActualPointTop (Page 501)	RW	-	-	Specifies the Y coordinate of the current corner point with reference to the screen origin.
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	RW	RW	Specifies the fill pattern.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	RW	RW	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	RW	RW	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DeviceStyle	-	-	-	-
EdgeStyle (Page 661)	RW	RW	RW	Specifies the line style.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor	RW	-	-	-
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

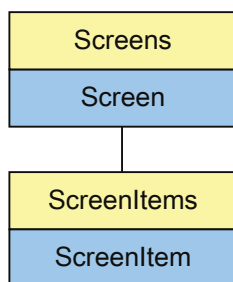
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
LineWidth	-	-	-	-
Location	-	-	-	-
Name	-	-	-	-
Points	-	-	-	-
PointsCount (Page 874)	RW	-	-	Specifies the number of corner points of the polyline or of the polygon.
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
RotationAngle (Page 886)	RW	-	-	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 887)	RW	-	-	Specifies the X coordinate of the pivot point for the object in runtime.
RotationCenterTop (Page 888)	RW	-	-	Specifies the Y coordinate of the pivot point for the object in runtime.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-69 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

Polyline (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Polyline" object. The Polyline object is an element of the ScreenItems list.

Type identifier in VBS

HMIPolyline

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-70 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
ActualPointIndex (Page 500)	RW	-	-	Specifies the number of the current corner point.
ActualPointLeft (Page 500)	RW	-	-	Specifies the X coordinate of the current corner point with reference to the screen origin.
ActualPointTop (Page 501)	RW	-	-	Specifies the Y coordinate of the current corner point with reference to the screen origin.
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BorderEndStyle (Page 569)	RW	-	-	Specifies the type of line ends.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	RW	RW	Specifies the line color.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DeviceStyle	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
EndStyle (Page 671)	RW	RW	RW	Specifies how the line end is displayed.
FillStyle (Page 691)	-	RW	RW	Specifies whether the background color is displayed with a broken line.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the object.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line end.
LineWidth (Page 794)	RW	RW	RW	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
Points	-	-	-	-

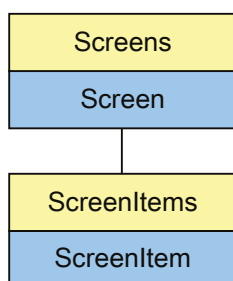
Properties	RT Professional	RT Advanced	Panel RT	Description
PointsCount (Page 874)	RW	-	-	Specifies the number of corner points of the polyline or of the polygon.
RotationAngle (Page 886)	RW	-	-	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 887)	RW	-	-	Specifies the X coordinate of the pivot point for the object in runtime.
RotationCenterTop (Page 888)	RW	-	-	Specifies the Y coordinate of the pivot point for the object in runtime.
Size	-	-	-	-
StartStyle (Page 961)	RW	RW	RW	Specifies how the line start is displayed.
Style (Page 985)	RW	RW	RW	Specifies the line style.
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the object.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	RW	RW	Specifies whether the object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-71 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

ProDiag overview (RT Professional)

Description



Represents the "ProDiagOverview" object. The ProDiagOverview object is an element of the ScreenItems list.

Type identifier in VBS

HMIProDiagOverview

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-72 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
Height (Page 724)	RW	RW	RW	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
Name	-	-	-	-
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.

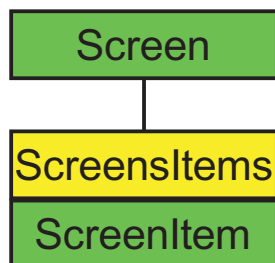
Properties	RT Professional	RT Advanced	Panel RT	Description
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	RW	RW	Specifies the width of the object in pixels.

Table 1-73 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

ProtectedAreaView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "EffectiveRangeName" (RFID)" object. The ProtectedAreaView object is an element of the ScreensItems list.

Type identifier in VBS

HMIProtectedAreaView

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-74 Properties

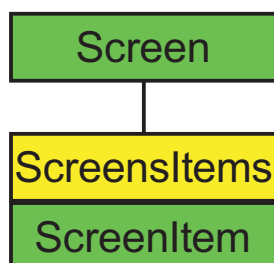
Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Font	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	-	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	-	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	-	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-75 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

RangeLabelView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "EffectiveRangeName" object. The RangeLabelView object is an element of the ScreenItems list.

Type identifier in VBS

HMIRangeLabelView

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-76 Properties

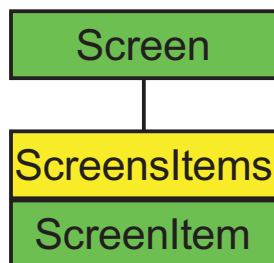
Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Font	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	-	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	-	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	-	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-77 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

RangeQualityView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "EffectiveRangeSignal" object. The RangeQualityView object is an element of the ScreensItems list.

Type identifier in VBS

HMIRangeQualityView

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-78 Properties

Properties	RT Profes-sional	RT Ad-vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	-	RW	Returns the layer in the screen in which an object is located.
Left (Page 774)	-	-	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	-	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

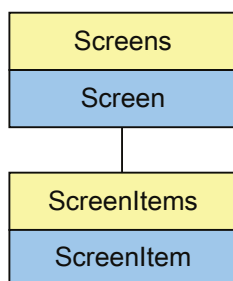
Table 1-79 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

See also

Height (Page 724)

Width (Page 1221)

RecipeView (Panels, Comfort Panels, RT Advanced)**Description**

Displays the "Recipe view" object. The RecipeView object is an element of the ScreenItems list.

If you change the settings for this object with a user-defined function, the changed settings are retained even after the screen is called again.

Note

The object "Simple RecipeView" cannot be dynamized with a user-defined function.

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-80 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AdvancedButtonPositions	-	-	-	-
AdvancedView	-	-	-	-
AllowEdit	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	-	RW	RW	Specifies the operating rights of the selected object in runtime.
BackButtonVisible	-	-	-	-
BackColor (Page 530)	-	RW	RW	Specifies the background color.
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
ColumnWidth	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ComboboxFont	-	-	-	-
CornerRadius	-	-	-	-
DataRecordNameCap- tion	-	-	-	-
DataRecordNrCaption	-	-	-	-
DeviceStyle	-	-	-	-
Display3D	-	-	-	-
DisplayButton2Plc	-	-	-	-
DisplayButtonCompari- son	-	-	-	-
DisplayButtonDelete	-	-	-	-
DisplayButtonFromPlc	-	-	-	-
DisplayButtonHelp	-	-	-	-
DisplayButtonNew	-	-	-	-
DisplayButtonSave	-	-	-	-
DisplayButtonSaveAs	-	-	-	-
DisplayComboBox	-	-	-	-
DisplayGridLines	-	-	-	-
DisplayLabeling	-	-	-	-
DisplayNumbers	-	-	-	-
DisplaySize	-	-	-	-
DisplayStatusBar	-	-	-	-
DisplayTable	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be oper- ated in runtime.
EntryNameCaption	-	-	-	-
EntryNameColumn- Width	-	-	-	-
EntryValueColFirst	-	-	-	-
EntryValueColumn- Width	-	-	-	-
EntryValueFieldLength	-	-	-	-
EntryValuePos	-	-	-	-
ES2RT_ButtonPositions	-	-	-	-
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
ES2RT_EntryNameCo- lumnWidth	-	-	-	-
ES2RT_EntryValueCo- lumnWidth	-	-	-	-
ES2RT_ListAreaHeight	-	-	-	-
ES2RT_ListAreaWidth	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (Page 707)	-	RW	RW	Specifies the color of the focus border when the object has the focus.
FocusWidth (Page 708)	-	RW	RW	Specifies the border width when the object has the focus.
Font	-	-	-	-
ForeColor (Page 716)	-	RW	RW	Specifies the font color of the text.
HeaderFont	-	-	-	-
Height	-	-	-	-
IsRunningUnderCE	-	-	-	-
IsVerticalScrollBarEnabled	-	-	-	-
KeyboardOnline	-	-	-	-
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
ListAreaHeight	-	-	-	-
ListAreaLeft	-	-	-	-
ListAreaTop	-	-	-	-
ListAreaWidth	-	-	-	-
Location	-	-	-	-
MaxNumberOfComboBoxCharacters	-	-	-	-
MenuButtonVisible	-	-	-	-
Name	-	-	-	-
NameColumnWidth	-	-	-	-
NumberOfButtons	-	-	-	-
NumberOfLines	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
PlcUDTFilter	-	-	-	-
ProhibitDataRecordTagInOnlySimpleView	-	-	-	-
Recipe	-	-	-	-
RecipeName (Page 882)	-	RW	RW	Specifies the name of the recipe that is displayed in the "Recipe view".
RecipeNameCaption	-	-	-	-
RecipeNrCaption	-	-	-	-
RecipeNrColFirst	-	-	-	-
RecipeNumber (Page 883)	-	RW	RW	Specifies the number of the recipe that is displayed in the "Recipe view".

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
RecordName (Page 883)	-	RW	RW	Specifies the name of the recipe data record that is displayed in the "Recipe view".
RecordNrColFirst	-	-	-	-
RecordNumber (Page 884)	-	R	R	Returns the number of the recipe data record that is currently being displayed in the "Recipe view".
RenameButtonVisible	-	-	-	-
SelectionBackColor (Page 917)	-	R	R	Returns the background color of the selected cells.
SelectionForeColor (Page 919)	-	R	R	Returns the foreground color of the selected cells.
SimpleView	-	-	-	-
Size	-	-	-	-
StatuslineFont	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (Page 991)	-	R	R	Returns the background color of the table cells.
TableEvenRowBack- Color	-	-	-	-
TableForeColor (Page 994)	-	R	R	Returns the font color used in the table cells of the object.
TableGridLineColor (Page 995)	-	R	R	Returns the color of the grid lines in the table.
TableHeaderBackColor (Page 996)	-	R	R	Returns the background color in the header of the table.
TableHeaderBackFill- Style	-	-	-	-
TableHeaderBorder- BackColor	-	-	-	-
TableHeaderBorderCol- or	-	-	-	-
TableHeaderBorder- Width	-	-	-	-
TableHeaderCornerRa- dius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGra- dientColor	-	-	-	-
TableHeaderFirstGra- dientOffset	-	-	-	-
TableHeaderForeColor (Page 998)	-	R	R	Returns the text color in the header of the table.
TableHeaderMiddleGra- dientColor	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TableHeaderPadding- Bottom	-	-	-	-
TableHeaderPadding- Left	-	-	-	-
TableHeaderPadding- Right	-	-	-	-
TableHeaderPadding- Top	-	-	-	-
TableHeaderSecond- GradientColor	-	-	-	-
TableHeaderSecond- GradientOffset	-	-	-	-
Tag4DataRecord	-	-	-	-
Tag4RecipeNumber	-	-	-	-
TextualObjectPositions	-	-	-	-
TextualObjectsAutoSize	-	-	-	-
TextualObjectsBorder- BackColor	-	-	-	-
TextualObjectsBorder- Color	-	-	-	-
TextualObjectsBorder- Width	-	-	-	-
TextualObjectsCorner- Radius	-	-	-	-
TextualObjectsEdgeS- tyle	-	-	-	-
TextualObjectsPadding- Bottom	-	-	-	-
TextualObjectsPaddin- gLeft	-	-	-	-
TextualObjectsPadding- Right	-	-	-	-
TextualObjectsPadding- Top	-	-	-	-
Top (Page 1086)	-	R	R	Returns the value of the Y coordinate.
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGra- dient	-	-	-	-
UseDesignColorSche- ma	-	-	-	-
UseTableHeaderFirst- Gradient	-	-	-	-
UseTableHeaderSe- condGradient	-	-	-	-
ValueCaption	-	-	-	-
ValueColumnWidth	-	-	-	-

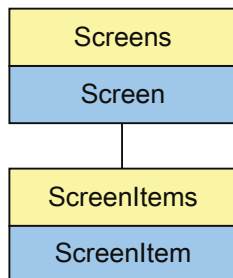
Properties	RT Professional	RT Advanced	Panel RT	Description
VerticalScrolling	-	-	-	-
ViewType	-	-	-	-
ViewTypeForSaveStream	-	-	-	-
Visible (Page 1208)	-	RW	RW	Specifies whether the selected object is visible.
VisibleItems	-	-	-	-
Width	-	-	-	-

Table 1-81 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	Yes	Yes	Enables the permanent area or the root screen.

Rectangle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "Rectangle" object. The Rectangle object is an element of the ScreenItems list.

Type identifier in VBS

HMIRectangle

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-82 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	RW	RW	RW	Specifies the background color of the selected object.
BackFillStyle (Page 535)	RW	RW	RW	Specifies the fill pattern of the selected object.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line for the selected object.
BorderColor (Page 566)	RW	RW	RW	Specifies the line color of the selected object.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines for the selected object.
BorderWidth (Page 581)	RW	RW	RW	Specifies the line thickness of the selected object.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the specified object.
DeviceStyle	-	-	-	-
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	RW	RW	Specifies the line style of the selected object.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern for the selected object.

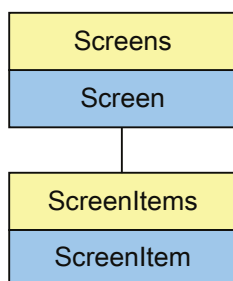
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line for the selected object.
Height (Page 724)	RW	RW	RW	Specifies the height of the selected object.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the selected object.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Name	-	-	-	Returns the object name as STRING.
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
RoundCornerHeight (Page 888)	RW	-	-	Specifies the corner radius.
RoundCornerWidth (Page 889)	RW	-	-	Specifies the corner radius.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	RW	RW	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate of the selected object.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSet- tings (Page 1153)	RW	-	-	Specifies whether the object is displayed with the shading defined in the active design.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	RW	RW	Specifies the width of the object in pixels.

Table 1-83 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

RoundButton (RT Professional)

Description



Represents the "Round button" object. The RoundButton object is an element of the ScreenItems list.

Type identifier in VBS

HMIRoundButton

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-84 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	RW	-	-	Specifies the background color of the selected object.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern of the selected object.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background of the selected object flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background for the selected object.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line for the selected object.
BorderBrightColor3D (Page 565)	RW	-	-	Specifies the frame color of the following frame parts in case of a 3D display of the selected object: Outer border parts top and bottom; inner border parts bottom and right
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line for the selected object.
BorderShadeColor3D	-	-	-	-
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines for the selected object.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness of the selected object.
BorderWidth3D (Page 584)	RW	-	-	Specifies the width of the border for 3D display of the selected object.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the specified object.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style of the selected object.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern for the selected object.
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line for the selected object.
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text of the selected object is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text of the selected object is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font of the selected object.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text for the selected object.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text of the selected object is underlined.
ForeColor (Page 716)	R	-	-	Specifies the font color of the text for the selected object.
Height (Page 724)	R	-	-	Specifies the height of the selected object.
HorizontalAlignment (Page 740)	R	-	-	Specifies the horizontal alignment of the text within the selected object.
Layer (Page 768)	R	-	-	Returns the layer in the screen in which an object is located.
Left (Page 774)	R	-	-	Returns the value of the X coordinate of the object.
LineEndShapeStyle (Page 791)	R	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
Mode (Page 828)	R	-	-	Returns the field type of the specified object.
Name	-	-	-	-
PictureAlignment (Page 868)	R	-	-	Returns the display type for the background screen in the process image.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
PictureDeactivated (Page 869)	R	-	-	Returns the graphic that is displayed in the "Disabled" state.
PictureOff (Page 870)	R	-	-	Returns the graphic that is displayed in the "Off" state.
PictureOn (Page 871)	R	-	-	Returns the screen that is displayed in the "On" state.
Pressed (Page 876)	R	-	-	Returns whether the specified object is displayed as pressed.
Radius (Page 880)	R	-	-	Returns the radius of the specified "Circle" object.
RelativeFillLevel (Page 884)	R	-	-	Returns the percentage to which the object is filled.
ShowFillLevel (Page 937)	R	-	-	Returns whether the specified object is filled.
Size	-	-	-	-
StyleSettings (Page 986)	R	-	-	Returns the style in which the object is displayed.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (Page 1004)	R	-	-	Returns the label for the text field.
TextOrientation (Page 1007)	R	-	-	Returns the text orientation of the specified object.
Toggle (Page 1055)	R	-	-	Returns whether the specified object engages after it has been operated in Runtime.
ToolTipText (Page 1085)	R	-	-	Returns the tooltip text.
Top (Page 1086)	R	-	-	Returns the value of the Y coordinate of the object.
Transparency (Page 1091)	R	-	-	Returns the object transparency in percent.
TransparentColorDeactivatedPicture (Page 1093)	R	-	-	Returns the color of the assigned bitmap object to be set to "transparent" for the "Disabled" state.
TransparentColorPictureOff (Page 1093)	R	-	-	Returns the color of the assigned bitmap object to be set to "transparent" for the "Off" state.
TransparentColorPictureOn (Page 1094)	R	-	-	Returns the color of the assigned bitmap object to be set to "transparent" for the "On" state.
UseDesignColorScheme (Page 1152)	R	-	-	Returns whether the colors defined in the global color scheme of the current design are used for this object.
UseDesignShadowSettings (Page 1153)	R	-	-	Returns whether the object is displayed with the shading defined in the active design.
UseTransparentColorDeactivatedPicture (Page 1164)	R	-	-	Returns whether the transparent color specified with the "TransparentColorDeactivatedPicture" property is used for the "Disabled" state.
UseTransparentColorPictureOff (Page 1164)	R	-	-	Returns whether the transparent color defined with the property "TransparentColorPictureOff" is used for the "Off" state.

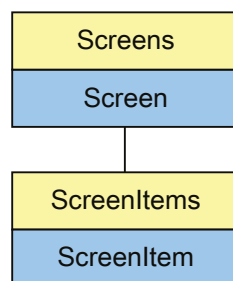
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
UseTransparentColorPictureOn (Page 1165)	R	-	-	Returns whether the transparent color defined with the property "TransparentColorPictureOn" is used for the "On" state.
VerticalAlignment (Page 1204)	R	-	-	Returns the vertical alignment of the text of the specified object.
Visible (Page 1208)	R	-	-	Returns whether the specified object is visible.
Width (Page 1221)	R	-	-	Returns the width of the object in pixels.

Table 1-85 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

S7GraphOverview (RT Professional)

Description



Represents the "GRAPH overview" object. The S7GraphOverview object is an element of the ScreenItems list.

Type identifier in VBS

HMIS7GraphOverview

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-86 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AssociatedS7Graph-DBName	-	-	-	-
AssociatedS7Graph-DBTag	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
ErrorColor (Page 673)	RW	-	-	Specifies the color of errors in the GRAPH overview.
Errorflag (Page 675)	RW	-	-	Specifies whether an error description is displayed in the device / detail view.
Height (Page 724)	-	-	-	-
HighlightColor (Page 729)	RW	-	-	Specifies the highlight color in the GRAPH overview.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
PathHeaderBackColor (Page 864)	-	-	-	-
PathHeaderFont (Page 864)	RW	-	-	Specifies the header font.
PathHeaderTextColor (Page 865)	RW	-	-	Specifies the font color in the header of the GRAPH overview.
SeparatorColor (Page 924)	RW	-	-	Specifies the color of the separation lines in the selection list.
Size	-	-	-	-
StepBackColor (Page 982)	RW	-	-	Specifies the background color of the step.
StepFont (Page 983)	RW	-	-	Specifies the font color of the step.

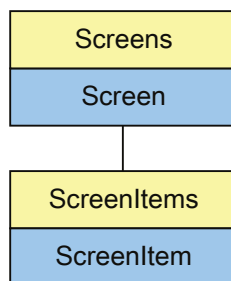
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
StepTextColor (Page 984)	RW	-	-	Specifies the text color of the step.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	RW	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	-	-	-	-

Table 1-87 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

ScreenWindow (RT Professional)

Description



Represents the "ScreenWindow" object. The ScreenWindow object is an element of the ScreenItems list.

Type identifier in VBS

HMIScreenWindow

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

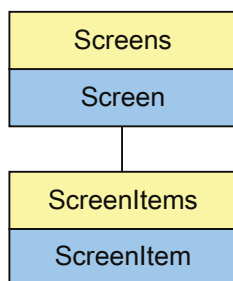
Properties	RT Professional	RT Advanced	Panel RT	Description
AdaptScreenToWindow (Page 503)	R	-	-	Specifies whether or not the screen visualized in a screen window is adapted to fit the size of the screen window in runtime.
AdaptWindowtoScreen (Page 504)	R	-	-	Specifies whether the screen window is adapted to fit the screen it visualizes in runtime.
AllTagTypesAllowed	-	-	-	-
BorderEnabled (Page 569)	R	-	-	Specifies whether the window is displayed with a border in runtime.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionText (Page 594)	RW	-	-	Specifies the text to be displayed in the header.
Flashing	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalScrollBarPosition (Page 742)	RW	-	-	Specifies the position of the slider in the horizontal scroll bar.
IndipendentWindow	-	-	-	-
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LeftOffset (Page 779)	RW	-	-	Sets the horizontal shift of the zero point of the display of a screen that is larger than the screen window.
Location	-	-	-	-
MenuToolBarConfig (Page 804)	RW	-	-	Determines the configuration file with user-defined menu and toolbars.
MonitorNumber (Page 829)	R	-	-	Specifies the number of the monitor whose content is displayed in the screen window.
Name	-	-	-	-
ProcessTag	-	-	-	-
ScreenName (Page 904)	R	-	-	Specifies the screen to be displayed in the screen window in runtime.
ScreenScaleMode (Page 905)	R	-	-	Specifies the scaling mode of the displayed screen.
ServerPrefix	RW	-	-	Specifies the server on which the screen is located that is displayed in the screen window in runtime or returns the server name.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ShowCaption (Page 931)	R	-	-	Specifies whether the title bar is shown.
ShowScrollbars (Page 944)	R	-	-	Specifies whether scroll bars are displayed.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TagPrefix (Page 1001)	RW	-	-	Specifies the tag prefix that is prefixed for all tags contained in the screen window.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
TopOffset (Page 1089)	RW	-	-	Sets the vertical shift of the zero point of the display of a screen that is larger than the screen window.
VerticalScrollbarPosi- tion (Page 1206)	RW	-	-	Specifies the position of the slider in the vertical scroll bar of the object.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.
WindowCloseEnabled (Page 1224)	R	-	-	Specifies whether a window can be closed in Run- time.
WindowMaximizeEna- bled (Page 1224)	R	-	-	Specifies whether the object can be maximized in Runtime.
WindowMovingEnabled (Page 1225)	R	-	-	Specifies whether the object can be moved in Run- time.
WindowOnTop (Page 1226)	R	-	-	Specifies whether the object always remains in the foreground in Runtime.
WindowSizingEnabled (Page 1227)	R	-	-	Specifies whether the size can be changed.
WindowStartupPosition (Page 1228)	R	-	-	Specifies the position and the mode of the screen window when an independent screen window is con- figured.
ZoomFactor (Page 1253)	RW	-	-	Specifies the zoom factor of a screen or screen win- dow.

Table 1-88 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cy- cle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during run- time.

ScriptDiagnostics (RT Professional)**Description**

Represents the "ApplicationWindow" object. The ApplicationWindow object is an element of the ScreenItems list.

Type identifier in VBS

HMIApplicationWindow

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-89 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
BorderEnabled (Page 569)	RW	-	-	Specifies whether the window is displayed with a border in runtime.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
GSCRuntimeAllowed	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
ShowCaption (Page 931)	RW	-	-	Specifies whether the title bar is shown.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Template (Page 1003)	RW	-	-	Specifies the template for displaying window content in the "Print job/Script diagnostics" object.
Top (Page 1086)	-	-	-	-
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.
WindowCloseEnabled (Page 1224)	RW	-	-	Specifies whether a window can be closed in Runtime.
WindowMaximizeEnabled (Page 1224)	RW	-	-	Specifies whether the object can be maximized in Runtime.
WindowMovingEnabled (Page 1225)	RW	-	-	Specifies whether the object can be moved in Runtime.
WindowOnTop (Page 1226)	RW	-	-	Specifies whether the object always remains in the foreground in Runtime.
WindowsContents (Page 1226)	RW	-	-	Specifies the content of the print job or script diagnostics.
WindowSizingEnabled (Page 1227)	RW	-	-	Specifies whether the size can be changed.

Table 1-90 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
AUTOHOTSPOT	Yes	-	-	Enables the permanent area or the root screen.
AUTOHOTSPOT	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
AUTOHOTSPOT	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

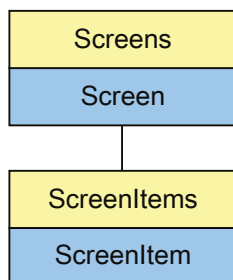
See also

Activate (Page 1254)

ActivateDynamic (Page 1256)

DeactivateDynamic (Page 1263)

Name (Page 831)

Slider (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "Slider" object. The Slider object is an element of the ScreenItems list.

Type identifier in VBS

HMISlider

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-91 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AlarmLowerLimitColor	-	-	-	-
AlarmUpperLimitColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Authorization (Page 521)	RW	RW	RW	Specifies operating rights.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern of the selected object.
BackPicture (Page 542)	RW	-	-	Specifies the background graphic.
BarBackColor (Page 543)	RW	RW	RW	Specifies the color of the bar background for the selected object.
BarColor (Page 546)	RW	RW	RW	Specifies the color of the bar.
BarOrientation	-	-	-	-
BorderBackColor	-	-	-	-
BorderBrightColor3D (Page 565)	RW	RW	RW	Specifies the border color of the following border parts for 3D presentation.
BorderColor	-	-	-	-
BorderInnerStyle3D (Page 576)	RW	-	-	Specifies the appearance of the inner part of the object border.
BorderInnerWidth3D (Page 577)	RW	RW	RW	Specifies the width of the inner border for 3D display.
BorderOuterStyle3D	-	-	-	-
BorderOuterWidth3D (Page 578)	RW	RW	RW	Specifies the width of the outer border for 3D display.
BorderShadeColor3D (Page 579)	RW	RW	RW	Specifies the border color of the following border parts for 3D presentation.
BorderWidth (Page 581)	RW	RW	RW	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (Page 591)	RW	RW	RW	Specifies the text to be displayed in the header.
CompatibilityMode	-	-	-	-
ContinuousChange (Page 637)	RW	-	-	Specifies whether to transfer the value of the "ProcessValue" when the mouse button is released, or immediately on a change of the slider position in runtime.
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
Flashing (Page 696)	RW	-	-	Specifies that flashing is activated.
FocusColor (Page 707)	RW	RW	RW	Specifies the color of the focus border when the object has the focus.
FocusWidth (Page 708)	RW	RW	RW	Specifies the border width when the object has the focus.
Font (Page 709)	RW	-	-	Specifies the font.
ForeColor (Page 716)	RW	RW	RW	Specifies the font color of the text.
ForeColorTransparency	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.

Properties	RT Professional	RT Advanced	Panel RT	Description
LabelColor (Page 762)	RW	RW	RW	Specifies the color of the scale label.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LimitRangeCollection	-	-	-	-
Location	-	-	-	-
MaximumValue (Page 803)	RW	RW	RW	Specifies the maximum value of the scale in the selected object.
MinimumValue (Page 826)	RW	RW	RW	Specifies the minimum value of the scale in the selected object.
Name (Page 831)	RW	-	-	Specifies the object name.
PositionFont (Page 875)	RW	-	-	Sets the font for the label of the slider.
ProcessValue (Page 877)	RW	RW	RW	Specifies the default for the value to be displayed.
ScaleLabelFieldLength	-	-	-	-
ScalePosition	-	-	-	-
ShowBar (Page 930)	RW	RW	RW	Specifies whether the displayed process value is also shown by a filled bar.
ShowLimitLines	-	-	-	-
ShowLimitMarkers	-	-	-	-
ShowLimitRanges	-	-	-	-
ShowPosition (Page 942)	RW	RW	RW	Adds a numerical display of the value of the slider position.
ShowScale	-	-	-	-
ShowThumb (Page 949)	R	RW	RW	Displays the slider of the "Slider" object.
ShowTickLabels (Page 950)	R	RW	RW	Specifies whether the label is shown in the scale.
ShowTicks (Page 950)	R	RW	RW	Specifies whether the marks in the scale are displayed.
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ThumbBackColor (Page 1010)	R	RW	RW	Specifies the background color of the slider.
ThumbPicture (Page 1010)	R	-	-	Returns a graphic for the slider element in the slider.
TickStyle (Page 1012)	R	-	-	Returns how the marks are displayed in the scale.
Top (Page 1086)	R	RW	RW	Specifies the value of the Y coordinate.
Transparency (Page 1091)	R	-	-	Returns the object transparency in percent.
UseDesignColorScheme (Page 1152)	R	-	-	Returns whether the colors defined in the global color scheme of the current design are used for this object.

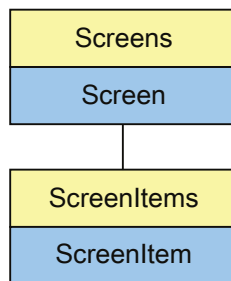
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
UseDesignShadowSet- tings (Page 1153)	R	-	-	Returns whether the object is displayed with global shading.
UseTwoHandOperation	-	-	-	-
Visible (Page 1208)	R	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-92 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

SmartClientView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "Sm@rtClient View" object. The SmartClientView object is an element of the ScreenItems list.

Type identifier in VBS

HMISmartClientView

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-93 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AdressPreview	-	-	-	-
AllowMenu	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
ConnectionType	-	-	-	-
ConnectOnStart	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
EncryptCommunication	-	-	-	-
Flashing	-	-	-	-
Font	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
LocalCursor	-	-	-	-
Location	-	-	-	-
Machine	-	-	-	-
MachineName (Page 801)	-	RW	RW	Sets the network ID of the device that is to be monitored.
Name	-	-	-	-
Password (Page 863)	-	RW	RW	Sets the password for remote monitoring.
PasswordsMustBeEncrypted	-	-	-	-
ScaleDenominator	-	-	-	-
ScaleNumerator	-	-	-	-
Scaling	-	-	-	-
ServerScale	-	-	-	-
Shared	-	-	-	-
ShowControls	-	-	-	-
Size	-	-	-	-

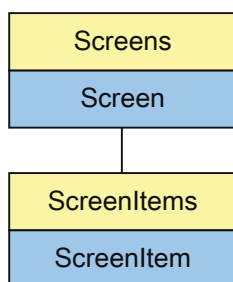
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TcpPortNr	-	-	-	-
Top (Page 1086)	-	RW	RW	Specifies the value of the Y coordinate.
UseCurserKeyScroll	-	-	-	-
ViewOnly (Page 1207)	-	RW	RW	Specifies whether the Sm@rtClient display will be used for remote monitoring or remote maintenance.
Visible (Page 1208)	-	RW	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-94 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	-	Yes	Yes	Enables the permanent area or the root screen.

StatusForce (Panels, Comfort Panels, RT Advanced)

Description



Represents the "Watch table" object. The StatusForce object is an element of the ScreenItems list.

If you change the settings for this object with a user-defined function, the changed settings are retained even after the screen is called again.

Type identifier in VBS

HMISStatusForce

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-95 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Appearance	-	-	-	-
Authorization (Page 521)	-	RW	RW	Specifies operating rights.
BackColor (Page 530)	-	RW	RW	Specifies the background color.
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextBit	-	-	-	-
ColumnTextConnection	-	-	-	-
ColumnTextDataType	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ColumnTextDbNumber	-	-	-	-
ColumnTextFormat	-	-	-	-
ColumnTextOffset	-	-	-	-
ColumnTextRead	-	-	-	-
ColumnTextType	-	-	-	-
ColumnTextWrite	-	-	-	-
ColumnWidth	-	-	-	-
CornerRadius	-	-	-	-
CountVisibleItems	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (Page 707)	-	RW	RW	Specifies the color of the focus border when the object has the focus.
FocusWidth (Page 708)	-	RW	RW	Specifies the border width when the object has the focus.
Height	-	-	-	-
IsRunningUnderCE	-	-	-	-
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
ListAreaHeight	-	-	-	-
ListAreaLeft	-	-	-	-
ListAreaTop	-	-	-	-
ListAreaWidth	-	-	-	-
Location	-	-	-	-
Name	-	-	-	-
Object	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
PLCFilter	-	-	-	-
SelectionBackColor (Page 917)	-	RW	RW	Specifies the background color of the selected cells.
SelectionForeColor (Page 919)	-	RW	RW	Specifies the foreground color of the selected cells.
SetOfVisibleColumns	-	-	-	-

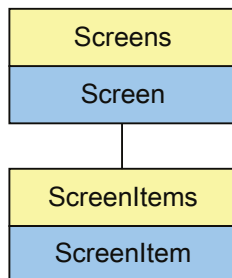
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ShowReadButton	-	-	-	-
ShowTableGridlines (Page 948)	-	RW	RW	Specifies whether grid lines are shown in the table.
ShowWriteButton	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (Page 991)	-	RW	RW	Specifies the background color of the table cells.
TableEvenRowBack- Color	-	-	-	-
TableFont	-	-	-	-
TableForeColor (Page 994)	-	RW	RW	Specifies the text color used in the table cells of the object.
TableGridlineColor	-	-	-	-
TableHeaderBackColor (Page 996)	-	RW	RW	Specifies the background color in the header of the table.
TableHeaderBackFill- Style	-	-	-	-
TableHeaderBorder- BackColor	-	-	-	-
TableHeaderBorderCol- or	-	-	-	-
TableHeaderBorder- Width	-	-	-	-
TableHeaderCornerRa- dius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGra- dientColor	-	-	-	-
TableHeaderFirstGra- dientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderForeColor (Page 998)	-	R	R	Returns the text color in the header of the table of the specified object.
TableHeaderMiddleGra- dientColor	-	-	-	-
TableHeaderPadding- Bottom	-	-	-	-
TableHeaderPaddin- gLeft	-	-	-	-
TableHeaderPadding- Right	-	-	-	-
TableHeaderPadding- Top	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TableHeaderSecond-GradientColor	-	-	-	-
TableHeaderSecond-GradientOffset	-	-	-	-
ToolTipText (Page 1085)	-	R	R	Specifies the tooltip text.
Top (Page 1086)	-	R	R	Returns the value of the Y coordinate.
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGra- dient	-	-	-	-
UseDesignColorSche- ma	-	-	-	-
UseTableHeaderFirst- Gradient	-	-	-	-
UseTableHeaderSe- condGradient	-	-	-	-
Visible (Page 1208)	-	R	R	Returns whether the specified object is visible.
Width	-	-	-	-

Table 1-96 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	-	Yes	Yes	Enables the permanent area or the root screen.

Switch (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "Switch" object. The Switch object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode:

Property	"Switch with text"	"Switch with graphic"	"Switch"
CaptionColor (Page 593)	--	--	x
CaptionText (Page 594)	--	--	x
HorizontalAlignment (Page 740)	X	--	--
InnerBackColorOff (Page 748)	--	--	x
InnerBackColorOn (Page 749)	--	--	x
TextOn (Page 1006)	x	--	x
TextOff (Page 1006)	x	--	x
VerticalAlignment (Page 1204)	x	--	--

Type identifier in VBS

HMISwitch

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-97 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AboveUpperLimitColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	-	RW	RW	Specifies operating rights.
BackColor (Page 530)	-	RW	RW	Specifies the background color.
BackFillStyle	-	-	-	-
BackFlashingColorOff	-	-	-	-
BackFlashingColorOn	-	-	-	-
BackFlashingEnabled	-	-	-	-
BackFlashingRate	-	-	-	-
BelowLowerLimitColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderBrightColor3D	-	-	-	-
BorderColor	-	-	-	-
BorderFlashingColorOff	-	-	-	-
BorderFlashingColorOn	-	-	-	-
BorderFlashingEnabled	-	-	-	-
BorderFlashingRate	-	-	-	-
BorderShadeColor3D	-	-	-	-
BorderWidth	-	-	-	-
BorderWidth3D	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionColor (Page 593)	-	RW	RW	Specifies the color of the text to be displayed in the title bar.
CaptionFont	-	-	-	-
CaptionText (Page 594)	-	RW	RW	Specifies the text to be displayed in the header.
CornerRadius	-	-	-	-
CornerStyle	-	-	-	-
DeviceStyle	-	-	-	-
DrawInsideFrame	-	-	-	-
EdgeStyle	-	-	-	-

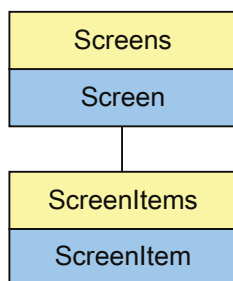
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
FirstGradientColor	-	-	-	-
FirstGradientOffset	-	-	-	-
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingColorOff	-	-	-	-
FlashingColorOn	-	-	-	-
FlashingEnabled	-	-	-	-
FlashingOnLimitViolation	-	-	-	-
FlashingRate	-	-	-	-
FocusColor (Page 707)	-	RW	RW	Specifies the color of the focus border when the object has the focus.
FocusWidth (Page 708)	-	RW	RW	Specifies the border width when the object has the focus.
Font	-	-	-	-
ForeColor (Page 716)	-	RW	RW	Specifies the font color of the text.
Height	-	-	-	-
HelpText (Page 728)	-	RW	RW	Specifies the tooltip that is shown in Runtime as user help for the specified object.
HorizontalAlignment (Page 740)	-	RW	RW	Specifies the horizontal alignment of the text.
HorizontalPictureAlignment	-	-	-	-
HotKey	-	-	-	-
InnerBackColorOff (Page 748)	-	RW	RW	Sets the color underneath the slider of the "Switch" object for the object in OFF state.
InnerBackColorOn (Page 749)	-	RW	RW	Sets the color underneath the slider of the "Switch" object for the object in ON state.
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate.
LineEndShapeStyle	-	-	-	-
Location	-	-	-	-
MiddleGradientColor	-	-	-	-
Mode	-	-	-	-
Name	-	-	-	-
OnValue	-	-	-	-
PictureAreaBottomMargin	-	-	-	-
PictureAreaLeftMargin	-	-	-	-
PictureAreaRightMargin	-	-	-	-
PictureAreaTopMargin	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
PictureAutoSizing	-	-	-	-
PictureOff	-	-	-	-
PictureOn	-	-	-	-
ProcessValue	-	-	-	-
SecondGradientColor	-	-	-	-
SecondGradientOffset	-	-	-	-
ShowCaption	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
SwitchOrientation	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextAreaBottomMargin	-	-	-	-
TextAreaLeftMargin	-	-	-	-
TextAreaRightMargin	-	-	-	-
TextAreaTopMargin	-	-	-	-
TextOff (Page 1006)	-	R	R	Returns the text that is displayed in the "Off" state.
TextOn (Page 1006)	-	R	R	Returns the text that is displayed in the "On" state.
TextOrientation	-	-	-	-
ToolTipText (Page 1085)	-	R	R	Returns the tooltip text.
Top (Page 1086)	-	R	R	Returns the value of the Y coordinate.
UseDesignColorSche- ma	-	-	-	-
UseFirstGradient	-	-	-	-
UseSecondGradient	-	-	-	-
VerticalAlignment (Page 1204)	-	R	R	Returns the vertical alignment of the text in the speci- fied object.
VerticalPictureAlign- ment	-	-	-	-
Visible (Page 1208)	-	R	R	Returns whether the specified object is visible.
Width	-	-	-	-

Table 1-98 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	-	Yes	Yes	Enables the permanent area or the root screen.

SymbolicIOField (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "SymbolicIOField" object. The SymbolicIOField object is an element of the ScreenItems list.

The availability of the following object properties depends on the selected mode:

Property	"Input"	"Output"	"Input/Output"	"Two states"
BackColor (Page 530)	x	x	x	x
BorderColor (Page 566)	--	x	--	x
BorderWidth (Page 581)	--	--	--	x
Enabled (Page 664)	x	--	x	--
HelpText (Page 728)	x	--	x	--
VerticalAlignment (Page 1204)	--	x	--	x
HorizontalAlignment (Page 740)	--	x	--	x

Type identifier in VBS

HMISymbolicIOField

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-99 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AboveUpperLimitColor	-	-	-	-
AcceptOnExit (Page 495)	RW	-	-	Specifies whether the input field is confirmed automatically when it is left.
AdaptBorder (Page 502)	RW	-	-	Specifies whether the border of the object will be dynamically adapted to the text size.
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (Page 518)	RW	-	-	Specifies whether the reason for operating this object is logged.
Assignments (Page 520)	RW	-	-	Specifies a list which contains the assignments between the output value and the output text actually to be output.
Authorization (Page 521)	RW	RW	RW	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	RW	RW	RW	Specifies the background color of the selected object.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern of the selected object.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background of the selected object flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background for the selected object.
BelowLowerLimitColor	-	-	-	-
BitNumber (Page 550)	RW	-	-	Specifies the bit whose status must change to trigger a value change.
BorderBackColor	-	-	-	-
BorderColor (Page 566)	RW	RW	RW	Specifies the line color of the selected object.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the border line color of the selected object for the flash state "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the border line color of the selected object for the flash state "On".

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the border line of the selected object flashes in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line for the selected object.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines for the selected object.
BorderWidth (Page 581)	RW	RW	RW	Specifies the line thickness of the selected object.
BottomMargin	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionBackColor (Page 592)	RW	-	-	Specifies the background color of the caption for the selected object.
CaptionColor (Page 593)	RW	-	-	Specifies the color of the text that is displayed in the header of the selected object.
CornerRadius	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners of the specified object.
CountVisibleItems (Page 642)	RW	-	-	Specifies how many lines the drop-down list is to contain.
CursorControl (Page 643)	RW	-	-	Specifies whether the mouse cursor jumps to the next field of the TAB sequence after leaving a field.
DeviceStyle	-	-	-	-
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line of the selected object with a line thickness greater than 1 should be drawn within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style of the selected object.
EditOnFocus (Page 663)	RW	-	-	Specifies whether data input is immediately possible if the input field is selected using the "Tab" key.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
EvenRowBackColor	-	-	-	-
ExtraHeightOffset	-	-	-	-
FillPatternColor (Page 689)	R	-	-	Returns the color of the fill pattern for the specified object.
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingColorOff (Page 697)	R	-	-	Returns the color of the border line of the specified object for the flashing state "Off".
FlashingColorOn (Page 699)	R	-	-	Returns the color of the border line for the specified object for the flashing state "On".
FlashingEnabled (Page 701)	R	-	-	Returns whether the border line of the specified object flashes in Runtime.
FlashingOnLimitViolation	-	-	-	-
FlashingRate (Page 703)	R	-	-	Returns the flash rate of the border line of the specified object.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
FocusColor	-	-	-	-
FocusWidth	-	-	-	-
Font	-	-	-	-
FontBold (Page 712)	R	-	-	Returns whether the text of the specified object is displayed bold.
FontItalic (Page 712)	R	-	-	Returns whether the text of the specified object is displayed italic.
FontName (Page 713)	R	-	-	Returns the font of the specified object.
FontSize (Page 715)	R	-	-	Returns the font size of the text for the specified object.
FontUnderline (Page 715)	R	-	-	Returns whether the text of the specified object is displayed underlined.
ForeColor (Page 716)	R	RW	RW	Specifies the font color of the text for the selected object.
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
HelpText (Page 728)	-	RW	RW	Specifies the tooltip that is shown in Runtime as user help for the specified object.
HorizontalAlignment (Page 740)	R	RW	RW	Specifies the horizontal alignment of the text within the selected object.
InputValue (Page 751)	R	-	-	Returns the input value.
ItemBorderStyle (Page 754)	R	-	-	Returns the line style of the separation lines in the selection list of the specified object.
Layer (Page 768)	R	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	R	RW	RW	Specifies the value of the X coordinate of the selected object.
LeftMargin	-	-	-	-
LineEndShapeStyle (Page 791)	R	-	-	Returns the shape of the line ends.
Location	-	-	-	-
LogOperation (Page 798)	R	-	-	Returns whether an alarm is output on the alarm system after operation of this object.
Mode (Page 828)	R	-	-	Returns the field type of the specified object.
Name	-	-	-	-
OnValue	-	-	-	-
ProcessValue (Page 877)	R	-	-	Returns the default for the value to be displayed.
RightMargin	-	-	-	-
SelectBackColor (Page 909)	R	-	-	Returns the background color of the selected text entry for the specified object.
SelectForeColor (Page 916)	R	-	-	Returns the color of the selected text entry for the specified object.
SeparatorBackColor (Page 923)	R	-	-	Returns the background color of the broken separation lines in the selection list of the specified object.
SeparatorColor (Page 924)	R	-	-	Returns the color of the separation lines in the selection list of the specified object.

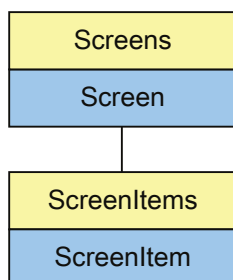
Properties	RT Professional	RT Advanced	Panel RT	Description
SeparatorCornerStyle (Page 924)	R	-	-	Returns the shape of the corners for the specified object.
SeparatorLineEndShapeStyle (Page 925)	R	-	-	Returns the shape of the line end for the specified object.
SeparatorStyle (Page 926)	R	-	-	Returns the line style of the separation lines in the selection list of the specified object.
SeparatorWidth (Page 926)	R	-	-	Returns the separation line width in the selection list of the specified object.
ShowBadTagState (Page 930)	R	-	-	Returns whether the object is grayed out in the event of a bad quality code or tag status.
ShowDropDownButton (Page 933)	R	-	-	Returns whether a button for the selection list is displayed.
ShowDropDownList	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextList (Page 1005)	R	-	-	Returns a list containing the output text actually to be output for each output value.
TextOff	-	-	-	-
TextOn	-	-	-	-
TextOrientation (Page 1007)	R	-	-	Returns the text orientation of the specified object.
ToolTipText (Page 1085)	R	R	R	Returns the tooltip text.
Top (Page 1086)	R	R	R	Returns the value of the Y coordinate of the specified object.
TopMargin	-	-	-	-
Transparency (Page 1091)	R	-	-	Returns the object transparency in percent.
UseDesignColorScheme (Page 1152)	R	-	-	Returns whether the colors defined in the global color scheme of the current design are used for this object.
UseDesignShadowSettings (Page 1153)	R	-	-	Returns whether the object is displayed with the shading defined in the active design.
VerticalAlignment (Page 1204)	R	RW	RW	Specifies the vertical alignment of the text in the specified object.
Visible (Page 1208)	R	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-100 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

SymbolLibrary (Panels, Comfort Panels, RT Advanced, RT Professional)

Description



Represents the "SymbolLibrary" object. The SymbolLibrary object is an element of the ScreenItems list.

Type identifier in VBS

HMISymbolLibrary

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-101 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AboveUpperLimitColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	-	RW	RW	Specifies operating rights.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BelowLowerLimitColor	-	-	-	-
BlinkColor (Page 551)	RW	RW	RW	Specifies the color in which the object flashes in run- time.
BlinkMode (Page 551)	RW	-	-	Specifies the type of flash graphic for the specified object.
BlinkSpeed (Page 552)	RW	-	-	Sets the flash rate.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
ChangeMouseCursor (Page 600)	RW	-	-	Specifies how the appearance of the cursor changes in runtime when it is above the icon.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be oper- ated in runtime.
FillColorMode (Page 688)	RW	-	-	Specifies the type of foreground for the selected ob- ject.
FixedAspectRatio (Page 695)	RW	-	-	Specifies whether the aspect ratio should be re- tained on changes of the icon size, or follow the change dynamically.
Flashing	-	-	-	-
FlashingOnLimitViola- tion	-	-	-	-
Flip (Page 706)	RW	-	-	Specifies whether the symbol is flipped on the verti- cal and / or horizontal center axis of the symbol.
ForeColor (Page 716)	RW	RW	RW	Specifies the font color of the text.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
ProcessValue	-	-	-	-
Rotation (Page 886)	RW	-	-	Specifies the angle of rotation in degrees.

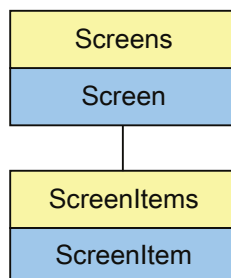
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-102 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

SysDiagControl (Panels, Comfort Panels, RT Advanced)

Description



Represents the "System diagnostic view" object. The SystemDiagnoseView object is an element of the ScreenItems list.

Type identifier in VBS

HMI SysDiagView

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
BackgroundColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
BufferViewColumnOrder	-	-	-	-
BufferViewInternalRowOrder	-	-	-	-
BusyText	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
BV_ColumnWidth_Date	-	-	-	-
BV_ColumnWidth_Event	-	-	-	-
BV_ColumnWidth_EventSeverity	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BV_ColumnWidth_EventState	-	-	-	-
BV_ColumnWidth_Number	-	-	-	-
BV_ColumnWidth_Time	-	-	-	-
BV_ItemText_Date	-	-	-	-
BV_ItemText_Event	-	-	-	-
BV_ItemText_EventSeverity	-	-	-	-
BV_ItemText_EventState	-	-	-	-
BV_ItemText_Number	-	-	-	-
BV_ItemText_Time	-	-	-	-
BV_ShowItem_Date	-	-	-	-
BV_ShowItem_Event	-	-	-	-
BV_ShowItem_EventSeverity	-	-	-	-
BV_ShowItem_EventState	-	-	-	-
BV_ShowItem_Number	-	-	-	-
BV_ShowItem_Time	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnSettings	-	-	-	-
ColumnSettingsBufferView	-	-	-	-
ComplexViewToolbar	-	-	-	-
ComplexViewToolbarBounds	-	-	-	-
ComponentInfoText	-	-	-	-
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	R	R	R	Returns whether the specified object can be operated in Runtime.
EnterButtonVisible	-	-	-	-
Es2RtTableBounds	-	-	-	-
EscButtonVisible	-	-	-	-
EsPreviewType	-	-	-	-
FitToSize	-	-	-	-
FitToSizeLowerRows	-	-	-	-
FitToSizeUpperRows	-	-	-	-
Flashing	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
HomeButtonVisible	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
InfoArea_Background- Color	-	-	-	-
InfoArea_ColumnsMov- able	-	-	-	-
InfoArea_DefaultText- Color	-	-	-	-
InfoArea_ErrorText- BackgroundColor	-	-	-	-
InfoArea_ErrorTextCol- or	-	-	-	-
InfoArea_FocusFrame- Color	-	-	-	-
InfoArea_FocusFrame- Width	-	-	-	-
InfoArea_Font	-	-	-	-
InfoArea_RootNodeText	-	-	-	-
InfoArea_Selection- BackgroundColor	-	-	-	-
InfoArea_SelectionFor- groundColor	-	-	-	-
InfoArea_ShowGrid- Lines	-	-	-	-
InfoArea_TableHeader- BackgroundColor	-	-	-	-
InfoArea_TableHeader- TextColor	-	-	-	-
InputAddressText	-	-	-	-
InspectorViewInternal- ColumnOrder	-	-	-	-
InspectorViewRowOr- der	-	-	-	-
ItemText_AKZ	-	-	-	-
ItemText_Descriptor	-	-	-	-
ItemText_ErrorText	-	-	-	-
ItemText_HardwareRe- vision	-	-	-	-
ItemText_IMDataVer- sion	-	-	-	-
ItemText_Installation- Date	-	-	-	-
ItemText_LADDR	-	-	-	-
ItemText_Manufacturer- ID	-	-	-	-
ItemText_Name	-	-	-	-
ItemText_OKZ	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ItemText_Operation- State	-	-	-	-
ItemText_OrderID	-	-	-	-
ItemText_ProfileID	-	-	-	-
ItemText_Rack	-	-	-	-
ItemText_Revision- Counter	-	-	-	-
ItemText_SerialNumber	-	-	-	-
ItemText_Slot	-	-	-	-
ItemText_SoftwareRevi- sion	-	-	-	-
ItemText_SpecificProfi- leData	-	-	-	-
ItemText_State	-	-	-	-
ItemText_Station	-	-	-	-
ItemText_SubAddress	-	-	-	-
ItemText_SubSlot	-	-	-	-
ItemText_SubSystem	-	-	-	-
ItemText_Type	-	-	-	-
IV_ShowItem_AKZ	-	-	-	-
IV_ShowItem_Descrip- tor	-	-	-	-
IV_ShowItem_ErrorText	-	-	-	-
IV_ShowItem_Hardwar- eRevision	-	-	-	-
IV_ShowItem_IMData- Version	-	-	-	-
IV_ShowItem_Installa- tionDate	-	-	-	-
IV_ShowItem_LADDR	-	-	-	-
IV_ShowItem_Manufac- turerID	-	-	-	-
IV_ShowItem_Name	-	-	-	-
IV_ShowItem_OKZ	-	-	-	-
IV_ShowItem_Opera- tionState	-	-	-	-
IV_ShowItem_OrderID	-	-	-	-
IV_ShowItem_ProfileID	-	-	-	-
IV_ShowItem_Rack	-	-	-	-
IV_ShowItem_Revision- Counter	-	-	-	-
IV_ShowItem_Serial- Number	-	-	-	-
IV_ShowItem_Slot	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
IV_ShowItem_SoftwareRevision	-	-	-	-
IV_ShowItem_Specific-ProfileData	-	-	-	-
IV_ShowItem_State	-	-	-	-
IV_ShowItem_Station	-	-	-	-
IV_ShowItem_SubAddress	-	-	-	-
IV_ShowItem_SubSlot	-	-	-	-
IV_ShowItem_SubSystem	-	-	-	-
IV_ShowItem_Type	-	-	-	-
Layer (Page 768)	R	R	R	Returns the layer in the screen in which an object is located.
Left (Page 774)	R	R	R	Returns the value of the X coordinate.
LinesPerDiagEntry	-	-	-	-
Location	-	-	-	-
MaxToolbarRows	-	-	-	-
Name	-	-	-	-
NavigateTo (Page 834)	RW	-	-	-
NavigationButtons	-	-	-	-
NavigationPath_Font	-	-	-	-
NavigationPath_Root-Text	-	-	-	-
NavigationPath_Text-Color	-	-	-	-
NavigationpathDiagbufferDetailText	-	-	-	-
NavigationpathDiagbufferText	-	-	-	-
OutputAddressText	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
RTPersistence	-	-	-	-
RTPersistenceAuthorization	-	-	-	-
RTPersistenceType	-	-	-	-
ShowMilliseconds	-	-	-	-
ShowNavigationButtons	-	-	-	-
ShowPathInformation	-	-	-	-
ShowSplittedView	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ShowToolBarBack- groundColor	-	-	-	-
SimpleViewToolBar	-	-	-	-
Size	-	-	-	-
SplittedViewRatio	-	-	-	-
StyleItem	-	-	-	-
SupportsInplaceEdit	-	-	-	-
SysDiagBuffButtonVisi- ble	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableEvenRowBack- Color	-	-	-	-
TableGridlineColor	-	-	-	-
TableHeaderBackFill- Style	-	-	-	-
TableHeaderBorder- BackColor	-	-	-	-
TableHeaderBorderCol- or	-	-	-	-
TableHeaderBorder- Width	-	-	-	-
TableHeaderCornerRa- dius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGra- dientColor	-	-	-	-
TableHeaderFirstGra- dientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderMiddleGra- dientColor	-	-	-	-
TableHeaderPadding- Bottom	-	-	-	-
TableHeaderPaddin- gLeft	-	-	-	-
TableHeaderPadding- Right	-	-	-	-
TableHeaderPadding- Top	-	-	-	-
TableHeaderSecond- GradientColor	-	-	-	-
TableHeaderSecond- GradientOffset	-	-	-	-
ToolBar_ButtonsHeight	-	-	-	-
ToolBar_ButtonsWidth	-	-	-	-

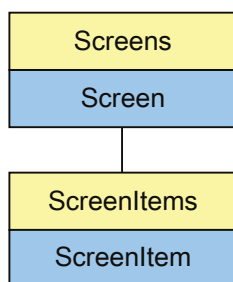
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ToolBarAlignment	-	-	-	-
ToolBarBackgroundColor	-	-	-	-
ToolBarButtonClick	-	-	-	-
ToolBarButtonHeight	-	-	-	-
ToolBarButtonSettings	-	-	-	-
ToolBarButtonWidth	-	-	-	-
ToolBarIconStyle	-	-	-	-
Top (Page 1086)	R	R	R	Returns the value of the Y coordinate.
UnitViewColumnOrder	-	-	-	-
UpdateButtonVisible	-	-	-	-
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGra- dient	-	-	-	-
UseDesignColorSche- ma	-	-	-	-
UseScadaRenderer- Style	-	-	-	-
UseSystemScrollbar- Width	-	-	-	-
UseTableHeaderFirst- Gradient	-	-	-	-
UseTableHeaderSe- condGradient	-	-	-	-
UV_ColumnWidth_AKZ	-	-	-	-
UV_ColumnWidth_De- scriptor	-	-	-	-
UV_ColumnWidth_In- stallationDate	-	-	-	-
UV_Colum- nWidth_LADDR	-	-	-	-
UV_Colum- nWidth_Name	-	-	-	-
UV_ColumnWidth_OKZ	-	-	-	-
UV_ColumnWidth_Op- erationState	-	-	-	-
UV_ColumnWidth_Or- derID	-	-	-	-
UV_ColumnWidth_Pro- fileID	-	-	-	-
UV_ColumnWidth_Rack	-	-	-	-
UV_ColumnWidth_Slot	-	-	-	-
UV_ColumnWidth_Soft- wareRevision	-	-	-	-
UV_ColumnWidth_Spe- cificProfileData	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
UV_ColumnWidth_State	-	-	-	-
UV_ColumnWidth_Sta- tion	-	-	-	-
UV_ColumnWidth_Sub- Address	-	-	-	-
UV_ColumnWidth_Sub- Slot	-	-	-	-
UV_ColumnWidth_Sub- System	-	-	-	-
UV_ColumnWidth_Type	-	-	-	-
UV_ShowItem_AKZ	-	-	-	-
UV_ShowItem_Descrip- tor	-	-	-	-
UV_ShowItem_Install- ationDate	-	-	-	-
UV_ShowItem_LADDR	-	-	-	-
UV_ShowItem_Name	-	-	-	-
UV_ShowItem_OKZ	-	-	-	-
UV_ShowItem_Opera- tionState	-	-	-	-
UV_ShowItem_OrderID	-	-	-	-
UV_ShowItem_ProfileID	-	-	-	-
UV_ShowItem_Rack	-	-	-	-
UV_ShowItem_Slot	-	-	-	-
UV_ShowItem_Softwar- eRevision	-	-	-	-
UV_ShowItem_Specific- ProfileData	-	-	-	-
UV_ShowItem_State	-	-	-	-
UV_ShowItem_Station	-	-	-	-
UV_ShowItem_SubAd- dress	-	-	-	-
UV_ShowItem_SubSlot	-	-	-	-
UV_ShowItem_SubSys- tem	-	-	-	-
UV_ShowItem_Type	-	-	-	-
ViewType	-	-	-	-
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-103 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.

TextField (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Represents the "TextField" object. The TextField object is an element of the ScreenItems list.

Type identifier in VBS

HMITextField

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Table 1-104 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AdaptBorder (Page 502)	RW	-	-	Specifies whether the border of the object will be dynamically adapted to the text size.
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	RW	RW	Specifies the background color.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
BackFlashingEnabled (Page 539)	RW	-	-	Specifies whether the background flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	RW	RW	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled (Page 573)	RW	-	-	Specifies whether the limit of the object can flash in runtime.
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
AUTOHOTSPOT	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	RW	RW	Specifies the line thickness.
BottomMargin	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	RW	RW	Specifies the line style.
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
FitToLargest	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Flashing	-	-	-	-
FlashingColorOff (Page 697)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
FlashingColorOn (Page 699)	RW	-	-	Specifies the color of the border line for the flashing status "On".
FlashingEnabled (Page 701)	RW	-	-	Specifies whether flashing is activated for the object in runtime.
FlashingRate (Page 703)	RW	-	-	Specifies the flash rate of the border line.
Font	-	-	-	-
FontBold (Page 712)	RW	-	-	Specifies whether the text is shown in bold.
FontItalic (Page 712)	RW	-	-	Specifies whether the text is shown in italics.
FontName (Page 713)	RW	-	-	Specifies the font.
FontSize (Page 715)	RW	-	-	Specifies the font size of the text.
FontUnderline (Page 715)	RW	-	-	Specifies whether the text is shown underlined.
ForeColor (Page 716)	RW	RW	RW	Specifies the font color of the text.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalAlignment (Page 740)	RW	RW	RW	Specifies the horizontal alignment of the text.
Layer (Page 768)	R	R	R	Returns the layer in the screen in which an object is located.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate.
LeftMargin	-	-	-	-
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
LineWrap	-	-	-	-
Location	-	-	-	-
Name	-	-	-	-
RelativeFillLevel (Page 884)	RW	-	-	Specifies the fill percentage for the object.
ReSizeable	-	-	-	-
RightMargin	-	-	-	-
RotationAngle (Page 886)	RW	-	-	Specifies the angle of rotation in degrees.
RotationCenterLeft (Page 887)	RW	-	-	Specifies the X coordinate of the pivot point for the object in runtime.
RotationCenterTop (Page 888)	RW	-	-	Specifies the Y coordinate of the pivot point for the object in runtime.
ShowFillLevel (Page 937)	RW	-	-	Specifies whether the selected object is filled.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (Page 1004)	RW	RW	RW	Specifies the label for the text field.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

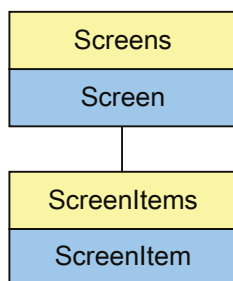
Properties	RT Professional	RT Advanced	Panel RT	Description
TextOrientation (Page 1007)	RW	-	-	Specifies the text orientation.
ToolTipText (Page 1085)	RW	R	R	Specifies the tooltip text.
Top (Page 1086)	RW	RW	RW	Specifies the value of the Y coordinate.
TopMargin	-	-	-	-
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorScheme (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
VerticalAlignment (Page 1204)	RW	RW	RW	Specifies the vertical alignment of the text in the specified object.
Visible (Page 1208)	RW	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-105 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

TrendRulerControl (RT Professional)

Description



Represents the "Value table" object. The TrendRulerControl object is an element of the ScreenItems list.

Type identifier in VBS

HMITrendRulerControl

Example

In the following example, the object with the name "Control1" is moved 16 pixels to the right:

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
```

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettings-ForDesignMode	-	-	-	-
AutoCompleteColumns (Page 523)	RW	-	-	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 523)	RW	-	-	Specifies whether empty rows are shown if the control is longer than the number of configured rows.
AutoPosition (Page 524)	RW	-	-	Specifies whether the value table is automatically placed under the object that supplies the value table with data.
AutoSelectionColors (Page 525)	RW	-	-	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.
AutoSelectionRectColor (Page 526)	RW	-	-	Specifies whether the selection frame is shown with the color defined by the system.
AutoShow (Page 527)	RW	-	-	Specifies whether the value table is automatically displayed.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BlockAlignment (Page 553)	RW	-	-	Specifies the alignment of the text within the column header of the referenced block.
BlockAutoPrecision (Page 553)	RW	-	-	Specifies whether the number of decimal places displayed in the current block is adjusted automatically.
BlockCaption (Page 554)	RW	-	-	Specifies the header for the current block.
BlockCount (Page 554)	RW	-	-	Sets the number of blocks.
BlockDateFormat (Page 555)	RW	-	-	Specifies the date format for the current block.
BlockExponentialFormat (Page 555)	RW	-	-	Specifies whether the values in the current block are displayed in exponential notation.
BlockHideText	-	-	-	-
BlockHideTitleText (Page 556)	RW	-	-	Specifies whether the header of the block is displayed as a text.
BlockId (Page 557)	RW	-	-	References a block with its block ID.
BlockIndex (Page 557)	RW	-	-	References a block.
BlockLength (Page 558)	RW	-	-	Specifies the number of characters for the current block.
BlockName (Page 558)	RW	-	-	Specifies the name of the current block.
BlockPrecision (Page 559)	RW	-	-	Specifies the number of decimal places in the current block.
Blocks	-	-	-	-
BlockShowDate (Page 559)	RW	-	-	Specifies that the date is to be displayed in the current block.
BlockShowIcon (Page 560)	RW	-	-	Specifies whether the content of the current block is displayed as an icon.
BlockShowTitleIcon (Page 561)	RW	-	-	Specifies whether the header of the current block is shown as an icon.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BlockTimeFormat (Page 561)	RW	-	-	Specifies the time format for the current block.
BlockUseSourceFormat (Page 562)	RW	-	-	Specifies whether the format of the connected control is used in the current block.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (Page 591)	RW	-	-	Specifies the text to be displayed in the header.
CellCut (Page 595)	RW	-	-	Specifies whether the contents of the cells are abbreviated if the cells are too narrow.
CellSpaceBottom (Page 596)	RW	-	-	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 597)	RW	-	-	Specifies the left indent used in the table cells.
CellSpaceRight (Page 597)	RW	-	-	Specifies the right indent used in the table cells.
CellSpaceTop (Page 598)	RW	-	-	Specifies the top margin of the table cells.
Closeable (Page 603)	RW	-	-	Specifies whether the object can be closed in run-time.
ColumnAdd (Page 606)	RW	-	-	Creates a new column.
ColumnCount (Page 609)	RW	-	-	Specifies the number of configured columns.
ColumnIndex (Page 613)	RW	-	-	References a column.
ColumnName (Page 616)	RW	-	-	Specifies the name of a column referenced with "ColumnIndex".
ColumnRemove (Page 619)	RW	-	-	Removes a column referenced with its name.
ColumnRepos (Page 619)	RW	-	-	For multiple columns, specifies the position of a column referenced with "ColumnIndex".
ColumnResize (Page 620)	RW	-	-	Specifies whether you can change the width of the columns.
Columns	-	-	-	-
ColumnScrollbar (Page 621)	RW	-	-	Specifies when the horizontal scroll bar is to be displayed.
ColumnSort (Page 624)	RW	-	-	Specifies the sorting type of a column referenced with "ColumnIndex".
ColumnSortIndex (Page 625)	RW	-	-	Specifies the sorting order for a column referenced with "ColumnIndex" in which the columns are sorted one after another.
ColumnTitleAlignment (Page 631)	RW	-	-	Specifies the alignment of the header in a column referenced with "ColumnIndex".
ColumnTitles (Page 632)	RW	-	-	Specifies whether the column header is displayed.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ColumnVisible (Page 633)	RW	-	-	Specifies whether the column referenced with "ColumnIndex" is displayed in the object.
ControlDesignMode (Page 638)	RW	-	-	Specifies the control design.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
ExportDelimiter	-	-	-	-
ExportDirectoryChange- able (Page 678)	RW	-	-	Specifies whether the data export directory can be changed in runtime.
ExportDirectoryname (Page 679)	RW	-	-	Specifies the target directory for the exported runtime data.
ExportFileExtension (Page 679)	RW	-	-	Specifies the file extension of the export file.
ExportFilename (Page 680)	RW	-	-	Specifies the name of the target file for the exported runtime data.
ExportFilename- Changeable (Page 681)	RW	-	-	Specifies whether the export file name can be changed in runtime.
ExportFormat	-	-	-	-
ExportFormatGuid (Page 682)	RW	-	-	Specifies the assignment of ID number and export provider.
ExportFormatName (Page 683)	RW	-	-	Specifies the export file format.
ExportParameters (Page 684)	RW	-	-	Specifies the parameters of the selected format by means of the properties dialog.
ExportSelection (Page 684)	RW	-	-	Specifies which runtime data of the control is exported.
ExportShowDialog (Page 685)	RW	-	-	Specifies whether the data export dialog is shown in runtime.
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Font (Page 709)	RW	-	-	Specifies the font.
GridLineColor (Page 721)	RW	-	-	Specifies the color for the grid lines.
GridLineWidth (Page 723)	RW	-	-	Specifies the width of the separation lines in pixels.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalGridLines (Page 741)	RW	-	-	Specifies whether horizontal separating lines are displayed.
IconSpace (Page 744)	RW	-	-	Specifies the spacing between icons and text in the table cells.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineBackgroundColor	-	-	-	-
LineColor (Page 790)	RW	-	-	Specifies the color of the window separation lines.
LineStyle	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Moveable (Page 829)	RW	-	-	Specifies whether the object can be moved in Run-time.
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
PrintJob (Page 876)	RW	-	-	Specifies a print job that is created in the "Reports" editor.
RowScrollbar (Page 889)	RW	-	-	Specifies when the vertical scroll bar is displayed.
RowTitleAlignment (Page 890)	RW	-	-	Specifies the type of row header alignment.
RowTitles (Page 891)	RW	-	-	Specifies whether numbered column headers are displayed.
RTPersistence (Page 891)	RW	-	-	Specifies whether online configurations are retained after a screen change.
RTPersistenceAuthori- zation (Page 892)	RW	-	-	Specifies the authorization required for online configuration in runtime.
RTPersistenceType (Page 893)	RW	-	-	Specifies how online configurations of WinCC are retained.
RulerColumns	-	-	-	-
RulerType (Page 895)	RW	-	-	Specifies the display mode of the value table.
SelectedCellColor (Page 909)	RW	-	-	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 910)	RW	-	-	Specifies the font color of the selected cell.
SelectedRowColor (Page 912)	RW	-	-	Specifies the background color of the selected row.
SelectedRowForeColor (Page 913)	RW	-	-	Specifies the font color of the selected row.
SelectedTitleColor (Page 915)	RW	-	-	Specifies the background color of the selected table header.
SelectedTitleForeColor (Page 916)	RW	-	-	Specifies the font color of the selected table header.
SelectionColoring (Page 918)	RW	-	-	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 920)	RW	-	-	Specifies whether a selection frame is used for the selected cells or rows.
SelectionRectColor (Page 920)	RW	-	-	Specifies the color of the selection rectangle in the alarm window.
SelectionRectWidth (Page 921)	RW	-	-	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
SelectionType (Page 922)	RW	-	-	Specifies the number of lines you can mark.
ShareSpaceWithSour- ceControl (Page 928)	RW	-	-	Specifies that the display area of the data source is adapted to the size of the value table.
ShowSortButton (Page 945)	RW	-	-	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 946)	RW	-	-	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 946)	RW	-	-	Specifies whether a sort index is displayed.
ShowTitle (Page 951)	RW	-	-	Specifies the style of the window border and window title for the object.
Size	-	-	-	-
Sizeable (Page 956)	RW	-	-	Specifies that the size of an object can be changed in runtime.
SortSequence (Page 958)	RW	-	-	Specifies how the sorting order is changed if the operator clicks on the column header in runtime.
SourceControl (Page 959)	RW	-	-	Specifies the trend or table view to which the value table is connected.
SourceControlType (Page 959)	RW	-	-	Sets the data source type.
StatisticAreaColumns	-	-	-	-
StatisticResultColumns	-	-	-	-
StatusbarBackColor (Page 963)	RW	-	-	Specifies the background color for status bars.
StatusbarElementAdd (Page 964)	RW	-	-	Creates a new, user-defined status bar element.
StatusbarElementAuto- Size (Page 965)	RW	-	-	Specifies whether the width of a status bar element referenced with "StatusbarElementIndex" is set automatically.
StatusbarElement- Count (Page 966)	RW	-	-	Specifies the number of configured status bar elements.
StatusbarElementIcon- Id (Page 967)	RW	-	-	References a status bar element with its icon ID.
StatusbarElementID (Page 967)	RW	-	-	References a status bar element with its element ID.
StatusbarElementIndex (Page 968)	RW	-	-	References a status bar element.
StatusbarElement- Name (Page 969)	RW	-	-	Specifies the name of a status bar element referenced with "StatusbarElementIndex".
StatusbarElementRe- move (Page 970)	RW	-	-	Removes the user-defined status bar element referenced with its name.
StatusbarElementRe- name (Page 971)	RW	-	-	Specifies the new name of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementRe- pos (Page 972)	RW	-	-	Specifies the position of an element referenced with "StatusbarElementIndex" in the status bar of the object.

Properties	RT Professional	RT Advanced	Panel RT	Description
StatusbarElements (Page 973)	-	-	-	-
StatusbarElementText (Page 973)	RW	-	-	Specifies the text of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementTooltipText (Page 974)	RW	-	-	Specifies the text for the tooltip of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementUser-Defined (Page 974)	RW	-	-	Specifies whether the configuration engineer has added a status bar element referenced with "StatusbarElementIndex" as a new user-defined element.
StatusbarElementVisible (Page 975)	RW	-	-	Specifies whether the status bar element referenced with "StatusbarElementIndex" is displayed in the object.
StatusbarElementWidth (Page 976)	RW	-	-	Specifies the width in pixels of a status bar element referenced with "StatusbarElementIndex".
StatusbarFont (Page 977)	RW	-	-	Specifies the font of the texts in the status bar.
StatusbarFontColor (Page 978)	RW	-	-	Specifies the color of the texts in the status bar.
StatusbarShowTooltips (Page 979)	RW	-	-	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 980)	RW	-	-	Specifies the default text in the status bar.
StatusbarUseBackColor (Page 980)	RW	-	-	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 981)	RW	-	-	Specifies whether the status bar of the control is displayed.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (Page 992)	RW	-	-	Specifies the background color of the table rows in the object.
TableColor2 (Page 992)	RW	-	-	Specifies the second background color of the table rows in the object.
TableForeColor (Page 994)	RW	-	-	Specifies the text color used in the table cells of the object.
TableForeColor2 (Page 995)	RW	-	-	Specifies the second text color used in the table cells of the object.
TitleColor (Page 1049)	RW	-	-	Specifies the background color of the table header.
TitleCut (Page 1050)	RW	-	-	Specifies whether the content of the fields in a title bar is to be shortened if the column is too narrow.
TitleDarkShadowColor (Page 1050)	RW	-	-	Specifies the color for the dark side of the 3D shading for column and row headers in the object table.
TitleForeColor (Page 1051)	RW	-	-	Specifies the text color for the table column and row headers of the object.
TitleGridLineColor (Page 1052)	RW	-	-	Specifies the color of the separation lines in the title bar of the table.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TitleLightShadowColor (Page 1053)	RW	-	-	Specifies the color for the light side of the 3D shading for table column and row headers of the object.
TitleSort (Page 1054)	RW	-	-	Specifies how sorting by column header is triggered.
TitleStyle (Page 1055)	RW	-	-	Specifies whether to use a shading color for the text in the column headers.
ToolBarAlignment (Page 1063)	RW	-	-	Specifies the position of the toolbar.
ToolBarBackColor (Page 1064)	RW	-	-	Specifies the background color of the toolbar.
ToolBarButtonActive (Page 1065)	RW	-	-	Specifies whether the function linked to the button is activated in runtime.
ToolBarButtonAdd (Page 1066)	RW	-	-	Creates a new user-defined button in the object's toolbar.
ToolBarButtonAuthori- zation (Page 1066)	RW	-	-	Specifies the authorization for the selected key function.
ToolBarButtonBe- ginGroup (Page 1067)	RW	-	-	Inserts a leading separator (vertical line) for the selected key function on the toolbar.
ToolBarButtonClick (Page 1068)	RW	-	-	Clicks a toolbar button.
ToolBarButtonCount (Page 1069)	RW	-	-	Specifies the number of configured buttons in the toolbar.
ToolBarButtonEnabled (Page 1070)	RW	-	-	Specifies whether a user-defined toolbar button can be operated.
ToolBarButtonHotKey (Page 1071)	RW	-	-	Specifies the hot key for the selected object button.
ToolBarButtonID (Page 1071)	RW	-	-	References a button with its ID.
ToolBarButtonIndex (Page 1072)	RW	-	-	References a button.
ToolBarButtonLocked (Page 1073)	RW	-	-	Specifies whether the locked, pressed state of a user-defined button referenced with "ToolBarButtonIndex" is displayed.
ToolBarButtonName (Page 1074)	RW	-	-	Specifies the name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRemove (Page 1075)	RW	-	-	Removes a user-defined button referenced with its name.
ToolBarButtonRename (Page 1076)	RW	-	-	Specifies the new name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRepos (Page 1077)	RW	-	-	Specifies the position of a button referenced with "ToolBarButtonIndex" in the toolbar of the object.
ToolBarButtons	-	-	-	-
ToolBarButtonTooltip- Text (Page 1078)	RW	-	-	Specifies the tooltip text for the user-defined button in the toolbar.
ToolBarButtonUserDe- fined (Page 1079)	RW	-	-	Specifies whether the project engineer has added the toolbar button as new user-defined button.
ToolBarButtonVisible (Page 1080)	RW	-	-	Specifies whether the button referenced with "ToolBarButtonIndex" is displayed in the toolbar.

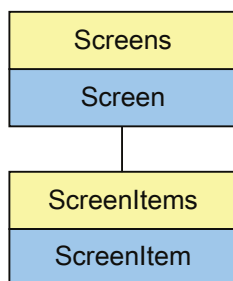
Properties	RT Professional	RT Advanced	Panel RT	Description
ToolbarShowTooltips (Page 1081)	RW	-	-	Specifies whether the tooltips for the key functions are displayed in runtime.
ToolbarUseBackColor (Page 1082)	RW	-	-	Specifies whether the background color for the toolbar is shown.
ToolbarUseHotKeys (Page 1083)	RW	-	-	Specifies whether the hotkeys are activated for buttons in the toolbar.
ToolbarVisible (Page 1084)	RW	-	-	Specifies whether the toolbar of the control is displayed.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
UseSelectedTitleColor (Page 1159)	RW	-	-	Specifies whether a selection color is used for the headers of selected table cells.
UseSourceBackColors (Page 1160)	RW	-	-	Specifies the use of the background color of the data source.
UseSourceForeColor (Page 1161)	RW	-	-	Specifies the use of the font color of the data source.
UseTableColor2 (Page 1161)	RW	-	-	Specifies whether a second row color is used for the representation of the table.
VerticalGridLines (Page 1205)	RW	-	-	Specifies whether vertical separation lines are displayed.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-106 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
Export (Page 1267)	Yes	-	-	Executes the "Export log" or "Export data" key function of the control.
GetRow (Page 1278)	Yes	-	-	Returns the row object designated by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 1280)	Yes	-	-	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetRulerBlock (Page 1281)	Yes	-	-	Returns the block object of the evaluation table designated by name or index as "ICCAxRulerBlock" type.
GetRulerBlockCollection (Page 1282)	Yes	-	-	Returns the list of all block objects of the evaluation table as "ICCAxCollection" type.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Methods	RT Professional	RT Advanced	Panel RT	Description
GetRulerColumn (Page 1283)	Yes	-	-	Returns the column object of the evaluation table designated by name or index as "ICCAxRulerColumn" type.
GetRulerColumnCollection (Page 1284)	Yes	-	-	Returns the list of all column objects of the evaluation table as "ICCAxCollection" type.
GetSelectedRow (Page 1286)	Yes	-	-	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 1287)	Yes	-	-	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatisticAreaColumn (Page 1289)	Yes	-	-	Returns the column object of the statistics area window of the evaluation table designated by name or index as "ICCAxRulerColumn" type.
GetStatisticAreaColumnCollection (Page 1290)	Yes	-	-	Returns the list of all column objects of the statistics area window of the evaluation table as "ICCAxCollection" type.
GetStatisticResultColumn (Page 1291)	Yes	-	-	Returns the column object of the statistics window of the evaluation table designated by name or index as "ICCAxRulerColumn" type.
GetStatisticResultColumnCollection (Page 1292)	Yes	-	-	Returns the list of all column objects of the statistics window of the evaluation table as "ICCAxCollection" type.
GetStatusbarElement (Page 1293)	Yes	-	-	Returns the element of the control status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 1294)	Yes	-	-	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetToolBarButton (Page 1301)	Yes	-	-	Returns the control toolbar button function designated by name or index as "ICCAxToolBarButton" type.
GetToolBarButtonCollection (Page 1302)	Yes	-	-	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
Print (Page 1331)	Yes	-	-	Executes the "Print" button function of the control.
SelectAll (Page 1346)	Yes	-	-	Selects all rows in a table-based control.
SelectRow (Page 1347)	Yes	-	-	Selects a specific row in a table-based control.
ShowHelp (Page 1352)	Yes	-	-	Executes the "Help" button function of the control.
ShowPropertyDialog (Page 1356)	Yes	-	-	Executes the "Configuration dialog" button function of the control.
UnselectAll (Page 1365)	Yes	-	-	Removes all selections from the cells of a table-based control.
UnselectRow (Page 1365)	Yes	-	-	Removes the selections from a specific cell of a table-based control.

TrendView (Panels, Comfort Panels, RT Advanced)**Description**

Represents the "TrendView" object. The TrendView object is an element of the ScreenItems list.

Type identifier in VBS

HMITrendView

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-107 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	-	RW	RW	Specifies the operating rights of the selected object in runtime.
AxisXBunchCount	-	-	-	-
AxisXMarkCount	-	-	-	-
AxisXNoOfDigits	-	-	-	-
AxisXShowBunchVal- ues	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AxisXStyle	-	-	-	-
AxisY1BunchCount	-	-	-	-
AxisY1MarkCount	-	-	-	-
AxisY1ShowBunchVal- ues	-	-	-	-
AxisY2BunchCount	-	-	-	-
AxisY2MarkCount	-	-	-	-
AxisY2ShowBunchVal- ues	-	-	-	-
BackColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBarHeight	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientCol- or	-	-	-	-
ButtonFirstGradientOff- set	-	-	-	-
ButtonMiddleGradient- Color	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradient- Color	-	-	-	-
ButtonSecondGradien- tOffset	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextDateTime	-	-	-	-
ColumnTextTagCon- nection	-	-	-	-
ColumnTextTrend	-	-	-	-
ColumnTextValue	-	-	-	-
ColumnTextXValue	-	-	-	-
ColumnWidth	-	-	-	-
CornerRadius	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
CountVisibleItems	-	-	-	-
Curves	-	-	-	-
DeviceStyle	-	-	-	-
DiagramAreaHeight	-	-	-	-
DiagramAreaLeft	-	-	-	-
DiagramAreaTop	-	-	-	-
DiagramAreaWidth	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	-	RW	RW	Specifies whether the selected object can be operated in runtime.
EnableNavigateButtons	-	-	-	-
EnableNavigateKeys	-	-	-	-
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (Page 707)	-	RW	RW	Specifies the color for the focus frame of the selected object when it is in focus.
FocusWidth (Page 708)	-	RW	RW	Specifies the border width of the specified object when the object is in focus.
Font	-	-	-	-
GridlineAxis	-	-	-	-
GridlineColor	-	-	-	-
GridlineEnabled	-	-	-	-
GridlineFillColor	-	-	-	-
GridlineStyle	-	-	-	-
Height	-	-	-	-
HelpText (Page 728)	-	RW	RW	Specifies the tooltip that is shown in runtime as user help for the specified object.
IsRunningUnderCE	-	-	-	-
Layer (Page 768)	-	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	RW	RW	Specifies the value of the X coordinate of the selected object.
Location	-	-	-	-
Look3D	-	-	-	-
MaxNrOfCurves	-	-	-	-
MinNrOfCurves	-	-	-	-
Name	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
RulerColor (Page 894)	-	RW	RW	Defines the color of the scale gradation (help line) of the axis label in the specified object.
ScaleColor (Page 896)	-	RW	RW	Specifies the color of the scale of the selected object.
SelectionBackColor	-	-	-	-
SelectionForeColor	-	-	-	-
ShowRuler (Page 942)	-	RW	RW	Defines whether a scale division (help line) is displayed for an axis label of the specified object.
ShowTableGridLines	-	-	-	-
ShowTimeAxis	-	-	-	-
ShowTimeAxisLabeling	-	-	-	-
ShowValueAxis1	-	-	-	-
ShowValueAxis1Label	-	-	-	-
ShowValueAxis2	-	-	-	-
ShowValueAxis2Label	-	-	-	-
ShowValueTable	-	-	-	-
ShowY1HlpLine	-	-	-	-
ShowY2HlpLine	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (Page 991)	-	RW	RW	Specifies the background color of the table cells for the selected object.
TableColumnsWidthAndOrder	-	-	-	-
TableEvenRowBackColor	-	-	-	-
TableFont	-	-	-	-
TableGridLineColor (Page 995)	-	R	R	Returns the color of the gridlines in the table of the selected object.
TableHeaderBackColor (Page 996)	-	R	R	Returns the background color in the header of the table of the selected object.
TableHeaderBackFill-Style	-	-	-	-
TableHeaderBorder-BackColor	-	-	-	-
TableHeaderBorderColor	-	-	-	-
TableHeaderBorder-Width	-	-	-	-
TableHeaderCornerRadius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
TableHeaderFirstGradientColor	-	-	-	-
TableHeaderFirstGradientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderForeColor (Page 998)	-	R	R	Returns the text color in the header of the table of the selected object.
TableHeaderMiddleGradientColor	-	-	-	-
TableHeaderPadding-Bottom	-	-	-	-
TableHeaderPadding-Left	-	-	-	-
TableHeaderPadding-Right	-	-	-	-
TableHeaderPadding-Top	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
TagForExternalTime	-	-	-	-
TimeAxisBegin	-	-	-	-
TimeAxisBeginTime	-	-	-	-
TimeAxisCountPoints	-	-	-	-
TimeAxisEnd	-	-	-	-
TimeAxisMode	-	-	-	-
TimeAxisRange	-	-	-	-
TimeAxisSide	-	-	-	-
TimeAxisTimeRange	-	-	-	-
ToolbarButtons	-	-	-	-
ToolbarButtonsForMigration	-	-	-	-
ToolbarEnabled	-	-	-	-
ToolbarHeight	-	-	-	-
ToolbarLeft	-	-	-	-
ToolbarStyle	-	-	-	-
ToolbarTop	-	-	-	-
ToolbarWidth	-	-	-	-
Top (Page 1086)	-	R	R	Returns the value of the Y coordinate of the selected object.
TrendsForPrinting	-	-	-	-
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGradient	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
UseDesignColorSche- ma	-	-	-	-
UseTableHeaderFirst- Gradient	-	-	-	-
UseTableHeaderSe- condGradient	-	-	-	-
ValueAxis1AutoRange	-	-	-	-
ValueAxis1Begin	-	-	-	-
ValueAxis1End	-	-	-	-
ValueAxis1LabelLength	-	-	-	-
ValueAxis1Style	-	-	-	-
ValueAxis2AutoRange	-	-	-	-
ValueAxis2Begin	-	-	-	-
ValueAxis2End	-	-	-	-
ValueAxis2LabelLength	-	-	-	-
ValueAxis2Style	-	-	-	-
ValueTableHeight	-	-	-	-
ValueTableLeft	-	-	-	-
ValueTableTop	-	-	-	-
ValueTableWidth	-	-	-	-
ValueY1HlpLine	-	-	-	-
ValueY2HlpLine	-	-	-	-
Visible (Page 1208)	-	R	R	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-108 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	-	Yes	Yes	Enables the permanent area or the root screen.

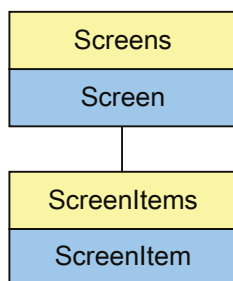
See also

Height (Page 724)

Width (Page 1221)

TubeArcObject (RT Professional)

Description



Represents the "TubeArc" object. The TubeArcObject is an element of the ScreenItems list.

Type identifier in VBS

HMITubeArcObject

Application

In the following example, the object with the name "TubeArcObject1" is moved 10 pixels to the right:

```
'VBS24
Dim objTubeArcObject
Set objTubeArcObject = ScreenItems("TubeArcObject1")
objTubeArcObject.Left = objTubeArcObject.Left + 10
```

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Table 1-109 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	-	-	Specifies the line color.
CornerStyle	-	-	-	-
DrawInsideFrame	-	-	-	-
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
EndAngle (Page 670)	RW	-	-	Specifies the angle at which the end point of the selected object deviates from the zero position (0°).
Flashing	-	-	-	-
FlashingColorOff	-	-	-	-
FlashingColorOn	-	-	-	-
FlashingEnabled	-	-	-	-
FlashingRate	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle	-	-	-	-
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (Page 880)	RW	-	-	Specifies the secondary axis.
RadiusWidth (Page 881)	RW	-	-	Specifies the main axis.
Size	-	-	-	-
StartAngle (Page 960)	RW	-	-	Specifies the angle by which the start point deviates from the zero position (0°).
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.

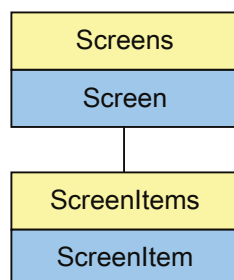
Properties	RT Professional	RT Advanced	Panel RT	Description
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-110 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

TubeDoubleTeeObject (RT Professional)

Description



Represents the "DoubleTee" object. The TubeDoubleTeeObject is an element of the ScreenItems list.

Type identifier in VBS

HMITubeDoubleTeeObject

Application

In the following example, the object with the name "TubeDoubleTeeObject1" is moved 10 pixels to the right:

```
'VBS21
Dim objTubeDoubleTeeObject
Set objTubeDoubleTeeObject = ScreenItems("TubeDoubleTeeObject1")
objTubeDoubleTeeObject.Left = objTubeDoubleTeeObject.Left + 10
```

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-111 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	-	-	Specifies the line color.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.

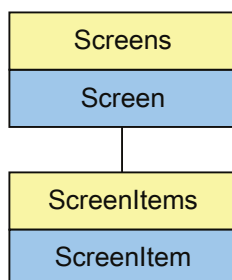
Properties	RT Professional	RT Advanced	Panel RT	Description
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-112 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

TubePolyline (RT Professional)

Description



Represents the "Pipe" object. The TubePolyline object is an element of the ScreenItems list.

Type identifier in VBS

HMITubePolyline

Application

In the following example, the object with the name "TubePolyline1" is moved 10 pixels to the right:

```
'VBS24
Dim objTubePolyline
Set objTubePolyline = ScreenItems("TubePolyline1")
objTubePolyline.Left = objTubePolyline.Left + 10
```

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-113 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
ActualPointIndex (Page 500)	RW	-	-	Specifies the number of the current corner point.
ActualPointLeft (Page 500)	RW	-	-	Specifies the X coordinate of the current corner point with reference to the screen origin.
ActualPointTop (Page 501)	RW	-	-	Specifies the Y coordinate of the current corner point with reference to the screen origin.
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	-	-	Specifies the line color.
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DeviceStyle	-	-	-	-
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
Points	-	-	-	-
PointsCount (Page 874)	RW	-	-	Specifies the number of corner points of the polyline or of the polygon.

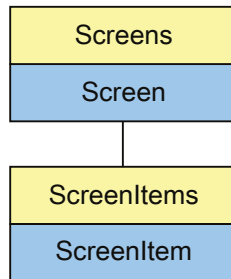
Properties	RT Professional	RT Advanced	Panel RT	Description
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-114 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

TubeTeeObject (RT Professional)

Description



Represents the "Tee" object. The TubeTeeObject is an element of the ScreenItems list.

Type identifier in VBS

HMITubeTeeObject

Application

In the following example, the object with the name "TubeTeeObject1" is moved 10 pixels to the right:

```

'VBS21
Dim objTubeTeeObject
Set objTubeTeeObject = ScreenItems("TubeTeeObject1")
objTubeTeeObject.Left = objTubeTeeObject.Left + 10
  
```

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-115 Properties

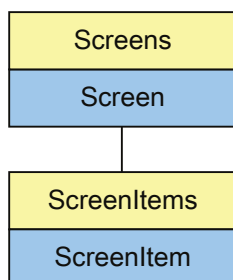
Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Authorization (Page 521)	RW	-	-	Specifies operating rights.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (Page 604)	RW	-	-	Specifies the line color.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
Height (Page 724)	RW	-	-	Specifies the height.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Name	-	-	-	-
RotationAngle (Page 886)	RW	-	-	Specifies the angle of rotation in degrees.
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (Page 1085)	RW	-	-	Specifies the tooltip text.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
Transparency (Page 1091)	RW	-	-	Specifies the object transparency in percent.
UseDesignColorSchema (Page 1152)	RW	-	-	Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.
UseDesignShadowSettings (Page 1153)	RW	-	-	Specifies whether the object is displayed with global shading.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-116 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

UserArchiveControl (RT Professional)

Description



Displays the "Recipe view" object. The UserArchiveControl object is an element of the ScreenItems list.

Type identifier in VBS

HMIUserArchiveControl

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-117 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettingsForDesignMode	-	-	-	-
ArchiveName (Page 517)	RW	-	-	Sets the log name.
ArchiveType (Page 518)	RW	-	-	Sets the log type.
AutoCompleteColumns (Page 523)	RW	-	-	Specifies whether empty columns are shown if the control is wider than the configured columns.
AutoCompleteRows (Page 523)	RW	-	-	Specifies whether empty rows are shown if the control is longer than the number of configured rows.
AutoSelectionColors (Page 525)	RW	-	-	Specifies whether the colors defined by the system are used as the selection colors for cells and rows.
AutoSelectionRectColor (Page 526)	RW	-	-	Specifies whether the selection frame is shown with the color defined by the system.
BackColor (Page 530)	RW	-	-	Specifies the background color.
Blocks	-	-	-	-
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (Page 591)	RW	-	-	Specifies the text to be displayed in the header.
CellCut	-	-	-	-
CellSpaceBottom (Page 596)	RW	-	-	Specifies the bottom margin of the table cells.
CellSpaceLeft (Page 597)	RW	-	-	Specifies the left indent used in the table cells.
CellSpaceRight (Page 597)	RW	-	-	Specifies the right indent used in the table cells.
CellSpaceTop (Page 598)	RW	-	-	Specifies the top margin of the table cells.
Closeable (Page 603)	RW	-	-	Specifies whether the object can be closed in runtime.
ColumnAlias (Page 606)	RW	-	-	Specifies the display name of the column referenced with "ColumnIndex".

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
ColumnAlignment (Page 607)	RW	-	-	Specifies the alignment of the column content in a column referenced with "ColumnIndex".
ColumnAutoPrecisions (Page 608)	RW	-	-	Specifies whether the number of decimal places for a column referenced with "ColumnIndex" is set automatically.
ColumnCaption (Page 608)	RW	-	-	Specifies the header of the current column.
ColumnCount (Page 609)	RW	-	-	Specifies the number of configured columns.
ColumnDateFormat (Page 609)	RW	-	-	Specifies the date format.
ColumnDMVarName (Page 610)	RW	-	-	Specifies the name of a tag.
ColumnExponentialFormat (Page 610)	RW	-	-	Specifies whether the values of a column referenced with "ColumnIndex" are displayed in exponential notation.
ColumnFlagNotNull (Page 611)	RW	-	-	Specifies whether the user archive field assigned to a column referenced with "ColumnIndex" has to have a value.
ColumnFlagUnique (Page 611)	RW	-	-	Specifies whether the user archive field assigned to a column referenced with "ColumnIndex" has to have a unique value.
ColumnHideText (Page 612)	RW	-	-	Specifies whether the text in a column referenced with "ColumnIndex" is hidden.
ColumnHideTitleText (Page 613)	RW	-	-	Specifies whether the header of a column referenced with "ColumnIndex" is hidden.
ColumnIndex (Page 613)	RW	-	-	References a column.
ColumnLeadingZeros (Page 614)	RW	-	-	Specifies how many leading zeros are displayed for the values of a column referenced with "ColumnIndex".
ColumnLength (Page 614)	RW	-	-	Specifies the number of characters displayed in a column referenced with "ColumnIndex".
ColumnMaxValue (Page 615)	RW	-	-	Specifies the maximum value defined in the user archive of a column referenced with "ColumnIndex".
ColumnMinValue (Page 615)	RW	-	-	Specifies the minimum value defined in the user archive of a column referenced with "ColumnIndex".
ColumnName (Page 616)	RW	-	-	Specifies the name of a column referenced with "ColumnIndex".
ColumnPosition (Page 617)	RW	-	-	Specifies the position of a column referenced with "ColumnIndex".
ColumnPrecisions (Page 617)	RW	-	-	Specifies the number of decimal places in a column referenced with "ColumnIndex".

Properties	RT Professional	RT Advanced	Panel RT	Description
ColumnReadAccess (Page 618)	RW	-	-	Specifies the read access authorization defined in the user archive for a column referenced with "ColumnIndex".
ColumnReadOnly (Page 618)	RW	-	-	Specifies whether the values of a column referenced with "ColumnIndex" can only be read.
ColumnRepos (Page 619)	RW	-	-	For multiple columns, specifies the position of a column referenced with "ColumnIndex".
ColumnResize (Page 620)	RW	-	-	Specifies whether you can change the width of the columns.
Columns	-	-	-	-
ColumnScrollbar (Page 621)	RW	-	-	Specifies when the horizontal scroll bar is to be displayed.
ColumnShowDate (Page 622)	RW	-	-	Specifies whether the date is to be displayed in a column referenced with "ColumnIndex".
ColumnShowIcon (Page 623)	RW	-	-	Specifies whether a symbol is to be displayed in a column referenced with "ColumnIndex".
ColumnShowTitleIcon (Page 623)	RW	-	-	Specifies whether a symbol is to be displayed in a column referenced with "ColumnIndex".
ColumnSort (Page 624)	RW	-	-	Specifies the sorting type of a column referenced with "ColumnIndex".
ColumnSortIndex (Page 625)	RW	-	-	Specifies the sorting order for a column referenced with "ColumnIndex" in which the columns are sorted one after another.
ColumnStartValue (Page 625)	RW	-	-	Specifies the start value defined in the user archive of a column referenced with "ColumnIndex".
ColumnStringLength (Page 626)	RW	-	-	Specifies the string length defined in the user archive for a column referenced with "ColumnIndex".
ColumnTimeFormat (Page 630)	RW	-	-	Specifies the time format for a column referenced with "ColumnIndex".
ColumnTitleAlignment (Page 631)	RW	-	-	Specifies the alignment of the header in a column referenced with "ColumnIndex".
ColumnTitles (Page 632)	RW	-	-	Specifies whether the column header is displayed.
ColumnType (Page 632)	RW	-	-	Specifies the data type defined in the user archive of a column referenced with "ColumnIndex".
ColumnVisible (Page 633)	RW	-	-	Specifies whether the column referenced with "ColumnIndex" is displayed in the object.
ColumnWriteAccess (Page 634)	RW	-	-	Specifies the write access authorization defined in the user archive for a column referenced with "ColumnIndex".

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
ControlDesign-Mode (Page 638)	RW	-	-	Specifies the control design.
DataProviderGuid (Page 647)	R	-	-	Determines the GUID of the control.
DataSource	-	-	-	-
DefaultFilterEom	-	-	-	-
Enabled	-	-	-	-
EnableDelete (Page 668)	RW	-	-	Specifies whether data can be deleted in runtime.
EnableEdit (Page 668)	RW	-	-	Specifies whether the data shown can be edited in runtime.
EnableInsert (Page 669)	RW	-	-	Specifies whether data can be inserted in runtime.
ExportDelimiter	-	-	-	-
ExportDirectory-Changeable (Page 678)	RW	-	-	Specifies whether the data export directory can be changed in runtime.
ExportDirectory-name (Page 679)	RW	-	-	Specifies the target directory for the exported runtime data.
ExportFileExtension (Page 679)	RW	-	-	Specifies the file extension of the export file.
ExportFilename (Page 680)	RW	-	-	Specifies the name of the target file for the exported runtime data.
ExportFilename-Changeable (Page 681)	RW	-	-	Specifies whether the export file name can be changed in runtime.
ExportFormat	-	-	-	-
ExportFormatGuid (Page 682)	RW	-	-	Specifies the assignment of ID number and export provider.
ExportFormat-Name (Page 683)	RW	-	-	Specifies the export file format.
ExportParameters (Page 684)	RW	-	-	Specifies the parameters of the selected format by means of the properties dialog.
ExportSelection (Page 684)	RW	-	-	Specifies which runtime data of the control is exported.
ExportShowDialog (Page 685)	RW	-	-	Specifies whether the data export dialog is shown in runtime.
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Filter	-	-	-	-
FilterSQL (Page 692)	RW	-	-	Sets an SQL statement for the filter criterion.
Font (Page 709)	RW	-	-	Specifies the font.
GridLineColor (Page 721)	RW	-	-	Specifies the color for the grid lines.

Properties	RT Professional	RT Advanced	Panel RT	Description
GridLineWidth (Page 723)	RW	-	-	Specifies the width of the separation lines in pixels.
Height (Page 724)	RW	-	-	Specifies the height.
HorizontalGridLines (Page 741)	RW	-	-	Specifies whether horizontal separating lines are displayed.
IconSpace (Page 744)	RW	-	-	Specifies the spacing between icons and text in the table cells.
Layer	-	-	-	-
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineBackground-Color	-	-	-	-
LineColor (Page 790)	RW	-	-	Specifies the color of the window separation lines.
LineStyle	-	-	-	-
LineWidth (Page 794)	RW	-	-	Specifies the line thickness.
Location	-	-	-	-
Moveable (Page 829)	RW	-	-	Specifies whether the object can be moved in runtime.
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
PrintJob (Page 876)	RW	-	-	Specifies a print job that is created in the "Reports" editor.
RowScrollbar (Page 889)	RW	-	-	Specifies when the vertical scroll bar is displayed.
RowTitleAlignment (Page 890)	RW	-	-	Specifies the type of row header alignment.
RowTitles (Page 891)	RW	-	-	Specifies whether numbered column headers are displayed.
RTPersistence (Page 891)	RW	-	-	Specifies whether online configurations are retained after a screen change.
RTPersistenceAuthorization (Page 892)	RW	-	-	Specifies the authorization required for online configuration in runtime.
RTPersistence-Type (Page 893)	RW	-	-	Specifies how online configurations of WinCC are retained.
SelectArchive-Name (Page 908)	RW	-	-	Specifies that the selection dialog for the data source of the recipe view is displayed initially in Runtime.
SelectedCellColor (Page 909)	RW	-	-	Specifies the background color of the selected cell.
SelectedCellForeColor (Page 910)	RW	-	-	Specifies the font color of the selected cell.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
SelectedID (Page 911)	RW	-	-	Specifies the ID of the data record selected in the recipe view.
SelectedRowColor (Page 912)	RW	-	-	Specifies the background color of the selected row.
SelectedRowForeColor (Page 913)	RW	-	-	Specifies the font color of the selected row.
SelectedTitleColor (Page 915)	RW	-	-	Specifies the background color of the selected table header.
SelectedTitleForeColor (Page 916)	RW	-	-	Specifies the font color of the selected table header.
SelectionColoring (Page 918)	RW	-	-	Specifies whether selection colors are used for cells or rows.
SelectionRect (Page 920)	RW	-	-	Specifies whether a selection frame is used for the selected cells or rows.
SelectionRectColor (Page 920)	RW	-	-	Specifies the color of the selection rectangle in the alarm window.
SelectionRectWidth (Page 921)	RW	-	-	Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".
SelectionType (Page 922)	RW	-	-	Specifies the number of lines you can mark.
ShowSortButton (Page 945)	RW	-	-	Specifies whether the sorting button is shown above the vertical scroll bar.
ShowSortIcon (Page 946)	RW	-	-	Specifies whether the sorting icon is displayed.
ShowSortIndex (Page 946)	RW	-	-	Specifies whether a sort index is displayed.
ShowTitle (Page 951)	RW	-	-	Specifies the style of the window border and window title for the object.
Size	-	-	-	-
Sizeable (Page 956)	RW	-	-	Specifies that the size of an object can be changed in runtime.
SortSequence (Page 958)	RW	-	-	Specifies how the sorting order is changed if the operator clicks on the column header in runtime.
StatusbarBackColor (Page 963)	RW	-	-	Specifies the background color for status bars.
StatusbarElementAdd (Page 964)	RW	-	-	Creates a new, user-defined status bar element.
StatusbarElementAutoSize (Page 965)	RW	-	-	Specifies whether the width of a status bar element referenced with "StatusbarElementIndex" is set automatically.
StatusbarElementCount (Page 966)	RW	-	-	Specifies the number of configured status bar elements.
StatusbarElementIconID (Page 967)	RW	-	-	References a status bar element with its icon ID.
StatusbarElementID (Page 967)	RW	-	-	References a status bar element with its element ID.

Properties	RT Professional	RT Advanced	Panel RT	Description
StatusbarElementIndex (Page 968)	RW	-	-	References a status bar element.
StatusbarElementName (Page 969)	RW	-	-	Specifies the name of a status bar element referenced with "StatusbarElementIndex".
StatusbarElementRemove (Page 970)	RW	-	-	Removes the user-defined status bar element referenced with its name.
StatusbarElementRename (Page 971)	RW	-	-	Specifies the new name of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementRepos (Page 972)	RW	-	-	Specifies the position of an element referenced with "StatusbarElementIndex" in the status bar of the object.
StatusbarElements	-	-	-	-
StatusbarElementText (Page 973)	RW	-	-	Specifies the text of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementTooltipText (Page 974)	RW	-	-	Specifies the text for the tooltip of a user-defined status bar element referenced with "StatusbarElementIndex".
StatusbarElementUserDefined (Page 974)	RW	-	-	Specifies whether the configuration engineer has added a status bar element referenced with "StatusbarElementIndex" as a new user-defined element.
StatusbarElementVisible (Page 975)	RW	-	-	Specifies whether the status bar element referenced with "StatusbarElementIndex" is displayed in the object.
StatusbarElementWidth (Page 976)	RW	-	-	Specifies the width in pixels of a status bar element referenced with "StatusbarElementIndex".
StatusbarFont (Page 977)	RW	-	-	Specifies the font of the texts in the status bar.
StatusbarFontColor (Page 978)	RW	-	-	Specifies the color of the texts in the status bar.
StatusbarShowTooltips (Page 979)	RW	-	-	Specifies whether tooltips are displayed for the elements of the status bar in runtime.
StatusbarText (Page 980)	RW	-	-	Specifies the default text in the status bar.
StatusbarUseBackColor (Page 980)	RW	-	-	Specifies whether the background color of the status bar is shown.
StatusbarVisible (Page 981)	RW	-	-	Specifies whether the status bar of the control is displayed.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (Page 992)	RW	-	-	Specifies the background color of the table rows in the object.
TableColor2 (Page 992)	RW	-	-	Specifies the second background color of the table rows in the object.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Professional	RT Advanced	Panel RT	Description
TableForeColor (Page 994)	RW	-	-	Specifies the text color used in the table cells of the object.
TableForeColor2 (Page 995)	RW	-	-	Specifies the second text color used in the table cells of the object.
TimeBase (Page 1027)	RW	-	-	Specifies the time zone in which the time values are displayed.
TitleColor (Page 1049)	RW	-	-	Specifies the background color of the table header.
TitleCut (Page 1050)	RW	-	-	Specifies whether the content of the fields in a title bar is to be shortened if the column is too narrow.
TitleDarkShadow-Color (Page 1050)	RW	-	-	Specifies the color for the dark side of the 3D shading for column and row headers in the object table.
TitleForeColor (Page 1051)	RW	-	-	Specifies the text color for the table column and row headers of the object.
TitleGridLineColor (Page 1052)	RW	-	-	Specifies the color of the separation lines in the title bar of the table.
TitleLightShadow-Color (Page 1053)	RW	-	-	Specifies the color for the light side of the 3D shading for table column and row headers of the object.
TitleSort (Page 1054)	RW	-	-	Specifies how sorting by column header is triggered.
TitleStyle (Page 1055)	RW	-	-	Specifies whether to use a shading color for the text in the column headers.
ToolbarAlignment (Page 1063)	RW	-	-	Specifies the position of the toolbar.
ToolbarBackColor (Page 1064)	RW	-	-	Specifies the background color of the toolbar.
ToolbarButtonActive (Page 1065)	RW	-	-	Specifies whether the function linked to the button is activated in runtime.
ToolbarButtonAdd (Page 1066)	RW	-	-	Creates a new user-defined button in the object's toolbar.
ToolbarButtonAuthorization (Page 1066)	RW	-	-	Specifies the authorization for the selected key function.
ToolbarButtonBeginGroup (Page 1067)	RW	-	-	Inserts a leading separator (vertical line) for the selected key function on the toolbar.
ToolbarButtonClick (Page 1068)	RW	-	-	Clicks a toolbar button.
ToolbarButtonCount (Page 1069)	RW	-	-	Specifies the number of configured buttons in the toolbar.
ToolbarButtonEnabled (Page 1070)	RW	-	-	Specifies whether a user-defined toolbar button can be operated.
ToolbarButtonHotKey (Page 1071)	RW	-	-	Specifies the hot key for the selected object button.

Properties	RT Professional	RT Advanced	Panel RT	Description
ToolBarButtonID (Page 1071)	RW	-	-	References a button with its ID.
ToolBarButtonIndex (Page 1072)	RW	-	-	References a button.
ToolBarButtonLocked (Page 1073)	RW	-	-	Specifies whether the locked, pressed state of a user-defined button referenced with "ToolBarButtonIndex" is displayed.
ToolBarButtonName (Page 1074)	RW	-	-	Specifies the name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRemove (Page 1075)	RW	-	-	Removes a user-defined button referenced with its name.
ToolBarButtonRename (Page 1076)	RW	-	-	Specifies the new name of a user-defined button referenced with "ToolBarButtonIndex".
ToolBarButtonRepos (Page 1077)	RW	-	-	Specifies the position of a button referenced with "ToolBarButtonIndex" in the toolbar of the object.
ToolBarButtons	-	-	-	-
ToolBarButtonTooltipText (Page 1078)	RW	-	-	Specifies the tooltip text for the user-defined button in the toolbar.
ToolBarButtonUserDefined (Page 1079)	RW	-	-	Specifies whether the project engineer has added the toolbar button as new user-defined button.
ToolBarButtonVisible (Page 1080)	RW	-	-	Specifies whether the button referenced with "ToolBarButtonIndex" is displayed in the toolbar.
ToolBarShowToolTips (Page 1081)	RW	-	-	Specifies whether the tooltips for the key functions are displayed in runtime.
ToolBarUseBackColor (Page 1082)	RW	-	-	Specifies whether the background color for the toolbar is shown.
ToolBarUseHotKeys (Page 1083)	RW	-	-	Specifies whether the hotkeys are activated for buttons in the toolbar.
ToolBarVisible (Page 1084)	RW	-	-	Specifies whether the toolbar of the control is displayed.
Top (Page 1086)	RW	-	-	Specifies the value of the Y coordinate.
UseSelectedTitleColor (Page 1159)	RW	-	-	Specifies whether a selection color is used for the headers of selected table cells.
UseTableColor2 (Page 1161)	RW	-	-	Specifies whether a second row color is used for the representation of the table.
VerticalGridLines (Page 1205)	RW	-	-	Specifies whether vertical separation lines are displayed.
Visible (Page 1208)	RW	-	-	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

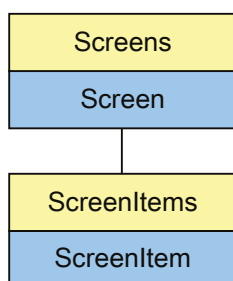
Table 1-118 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property with VBScript during runtime.
CopyRows (Page 1261)	Yes	-	-	Executes the "Copy lines" key function of the control.
CutRows (Page 1263)	Yes	-	-	Executes the "Cut rows" key function of the recipe view.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.
DeleteRows (Page 1265)	Yes	-	-	Executes the "Delete rows" key function of the recipe view.
Export (Page 1267)	Yes	-	-	Executes the "Export log" or "Export data" key function of the control.
GetColumn (Page 1267)	Yes	-	-	Returns the column object of the recipe view designated by name or index as "ICCAxUAColumn" type.
GetColumnCollection (Page 1268)	Yes	-	-	Returns the list of all column objects of the recipe view as "ICCAxCollection" type.
GetRow (Page 1278)	Yes	-	-	Returns the row object designated by its row number in the table-based controls as "ICCAxDataRow" type.
GetRowCollection (Page 1280)	Yes	-	-	Returns the list of all row objects of the table-based controls as "ICCAxDataRowCollection" type.
GetSelectedRow (Page 1286)	Yes	-	-	Returns the selected row object of a table-based control as "ICCAxDataRow" type.
GetSelectedRows (Page 1287)	Yes	-	-	Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.
GetStatusbarElement (Page 1293)	Yes	-	-	Returns the element of the control status bar designated by name or index as "ICCAxStatusbarElement" type.
GetStatusbarElementCollection (Page 1294)	Yes	-	-	Returns the list of all status bar elements of the control as "ICCAxCollection" type.
GetToolBarButton (Page 1301)	Yes	-	-	Returns the control toolbar button function designated by name or index as "ICCAxToolBarButton" type.
GetToolBarButtonCollection (Page 1302)	Yes	-	-	Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.
MoveToFirst (Page 1322)	Yes	-	-	Executes the "First line" button function of the control.
MoveToLast (Page 1324)	Yes	-	-	Executes the "Last data record" button function of the control.
MoveToNext (Page 1325)	Yes	-	-	Executes the "Next data record" button function of the control.
MoveToPrevious (Page 1327)	Yes	-	-	Executes the "Previous data record" button function of the control.

Methods	RT Professional	RT Advanced	Panel RT	Description
PasteRows (Page 1330)	Yes	-	-	Executes the "Insert rows" button function of the recipe view.
Print (Page 1331)	Yes	-	-	Executes the "Print" button function of the control.
ReadTags (Page 1336)	Yes	-	-	Executes the "Read tags" button function of the recipe view.
SelectAll (Page 1346)	Yes	-	-	Selects all rows in a table-based control.
SelectRow (Page 1347)	Yes	-	-	Selects a specific row in a table-based control.
ServerExport (Page 1347)	Yes	-	-	Executes the "Export log" button function of the recipe view.
ServerImport (Page 1348)	Yes	-	-	Executes the "Import log" button function of the recipe view.
ShowHelp (Page 1352)	Yes	-	-	Executes the "Help" button function of the control.
ShowPropertyDialog (Page 1356)	Yes	-	-	Executes the "Configuration dialog" button function of the control.
ShowSelectArchive (Page 1357)	Yes	-	-	Executes the "Select data connection" button function of the recipe view.
ShowSelection (Page 1357)	Yes	-	-	Executes the "Selection dialog" button function of the recipe view.
ShowSelectTimeBase (Page 1358)	Yes	-	-	Executes the "Timebase dialog" button function of the recipe view.
ShowSort (Page 1359)	Yes	-	-	Executes the "Sorting dialog" button function of the recipe view.
UnselectAll (Page 1365)	Yes	-	-	Removes all selections from the cells of a table-based control.
UnselectRow (Page 1365)	Yes	-	-	Removes the selections from a specific cell of a table-based control.
WriteTags (Page 1369)	Yes	-	-	Executes the "Write tags" button function of the recipe view.

UserView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "UserView" object. The UserView object is an element of the ScreenItems list.

Note

The object "Simple UserView" cannot be dynamized with a user-defined function.

Type identifier in VBS

HMIUserView

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-119 Properties

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AnimationIgnore	-	-	-	-
Appearance	-	-	-	-

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Authorization (Page 521)	-	RW	RW	Specifies the operating rights of the selected object in runtime.
BackColor (Page 530)	-	RW	RW	Specifies the background color of the selected object.
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Columns	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextGroup	-	-	-	-
ColumnTextLogTime	-	-	-	-
ColumnTextPassword	-	-	-	-
ColumnTextUser	-	-	-	-
CornerRadius	-	-	-	-
CountVisibleItems	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (Page 664)	RW	RW	RW	Specifies whether the selected object can be operated in runtime.
Es2rtButtonPositions	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height of the selected object.
IsMinPasswordValue-Set	-	-	-	-
IsRunningUnderCE	-	-	-	-
Layer (Page 768)	RW	RW	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	RW	RW	RW	Specifies the value of the X coordinate of the selected object.
ListAreaHeight	-	-	-	-
ListAreaWidth	-	-	-	-
Location	-	-	-	-
MinPasswordValue	-	-	-	-
Name	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
RTPersistence	-	-	-	-

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
RTPersistenceAutho- rization	-	-	-	-
RTPersistenceType	-	-	-	-
SelectionBackColor (Page 917)	-	RW	RW	Specifies the background color of the selected cells.
SelectionForeColor (Page 919)	-	RW	RW	Specifies the foreground color of the selected cells.
ShowColumnHeaders	-	-	-	-
ShowTableGridlines	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (Page 991)	-	RW	RW	Specifies the background color of the table cells for the selected object.
TableEvenRowBack- Color	-	-	-	-
TableFont	-	-	-	-
TableForeColor (Page 994)	-	RW	RW	Specifies the text color used in the table cells of the object.
TableGridLineColor (Page 995)	-	RW	RW	Determines the color of the gridlines in the table of the given object.
TableHeaderBackColor (Page 996)	-	RW	RW	Specifies the background color in the header of the table of the selected object.
TableHeaderBackFill- Style	-	-	-	-
TableHeaderBorder- BackColor	-	-	-	-
TableHeaderBorderCol- or	-	-	-	-
TableHeaderBorder- Width	-	-	-	-
TableHeaderCornerRa- dius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGra- dientColor	-	-	-	-
TableHeaderFirstGra- dientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderForeColor (Page 998)	-	RW	RW	Specifies the text color in the header of the table of the selected object.
TableHeaderMiddleGra- dientColor	-	-	-	-
TableHeaderPadding- Bottom	-	-	-	-

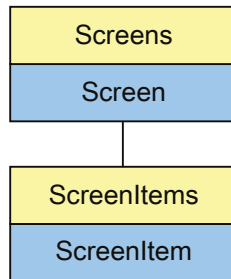
Properties	RT Professional	RT Advanced	Panel RT	Description
TableHeaderPaddingLeft	-	-	-	-
TableHeaderPaddingRight	-	-	-	-
TableHeaderPaddingTop	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
Top (Page 1086)	R	RW	RW	Specifies the value of the Y coordinate of the selected object.
UseDesignColorSchema	-	-	-	-
UseTableHeaderFirstGradient	-	-	-	-
UseTableHeaderSecondGradient	-	-	-	-
VerticalScrolling	-	-	-	-
ViewType	-	-	-	-
ViewTypeForSaveStream	-	-	-	-
Visible (Page 1208)	R	RW	RW	Specifies whether the selected object is visible.
Width (Page 1221)	RW	-	-	Specifies the width of the object in pixels.

Table 1-120 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	Yes	Yes	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

WindowSlider (RT Professional)

Description



Represents the "Window slider" object. The WindowSlider object is an element of the ScreenItems list.

Type identifier in VBS

HMIWindowSlider

Abbreviation	Access in Runtime
R	Read
RW	Read and write
-	No access

Table 1-121 Properties

Properties	RT Profes-sional	RT Ad-vanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (Page 518)	RW	-	-	Specifies whether the reason for operating this object is logged.
Authorization (Page 521)	RW	-	-	Specifies operating rights.
BackColor (Page 530)	RW	-	-	Specifies the background color.
BackColorBottom (Page 533)	RW	-	-	Specifies the color for the lower / right part of the object.

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
BackColorTop (Page 534)	RW	-	-	Specifies the color for the upper / left part of the object.
BackFillStyle (Page 535)	RW	-	-	Specifies the fill pattern.
BackFlashingColorOff (Page 537)	RW	-	-	Specifies the background color for flash state "Off".
BackFlashingColorOn (Page 538)	RW	-	-	Specifies the background color for flash state "On".
AUTOHOTSPOT	RW	-	-	Specifies whether the background flashes in runtime.
BackFlashingRate (Page 540)	RW	-	-	Specifies the flash rate of the background.
BorderBackColor (Page 562)	RW	-	-	Specifies the background color of the broken border line.
BorderColor (Page 566)	RW	-	-	Specifies the line color.
BorderFlashingColorOff (Page 570)	RW	-	-	Specifies the color of the border line for the flashing status "Off".
BorderFlashingColorOn (Page 571)	RW	-	-	Specifies the color of the border line for the flashing status "On".
BorderFlashingEnabled	-	-	-	-
BorderFlashingRate (Page 574)	RW	-	-	Specifies the flash rate of the border line.
BorderStyle (Page 580)	RW	-	-	Specifies the type of border lines.
BorderWidth (Page 581)	RW	-	-	Specifies the line thickness.
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (Page 639)	RW	-	-	Specifies the shape of the corners.
DrawInsideFrame (Page 659)	RW	-	-	Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.
EdgeStyle (Page 661)	RW	-	-	Specifies the line style.
Enabled (Page 664)	RW	-	-	Specifies whether the selected object can be operated in runtime.
FillingDirection (Page 691)	RW	-	-	Specifies the fill direction.
FillPatternColor (Page 689)	RW	-	-	Specifies the color of the fill pattern.
Flashing	-	-	-	-
Height (Page 724)	RW	-	-	Specifies the height.
HighLimitColor (Page 730)	RW	-	-	Specifies the color of the top or right scroll button in a scroll bar.
JumpToLimitsAfterMouseClick (Page 762)	RW	-	-	Specifies whether the slider is set to the associated end value.
Layer (Page 768)	RW	-	-	Specifies the layer in the screen that contains an object.

1.5 VBS object model (Panels, Comfort Panels, RT Advanced, RT Professional)

Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Left (Page 774)	RW	-	-	Specifies the value of the X coordinate.
LineEndShapeStyle (Page 791)	RW	-	-	Specifies the shape of the line ends.
Location	-	-	-	-
LogOperation (Page 798)	RW	-	-	Specifies whether, after operating this object, an alarm is output to the alarm system.
LowLimitColor (Page 800)	RW	-	-	Specifies the color of the bottom or left scroll button in a scroll bar.
MarginToBorder (Page 802)	RW	-	-	Specifies the width of the 3D border in pixels.
MaximumValue (Page 803)	RW	-	-	Specifies the maximum value of the scale in the selected object.
MinimumValue (Page 826)	RW	-	-	Specifies the minimum value of the scale in the selected object.
Name	-	-	-	-
OperationSteps (Page 844)	R	-	-	Returns the number of steps by which the slider of the scrollbar is moved with one mouse click.
ProcessValue (Page 877)	R	-	-	Returns the default for the value to be displayed.
RelativeFillLevel (Page 884)	R	-	-	Returns the percentage to which the object is filled.
ShowBadTagState (Page 930)	R	-	-	Returns whether the object is grayed out in the event of a bad quality code or tag status.
ShowFillLevel (Page 937)	R	-	-	Returns whether the specified object is filled.
Size	-	-	-	-
StyleSettings (Page 986)	R	-	-	Returns the style in which the object is displayed.
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextOrientation (Page 1007)	R	-	-	Returns the text orientation.
ThumbBackColor (Page 1010)	R	-	-	Returns the background color of the slider.
ToolTipText (Page 1085)	R	-	-	Returns the tooltip text.
Top (Page 1086)	R	-	-	Returns the value of the Y coordinate.
Transparency (Page 1091)	R	-	-	Returns the object transparency in percent.
UseDesignColorSche- ma (Page 1152)	R	-	-	Returns whether the colors defined in the global color scheme of the current design are used for this object.
UseDesignShadowSet- tings (Page 1153)	R	-	-	Returns whether the object is displayed with global shading.
Visible (Page 1208)	R	-	-	Returns whether the specified object is visible.

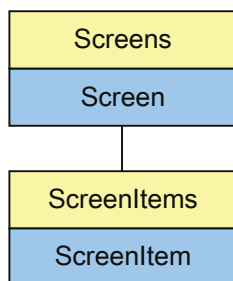
Properties	RT Professional	RT Advanced	Panel RT	Description
Width (Page 1221)	R	-	-	Returns the width of the object in pixels.
WindowsStyle (Page 1228)	R	-	-	Returns whether the object is displayed in the general Windows style.

Table 1-122 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	Yes	-	-	Enables the permanent area or the root screen.
ActivateDynamic (Page 1256)	Yes	-	-	Dynamically activates a trigger and the specified cycle for a property at runtime.
DeactivateDynamic (Page 1263)	Yes	-	-	Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/during runtime.

See also

BorderFlashingEnabled (Page 573)

WLANQualityView (Panels, Comfort Panels, RT Advanced)**Description**

Represents the "WLAN reception" object. The WlanQualityView object is an element of the ScreenItems list.

Type identifier in VBS

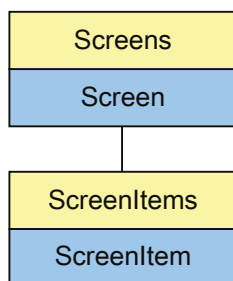
Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-123 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	-	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	-	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	-	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-124 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

ZoneLabelView (Panels, Comfort Panels, RT Advanced)**Description**

Represents the "Zone name" object. The ZoneLabelView object is an element of the ScreenItems list.

Type identifier in VBS

HMIZoneLabelView

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-125 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Font	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	-	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	-	RW	Specifies the value of the X coordinate.

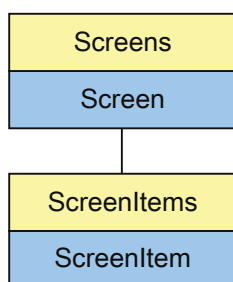
Properties	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	-	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-126 Methods

Methods	RT Profes- sional	RT Ad- vanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

ZoneQualityView (Panels, Comfort Panels, RT Advanced)

Description



Represents the "ZoneSignal" object. The ZoneQualityView object is an element of the ScreenItems list.

Type identifier in VBS

Abbreviation	Access in runtime
R	Read
RW	Read and write
-	No access

Table 1-127 Properties

Properties	RT Professional	RT Advanced	Panel RT	Description
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	-	-	-
Layer (Page 768)	-	-	RW	Specifies the layer in the screen that contains an object.
Left (Page 774)	-	-	RW	Specifies the value of the X coordinate.
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (Page 1086)	-	-	RW	Specifies the value of the Y coordinate.
Visible (Page 1208)	-	-	RW	Specifies whether the selected object is visible.
Width	-	-	-	-

Table 1-128 Methods

Methods	RT Professional	RT Advanced	Panel RT	Description
Activate (Page 1254)	-	-	Yes	Enables the permanent area or the root screen.

1.5.5 Properties (Panels, Comfort Panels, RT Advanced, RT Professional)

1.5.5.1 Properties A (Panels, Comfort Panels, RT Advanced, RT Professional)

AboveUpperLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color when the high limit is exceeded.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**AboveUpperLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- IOField

You have no access in runtime with the following format:

- GraphicIOField
- Switch
- SymbolLibrary
- SymbolicIOField

Color

Optional. A value or a constant that specifies the color for the "High limit violated" case.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

IOField (Page 332)

GraphicIOField (Page 324)

Switch (Page 420)

SymbolLibrary (Page 429)

SymbolicIOField (Page 424)

AcceptOnExit (RT Professional)

Description

Specifies whether the input field will be confirmed automatically when it is left.

Access in runtime: Read and write

Syntax

Object.**AcceptOnExit**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField
- SymbolicIOField

BOOLEAN

Optional TRUE, if the input field will be confirmed automatically when it is left.

See also

IOField (Page 332)

SymbolicIOField (Page 424)

AcceptOnFull (RT Professional)

Description

Specifies whether the input field will be left and confirmed automatically when the set number of values have been entered.

Access in runtime: Read and write

Syntax

Object.**AcceptOnFull**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if the input field will be left and confirmed automatically when the set number of values have been entered.

See also

IOField (Page 332)

AccessPath (RT Professional)**Description**

Returns the storage path of a screen.

Access during runtime: Read

Syntax

Object.**AccessPath**

Object

Necessary. A "Screen" object.

Example

In the following example, the path of the picture "ScreenWindow1" is issued:

```
'VBS67
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
MsgBox objScreen.AccessPath
```

See also

Screen (Page 215)

ActiveProject (RT Professional)**Description**

Returns the specified project.

Access in Runtime: Read

Syntax

Object.**ActiveProject**

Object

Required A "HMIRuntime" object.

See also

HMIRuntime (Page 208)

Project (Page 214)

ActiveScreen (Panels, Comfort Panels, RT Advanced)

Description

Returns an object of type "Screen" which shows the screen that currently has the focus.

Note

If you query the "ActiveScreen" property in a user-defined function, it may be due to a screen saver that the property does not return a valid "screen" object but "Nothing". A system message will be issued.

Access in runtime: Read

Syntax

Object.**ActiveScreen**

Object

Required. An object of the type "HMIRuntime".

Comments

Which screen is returned depends on whether the root screen or the permanent area has the focus.

The ActiveScreen property returns NOTHING if no screen has the focus. That is the case, for example, when another window has the focus. With the statement "If not [expression] is nothing" you are able to interrogate whether a screen is to be returned:

```
VBS example ActiveScreen
Dim objActiveScreen
Set objActiveScreen = HmiRuntime.ActiveScreen
If not objActiveScreen is nothing then
'found an active screen
HmiRuntime.Trace("There is an active screen." & vbCrLf)
Else
'found NO active screen
HmiRuntime.Trace("There is NO active screen." & vbCrLf)
End If
```

See also

HMIRuntime (Page 208)

ActiveScreen (RT Professional)

Description

Returns an object of type "Screen" which shows the screen that currently has the focus.

Note

If you query the "ActiveScreen" property in a function, it may be due to a ScreenSavers that the property does not return a valid "Screen" object but "Nothing" and a system alarm is displayed.

Access in runtime: Read

Syntax

Object.ActiveScreen

Object

Required. An object of the type "HMIRuntime".

Comments

Which screen is returned depends on whether the root screen or the permanent area has the focus.

The ActiveScreen property returns NOTHING if no screen has the focus. That is the case, for example, when another window has the focus. With the instruction "If not [printout] Is Nothing" you can query whether a screen is going to be returned:

```
'VBS_Example_ActiveScreen
Dim objActiveScreen
Set objActiveScreen = HmiRuntime.ActiveScreen
If Not objActiveScreen Is Nothing Then
'found an active screen
HmiRuntime.Trace("There is an active screen." & vbCrLf)
Else
'found NO active screen
HmiRuntime.Trace("There is NO active screen." & vbCrLf)
End If
```

See also

HMI Runtime (Page 208)

ActiveScreenItem (RT Professional)

Description

References the screen object that currently has the focus.

It is only if the screen of the respective "Screen" object is currently selected and has an input field that the "ActiveScreenItem" property of the "Screen" object will be occupied with a valid "ScreenItem" object. In all other cases if, for example, another screen from the "Screens" list, an independent window within WinCC or another application is selected, this property is not supplied on the screens, i.e. assigned "Nothing".

Application

You use the "ActiveScreenItem" object to address the properties of the object that has the focus in Runtime.

ActiveScreenItem (Panels, Comfort Panels, RT Advanced)

Description

References the screen object which currently has the focus.

The "ActiveScreenItem" property of the "Screen" object is only assigned a valid "ScreenItem" object if the screen of the corresponding "Screen" object is active and has an input field. In all other cases, for example, with a different screen from the "Screens" list, an independent window within WinCC or a different application will be selected, this property will not apply to the screens and will be occupied with "Nothing".

Application

The "ActiveScreenItem" object is used to address the object property which has the focus in runtime.

ActualPointIndex (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the number of the current corner point.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ActualPointIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Line

Int32

Optional. A value or a constant that specifies the number of the current corner point.

See also

Polygon (Page 379)

Polyline (Page 382)

TubePolyline (Page 464)

Line (Page 337)

ActualPointLeft (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the X coordinate of the current corner point with reference to the screen origin. The screen source is at the top left of the object. Every corner is identified by an index which is derived from the number ("PointCount") of the existing corners.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ActualPointLeft**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Line

Int32

Optional. A value or a constant that specifies the X coordinate of the current corner point with reference to the screen origin.

Comments

Changing the value can have an effect on the properties "Width" (object width) and "Left" (X coordinate of the object position).

See also

Polygon (Page 379)

Polyline (Page 382)

TubePolyline (Page 464)

Line (Page 337)

ActualPointTop (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the Y coordinate of the current corner point with reference to the screen origin. The screen source is at the top left of the object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ActualPointTop**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Line

Int32

Optional. A value or a constant that specifies the Y coordinate of the current corner point with reference to the screen origin.

Comments

Changing the value can have an effect on the properties "Height" (object height) and "Top" (X coordinate of the object position).

See also

Polygon (Page 379)

Polyline (Page 382)

TubePolyline (Page 464)

Line (Page 337)

AdaptBorder (RT Professional)

Description

Specifies whether the border of the object will be dynamically adapted to the text size.

Access in runtime: Read and write

Syntax

Object.**AdaptBorder**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox

- IOField
- OptionGroup
- SymbolicIOField

BOOLEAN

Optional.

TRUE, if the border of the object is dynamically adapted to the text size.

FALSE, if the border of the object is not dynamically adapted to the text size.

See also

Button (Page 266)

CheckBox (Page 275)

IOField (Page 332)

OptionGroup (Page 372)

SymbolicIOField (Page 424)

AdaptPicture (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

AdaptScreenToWindow (RT Professional)

Description

Specifies whether or not the screen displayed in a screen window adapts to the size of the screen window in runtime.

Access in runtime Read and write

Syntax

Object.**AdaptScreenToWindow**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Screenwindow

BOOLEAN

Optional.

TRUE, if the screen adapts to the screen window size.

FALSE, if the screen does not adapt to the screen window size.

See also

ScreenWindow (Page 405)

AdaptWindowtoScreen (RT Professional)

Description

Specifies whether the screen window adapts to the screen it displays in runtime.

Access in runtime Read and write

Syntax

Object.**AdaptWindowtoScreen**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Screenwindow

BOOLEAN

Optional.

TRUE, if the screen window size adapts to the screen.

FALSE, if the screen window size does not adapt to the screen.

See also

ScreenWindow (Page 405)

Address (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the web address that will be opened in the HTML browser.

Access in runtime: Read and write

Syntax

Object.**Address**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- HTMLBrowser

STRING

Optional. A value or a constant that contains the web address.

See also

HTMLBrowser (Page 330)

AddressEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

AdressPreview (RT Advanced)

Description

No access in runtime.

AdvancedButtonPositions (RT Advanced)

Description

No access in runtime.

AdvancedView (RT Advanced)

Description

No access in runtime.

Alarm (RT Professional)

Description

No access in runtime.

AlarmAreaHeight (RT Advanced)**Description**

No access in runtime.

AlarmAreaWidth (RT Advanced)**Description**

No access in runtime.

AlarmClasses (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

AlarmColor (RT Professional)**Description**

No access in runtime.

AlarmID (RT Professional)**Description**

Returns the AlarmID of the Alarm object. The AlarmID is unique and is assigned by the system.

AlarmID (readonly)

See also

Alarms (list) (Page 200)

AlarmLog (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

AlarmLogs (RT Professional)

Description

Returns an object of type "AlarmLogs".

Access in Runtime: Read

Syntax

Object.**AlarmLogs**

Object

Required A "Logging" object.

See also

Logging (Page 213)

AlarmLowerLimit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the low limit at which the alarm is triggered.

Access in runtime: Read and write

Syntax

Object.**AlarmLowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the low limit at which the alarm is triggered.

Comments

The type of evaluation (percentage or absolute) is defined using the "AlarmLowerLimitRelative" property.

The "AlarmLowerLimitEnable" property defines whether or not monitoring of this limit is enabled.

See also

Bar (Page 258)

AlarmLowerLimitColor (RT Professional)

Description

Specifies the bar color for the "AlarmLowerLimit" limit.

The "AlarmLowerLimitEnable" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**AlarmLowerLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

You have no access in runtime with the following format:

- Slider

Color

Optional. A value or a constant that specifies the bar color for the "AlarmLowerLimit" limit.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Slider (Page 410)

AlarmLowerLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the "AlarmLowerLimit" limit is monitored.

Access in runtime: Read and write

Syntax

Object.**AlarmLowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "AlarmLowerLimit" limit is monitored.

Comments

The following values will be defined via the properties "AlarmLowerLimit", "AlarmLowerLimitColor" and "AlarmLowerLimitRelative":

Limit

Representation upon reaching the limit

Type of evaluation

See also

Bar (Page 258)

AlarmLowerLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Establish whether the lower limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.

Access in runtime: Read and write

Syntax

Object.**AlarmLowerLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the lower limit at which the interrupt is triggered is evaluated as a percentage.

See also

Bar (Page 258)

AlarmSource (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

AlarmTextTag (RT Professional)**Description**

No access in runtime.

AlarmUpperLimit (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Establishes the upper limit at which the interrupt will be triggered.

Access in runtime: Read and write

Syntax

Object.**AlarmUpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that defines the upper limit at which an interrupt will be triggered.

Comments

The type of evaluation (percentage or absolute) is defined using the "AlarmUpperLimitRelative" property.

The "AlarmUpperLimitEnable" property defines whether or not monitoring of this limit is enabled.

See also

Bar (Page 258)

AlarmUpperLimitColor (RT Professional)

Description

Specifies the bar color for the "AlarmUpperLimit" limit.

The "AlarmUpperLimitEnable" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**AlarmUpperLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

You have no access in runtime with the following format:

- Slider

Color

Optional. A value or a constant that specifies the bar color for the "AlarmUpperLimit" limit.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Slider (Page 410)

AlarmUpperLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the "AlarmUpperLimit" limit is monitored.

Access in runtime: Read and write

Syntax

Object.**AlarmUpperLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "AlarmUpperLimit" limit is monitored.

Comments

The following values will be defined via the properties "AlarmUpperLimit", "AlarmUpperLimitColor" and "AlarmUpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

AlarmUpperLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Establish whether the upper limit at which the interrupt is triggered is evaluated as a percentage or an absolute value.

Access in runtime: Read and write

Syntax

Object.**AlarmUpperLimitRelative**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the high limit at which the interrupt is triggered is evaluated as a percentage.

See also

Bar (Page 258)

AllFilters (RT Professional)

Description

No access in runtime.

AllFiltersForHitlist (RT Professional)

Description

No access in runtime.

AllowEdit (RT Advanced)

Description

No access in runtime.

AllowMenu (RT Advanced)

Description

No access in runtime

AllServer (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the alarms of all available servers are displayed.

Access in runtime: Read and write

Syntax

Object.**AllServer**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl

BOOLEAN

Optional.

TRUE, if the alarms of all available servers are displayed.

FALSE, if only the alarms of the selected server are displayed.

See also

AlarmControl (Page 236)

AllTagTypesAllowed (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Analog (RT Professional)

Description

Specifies whether the clock should be shown as an analog clock.

Access in runtime: Read and write

Syntax

Object.**Analog**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Clock

BOOLEAN

Optional TRUE, if the clock should be shown as an analog clock.

See also

Clock (Page 287)

AngleMax (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the angle of the scale end of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**AngleMax**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the angle in degrees.

Comments

The start and end of the scale gradations are described with the properties "AngleMin" and "AngleMax" in angle degrees.

The value of the AngleMin property must always be less than the value of the AngleMax property. The zero degree angle is located at the 3 o'clock position on the scale. Positive angle values are counted clockwise.

See also

Gauge (Page 320)

AngleMin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the angle of the scale start of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**AngleMin**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the angle in degrees.

Comments

The start and end of the scale gradations are described with the properties "AngleMin" and "AngleMax" in angle degrees.

The value of the AngleMin property must always be less than the value of the AngleMax property.

The zero degree angle is located at the 3 o'clock position on the scale. Positive angle values are counted clockwise.

See also

Gauge (Page 320)

AnimationIgnore (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

Appearance (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ApplyProjectSettings (RT Professional)**Description**

Specifies whether the project settings are applied from the "HMI alarms" editor.

Access in runtime Read and write

Syntax

Object.**ApplyProjectSettings**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE when the alarm text blocks configured in the "HMI alarms" editor are applied in the alarm view with their properties. The alarm text blocks are displayed with these properties in the alarm view.

FALSE when the properties are not applied.

See also

AlarmControl (Page 236)

ApplyProjectSettingsForDesignMode (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ArchiveName (RT Professional)**Description**

Specifies the log name.

Access in runtime: Read and write

Syntax

Object.**ArchiveName**[=STRING]

Object

Required. A "ScreenItem" object with the characteristics:

- UserArchiveControl

STRING

Optional. A value or a constant that specifies the log name.

See also

UserArchiveControl (Page 469)

ArchiveType (RT Professional)**Description**

Sets the log type.

Access in runtime: Read and write

Syntax

Object.**ArchiveType**[=RecipeControlDataSourceType]

Object

Required. A "ScreenItem" object with the characteristics:

- UserArchiveControl

RecipeControlDataSourceType

Optional. Value or constant that specifies the log type.

Value	Designation	Explanation
0	Unknown	Specifies no log type.
1	Recipe	Specifies that the log type is a recipe.
2	Recipe query	Specifies that the log type is a recipe query.

See also

UserArchiveControl (Page 469)

AskOperationMotive (RT Professional)**Description**

Specifies whether the reason for operating this object is logged. If the object is operated in runtime, the operator enters the reason in a dialog.

Access in runtime Read and write

Syntax

Object.**AskOperationMotive**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- ComboBox
- IOField
- ListBox

- SymbolicIOField
- WindowSlider

BOOLEAN

Optional.TRUE if the reason for operating this object is to be logged.

See also

ComboBox (Page 290)

IOField (Page 332)

Listbox (Page 340)

SymbolicIOField (Page 424)

WindowSlider (Page 485)

AspectRatio (RT Professional)

Description

Specifies whether the aspect ratio of the Media Player is maintained when the size is modified.

Access in Runtime: Read and write

Syntax

Object.**AspectRatio**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the format:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the aspect ratio of the Media Player is not maintained when the size is modified.

FALSE, if the aspect ratio of the Media Player is not maintained when the size is modified.

See also

MediaPlayer (Page 343)

AssignedFilters (RT Professional)**Description**

No access in runtime.

AssignedHitlistFilters (RT Professional)**Description**

No access in runtime.

Assignments (RT Professional)**Description**

Specifies a list which contains the assignments between the output value and the output text actually to be output. The assignments are dependent on the set list type. You define the list type with the ListType property.

Access in runtime: Read and write

Syntax

Object.**Assignments**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- SymbolicIOField

STRING

Optional Specifies a list which contains the assignments between the output value and the output text actually to be output.

See also

SymbolicIOField (Page 424)

AssociatedS7GraphDBName (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

AssociatedS7GraphDBTag (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Authorization (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies operating rights.

Access in runtime: Read and write

Syntax

Object.**Authorization**[=HMIRTAuthorization]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- CheckBox*
- Circle *
- CircleSegment *
- CircularArc *
- ComboBox*
- Connector *
- DateTimeField **
- Ellipse *
- EllipseSegment *
- EllipticalArc *
- GraphicIOField
- GraphicView *
- IOField
- Line *
- ListBox *
- MultiLineEdit *
- OptionGroup *
- Polygon *

- Polyline *
- RecipeView **
- Rectangle *
- RoundButton *
- S7GraphOverview *
- Slider
- StatusForce **
- Switch **
- SymbolLibrary **
- SymbolicIOField
- TextField *
- TrendView
- TubeArcObject *
- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserView **
- WindowSlider *

* No access in RT Advanced

** No access in RT Professional

You have no access in runtime with the following format:

- AlarmView
- PLCCodeViewer
- PdfView
- ProtectedAreaNameView
- RangeLabelView
- SysDiagControl

HMIRTAuthorization

Optional. A value or a constant which specifies the operating rights.

AutoCompleteColumns (RT Professional)

Description

Specifies whether empty columns are shown if the control is wider than the configured columns.

Access in runtime Read and write

Syntax

Object.**AutoCompleteColumns**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the empty columns are shown.

FALSE if the empty columns are not shown.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AutoCompleteRows (RT Professional)

Description

Specifies whether empty rows are shown if the control is longer than the number of configured rows.

Access in runtime Read and write

Syntax

Object.**AutoCompleteRows**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the empty rows are shown.

FALSE if the empty rows are not shown.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AutoPosition (RT Professional)**Description**

Specifies whether the value table is automatically placed below the object that supplies the value table with data.

Access in Runtime: Read and write

Syntax

Object.**AutoPosition**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the object is automatically placed below the data source.

FALSE, if the object is placed at the configured position.

See also

TrendRulerControl (Page 444)

AutoScroll (RT Professional)

Description

Specifies whether autoscroll is used.

Access in runtime Read and write

Syntax

Object.**AutoScroll**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if autoscroll is used.

See also

AlarmControl (Page 236)

AutoSelectionColors (RT Professional)

Description

Specifies whether the colors defined by the system are used as the selection colors for cells and rows.

Access in runtime Read and write

Syntax

Object.**AutoSelectionColors**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the system color is used.

FALSE if the user-defined color is used.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AutoSelectionRectColor (RT Professional)**Description**

Specifies whether the selection frame is shown with the color defined by the system.

Access in runtime: Read and write

Syntax

Object.**AutoSelectionRectColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the system color is used.

FALSE if the user-defined color is used.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AutoShow (RT Professional)

Description

Specifies whether the value table is shown automatically.

Access in runtime: Read and write

Syntax

Object.**AutoShow**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the value table is shown automatically.

FALSE, if the value table is not shown automatically.

See also

TrendRulerControl (Page 444)

AutoSizing (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

AutoStart (RT Professional)

Description

No access in runtime.

AvailableStatusBarElements (RT Professional)**Description**

No access in runtime.

AvailableToolbarButtons (RT Professional)**Description**

No access in runtime.

AverageLast15Values (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the average of the last 15 values is to be shown.

Access in runtime Read and write

Syntax

Object.**AverageLast15Values**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

BOOLEAN

Optional.

TRUE if the average of the last 15 values is to be shown.

FALSE if the average of the last 15 values is not to be shown.

See also

Bar (Page 258)

AxisXBunchCount (RT Advanced)**Description**

No access in runtime.

AxisXMarkCount (RT Advanced)

Description

No access in runtime.

AxisXNoOfDigits (RT Advanced)

Description

No access in runtime.

AxisXShowBunchValues (RT Advanced)

Description

No access in runtime.

AxisXStyle (RT Advanced)

Description

No access in runtime.

AxisY1BunchCount (RT Advanced)

Description

No access in runtime.

AxisY1MarkCount (RT Advanced)

Description

No access in runtime.

AxisY1ShowBunchValues (RT Advanced)

Description

No access in runtime.

AxisY2BunchCount (RT Advanced)**Description**

No access in runtime.

AxisY2MarkCount (RT Advanced)**Description**

No access in runtime.

AxisY2ShowBunchValues (RT Advanced)**Description**

No access in runtime.

1.5.5.2 Properties B (Panels, Comfort Panels, RT Advanced, RT Professional)**BackButtonVisible (RT Advanced)****Description**

No access in runtime.

BackColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the background color.

Access in runtime: Read and write

Syntax

Object.**BackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl *
- AlarmView
- Bar

- Button **
- CheckBox *
- Circle
- CircleSegment *
- CircularArc *
- ComboBox *
- Connector *
- DateTimeField **
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl *
- Gauge
- GraphicIOField **
- GraphicView
- IOField
- Line
- ListBox *
- MultiLineEdit *
- OnlineTableControl *
- OnlineTrendControl *
- OptionGroup *
- Polygon
- Polyline
- RecipeView **
- Rectangle
- RoundButton *
- Slider
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField
- TextField
- TrendRulerControl *
- UserArchiveControl *

- UserView **
 - WindowSlider *
- * RT Advanced No access
- ** RT Professional No access

You have no access in runtime with the following format:

- TrendView
- TubeArcObject

Color

Optional. A value or a constant that specifies the background color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

The background color is not visible if the "BorderStyle" property has the value "0".

See also

AlarmControl (Page 236)
AlarmView (Page 251)
Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
ComboBox (Page 290)
Connector (Page 293)
DateTimeField (Page 296)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
FunctionTrendControl (Page 309)

Gauge (Page 320)
GraphicIOField (Page 324)
GraphicView (Page 327)
IOField (Page 332)
Line (Page 337)
Listbox (Page 340)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)
Polygon (Page 379)
Polyline (Page 382)
RecipeView (Page 390)
Rectangle (Page 396)
RoundButton (Page 399)
Slider (Page 410)
StatusForce (Page 415)
Switch (Page 420)
SymbolicIOField (Page 424)
SymbolLibrary (Page 429)
TextField (Page 440)
TrendRulerControl (Page 444)
TrendView (Page 454)
TubeArcObject (Page 460)
UserArchiveControl (Page 469)
UserView (Page 481)
WindowSlider (Page 485)

BackColorBottom (RT Professional)

Description

Specifies the color for the lower / right part of the object.

Access in runtime Read and write

Syntax

Object.**BackColorBottom**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- WindowSlider

Color

Optional. A value or a constant that specifies the color for the lower / right part of the object.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

WindowSlider (Page 485)

BackColorTop (RT Professional)

Description

Specifies the color for the upper / left part of the object.

Access in runtime Read and write

Syntax

Object.**BackColorTop**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- WindowSlider

Color

Optional. A value or a constant that specifies the color for the upper / left part of the object.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

WindowSlider (Page 485)

BackFillStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the fill pattern.

Access in runtime: Read and write

Syntax

Object.**BackFillStyle**[=FillStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar
- Button*
- CheckBox*
- Circle
- CircleSegment *
- Clock*
- ComboBox
- Ellipse
- EllipseSegment *
- Gauge*
- GraphicView*
- IOField*
- ListBox*
- OptionGroup*
- Polygon

- Rectangle
- RoundButton*
- Slider*
- SymbolLibrary*
- SymbolicIOField*
- TextField*
- WindowSlider*

* RT Advanced No access

You have no access in runtime with the following format:

- DateTimeField
- GraphicIOField
- Switch

FillStyle

Optional. A value or a constant that specifies the fill style.

Value	VB constant	Explanation
0	hmiFillStyleSolid	Object is filled with the specified color
1	hmiFillStyleTransparent	Transparent fill

See also

[Bar \(Page 258\)](#)
[Button \(Page 266\)](#)
[CheckBox \(Page 275\)](#)
[Clock \(Page 287\)](#)
[ComboBox \(Page 290\)](#)
[DateTimeField \(Page 296\)](#)
[Gauge \(Page 320\)](#)
[GraphicIOField \(Page 324\)](#)
[GraphicView \(Page 327\)](#)
[IOField \(Page 332\)](#)
[Listbox \(Page 340\)](#)
[OptionGroup \(Page 372\)](#)
[RoundButton \(Page 399\)](#)
[Slider \(Page 410\)](#)
[Switch \(Page 420\)](#)

SymbolicIOField (Page 424)

SymbolLibrary (Page 429)

WindowSlider (Page 485)

BackFlashingColorOff (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the background color for flash state "Off".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingColorOff**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following format:

- Switch

Color

Optional. A value or a constant that specifies the color of the background for the flashing status "off".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

GraphicView (Page 327)

IOField (Page 332)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicIOField (Page 424)

WindowSlider (Page 485)

BackFlashingColorOn (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the background color for flash state "On".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingColorOn**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField

- TextField
- WindowSlider

You have no access in runtime with the following format:

- Switch

Color

Optional. A value or a constant that specifies the color of the background for the flashing status "on".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
GraphicView (Page 327)
IOField (Page 332)
OptionGroup (Page 372)
RoundButton (Page 399)
Switch (Page 420)
SymbolicIOField (Page 424)
WindowSlider (Page 485)

BackFlashingEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the background flashes in runtime.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following format:

- Switch

BOOLEAN

Optional. TRUE if the background flashes in runtime.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

GraphicView (Page 327)

IOField (Page 332)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicIOField (Page 424)

WindowSlider (Page 485)

BackFlashingRate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the flash rate of the background.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BackFlashingRate**[=FlashingRate]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following format:

- Switch

FlashingRate

Optional. A value or a constant that specifies the flash rate of the background.

Value	VB constant	Explanation
0	hmiFlashingRateSlow	The length of the flashing interval is 250 ms.
1	hmiFlashingRateMedium	The length of the flashing interval is 500 ms.
2	hmiFlashingRateFast	The length of the flashing interval is 1000 ms.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

GraphicView (Page 327)

IOField (Page 332)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicOField (Page 424)

WindowSlider (Page 485)

BackgroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

See also

TrendRulerControl (Page 444)

BackPicture (RT Professional)

Description

Specifies the background graphic.

Access in runtime: Read and write

Syntax

Object.**BackPicture**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the characteristics:

- Gauge
- Slider

HmiObjectHandle

Optional. A value or a constant that specifies the background graphic.

See also

Gauge (Page 320)

Slider (Page 410)

BackStyle (RT Professional)

Description

Specifies the background style.

Access in runtime: Read and write

Syntax

Object.**BackStyle**[=Int32]

Object

Required. A "ScreenItem" object with the characteristics:

- Gauge

Int32

Optional. A value or a constant that specifies the background style.

Value	Designation	Explanation
0	Solid	The rectangular background of the display is filled with the specified border color. The scale disk is filled with the specified background color.
1	Transparent border	The rectangular background of the dial gauge is transparent. The scale disk is filled with the specified background color. As a result, a circular display can be shown.
2	Transparent	The rectangular background and the scale disk are transparent.

See also

Gauge (Page 320)

BarBackColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the bar background for the selected object.

Access in runtime Read and write

Syntax

Object.**BarBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar
- Slider

Color

Optional. A value or a constant that specifies the color of the bar background.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Slider (Page 410)

BarBackFillStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the fill style for the bar.

Access in runtime Read and write

Syntax

Object.**BarBackFillStyle**[=FillStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

FillStyle

Optional. A value or a constant that specifies the fill pattern of the bar.

Value	VB constant	Explanation
65536	<code>hmiFillStyleTransparent</code>	Transparent fill pattern
0	<code>hmiFillStyleSolid</code>	Solid fill pattern
131075	<code>hmiFillStyleBackwardDiagonal</code>	Diagonal fill pattern striped towards the top right
131076	<code>hmiFillStyleCross</code>	Checked fill pattern
131077	<code>hmiFillStyleDiagonalCross</code>	Diagonal checked fill pattern
131074	<code>hmiFillStyleForwardDiagonal</code>	Diagonal fill pattern striped towards the top left
131072	<code>hmiFillStyleHorizontal</code>	Horizontal striped fill pattern
131073	<code>hmiFillStyleVertical</code>	Vertical striped fill pattern
196608 - 196644	<code>hmiFillStylePattern1 - hmiFillStylePattern37</code>	Predefined fill pattern

See also

Bar (Page 258)

BarBackFlashingColorOff (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color ###

Access in Runtime: Read and write

Syntax

Object.**BarBackFlashingColorOff**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

Color

Optional. A value or a constant that specifies the color of ###

BarBackFlashingColorOn (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color ###

Access in Runtime: Read and write

Syntax

Object.**BarBackFlashingColorOn**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

Color

Optional. A value or a constant that specifies the color of ###

BarBackFlashingEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether ###

Access in Runtime: Read and write

Syntax

Object.**BarBackFlashingEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

BOOLEAN

Optional. TRUE, if ###

BarBackFlashingRate (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Sets the flash rate ###

Access in Runtime: Read and write

Syntax

Object.**BarBackFlashingRate**[=FlashingRate]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

FlashingRate

Optional. A value or a constant that specifies the flash rate ###

BarColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color of the bar.

Access in runtime Read and write

Syntax

Object.**BarColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Slider

Color

Optional. A value or a constant that specifies the color of the slider.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

The range extends from the value in "MinimumValue (Page 826)" to the position of the slider.

See also

Slider (Page 410)

BarEdgeStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies ###

Access in Runtime:

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**BarEdgeStyle**[=LineStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

LineStyle

Optional. A value or a constant, which ###

BarOrientation (RT Professional)**Description**

Specifies the bar alignment.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BarOrientation**[=BarOrientation]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

You have no access in runtime with the following format:

- Slider

BarOrientation

Optional. A value or a constant that specifies the bar direction.

Value	Designation	Description
0	Top	The bar is aligned vertically. The maximum scale value is located at the top edge of the bar.
1	Bottom	The bar is aligned vertically. The maximum scale value is located at the bottom edge of the bar.
2	Left	The bar is aligned horizontally. The maximum scale value is arranged on the left.
3	Right	The bar is aligned horizontally. The maximum scale value is arranged on the right.

See also

Bar (Page 258)

Slider (Page 410)

BaseScreenName (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Reads the name of the current root screen or triggers a root screen change by setting a new screen name.

Access in Runtime: Read and write

Syntax

Object.**BaseScreenName**[= STRING]

Object

Required An "HMIRuntime" object.

STRING

Optional A value or a constant that contains the screen name.

Comments

You can also use the property to establish which screen is currently being displayed.

See also

HMIRuntime (Page 208)

BelowLowerLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the "Low limit violated" event.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BelowLowerLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- IOField

You have no access in runtime with the following format:

- GraphicIOField
- Switch
- SymbolLibrary
- SymbolicIOField

Color

Optional. A value or a constant which specifies the color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

BitNumber (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the bit whose status must change to trigger a value change.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BitNumber**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- SymbolicIOField

You have no access in runtime with the following format:

- Button
- GraphicIOField

Int32

Optional Specifies the bit whose status must change to trigger a value change.

Comments

The tag being used must be of type BYTE, WORD or DWORD.

See also

SymbolicIOField (Page 424)

Button (Page 266)

GraphicIOField (Page 324)

BlinkColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color in which the object flashes in runtime.

Access in runtime Read and write

Syntax

Object.**BlinkColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolLibrary

Color

Optional. A value or a constant that specifies the flash color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

SymbolLibrary (Page 429)

BlinkMode (RT Professional)

Description

Specifies the type of flash graphic for the specified object.

Access in runtime Read and write

Syntax

Object.**BlinkMode**[=SymbolLibraryBlinkMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolLibrary

SymbolLibraryBlinkMode

Optional. A value or a constant that specifies the type of flash picture for the specified object.

Value	VB constant	Explanation
0	hmiSymbolLibraryFlashing-None	Flashing is switched off.
1	hmiSymbolLibraryFlashing-Invisible	The flash picture is invisible.
2	hmiSymbolLibraryFlashing-Shaded	The flash picture is given a colored, shaded surface. The color of the surface corresponds to the settings in the property "BlinkColor".
3	hmiSymbolLibraryFlashing-Solid	The flash picture is given a colored, unshaded surface. The color of the surface corresponds to the settings in the property "BlinkColor".

See also

SymbolLibrary (Page 429)

BlinkSpeed (RT Professional)**Description**

Sets the flash rate.

Fast - 250: The length of the flashing interval is 250 ms. Medium - 500: The length of the flashing interval is 500 ms.

Slow - 1000: The length of the flashing interval is 1000 ms. The default value is medium - 500.

Access in runtime Read and write

Syntax

Object.**BlinkSpeed**[=FlashingRate]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolLibrary

FlashingRate

Optional. A value or a constant that specifies the flash rate.

Value	VB constant	Explanation
0	hmiFlashingRateSlow	The length of the flashing interval is 250 ms.
1	hmiFlashingRateMedium	The length of the flashing interval is 500 ms.
2	hmiFlashingRateFast	The length of the flashing interval is 1000 ms.

See also

SymbolLibrary (Page 429)

BlockAlignment (RT Professional)**Description**

Specifies the alignment of the text within the column header of the referenced block.

Access in runtime Read and write

Syntax

Object.**BlockAlignment**[=HorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the format:

- TrendRulerControl

HorizontalAlignment

Optional. A value or constant that determines the horizontal alignment.

Value	VB constant	Description
0	hmiAlignmentLeft	The text in the column header is left-justified.
1	hmiAlignmentCentered	The text in the column header is centered.
2	hmiAlignmentRight	The text in the column header is right-justified.

See also

TrendRulerControl (Page 444)

BlockAutoPrecision (RT Professional)**Description**

Specifies whether the number of decimal places displayed in the current block is adjusted automatically.

Access in runtime: Read and write

Syntax

Object.**BlockAutoPrecisions**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the number of displayed decimal places is adapted automatically.

FALSE, if the number of displayed decimal places is not adapted automatically.

See also

TrendRulerControl (Page 444)

BlockCaption (RT Professional)**Description**

Specifies the header of the current block.

Access in runtime: Read and write

Syntax

Object.**BlockCaption**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

STRING

Optional. A value or a constant that specifies the header of the block.

See also

TrendRulerControl (Page 444)

BlockCount (RT Professional)**Description**

Specifies the number of blocks.

Access in runtime: Read and write

Syntax

Object.**BlockCount**[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

Int32

Optional. A value or a constant that specifies the number of blocks.

See also

TrendRulerControl (Page 444)

BlockDateFormat (RT Professional)

Description

Specifies the format for date information in the current block.

Access in runtime: Read and write

Syntax

Object.**BlockDateFormat**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

STRING

Optional. A value or a constant that specifies the format for date information.

See also

TrendRulerControl (Page 444)

BlockExponentialFormat (RT Professional)

Description

Specifies whether the values in the current block are displayed in exponential notation.

Access in runtime: Read and write

Syntax

Object.**BlockExponentialFormat**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the values are displayed in exponential notation.

FALSE, if the values are displayed in decimal notation.

See also

TrendRulerControl (Page 444)

BlockHideText (RT Professional)

Description

No access in runtime.

BlockHideTitleText (RT Professional)

Description

Specifies whether the header of the block is displayed as text.

Access in runtime: Read and write

Syntax

Object.**BlockHideTitleText**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the header of the block is displayed as text.

FALSE, if the header of the block is not displayed as text.

See also

TrendRulerControl (Page 444)

BlockId (RT Professional)

Description

References a block with its block ID. To access the properties of a block, you need to set "BlockId".

Access in runtime Read and write

Syntax

Object.**BlockId**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl

Int32

Optional. A value or a constant that specifies the ID of the current block.

See also

TrendRulerControl (Page 444)

BlockIndex (RT Professional)

Description

References a block. To access the properties of a block, you need to set "BlockIndex".

Values between 0 and "BlockIndex" minus 1 are valid for "BlockCount". The "BlockCount" property specifies the number of configured blocks.

Access in runtime Read and write

Syntax

Object.**BlockIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl

Int32

Optional. A value or a constant that specifies the number of the current block.

See also

TrendRulerControl (Page 444)

BlockLength (RT Professional)**Description**

Specifies the number of characters of the current block.

Access in runtime: Read and write

Syntax

Object.**BlockLength**[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

Int32

Optional. A value or a constant that specifies the number of characters.

See also

TrendRulerControl (Page 444)

BlockName (RT Professional)**Description**

Specifies the name of the current block.

Access in runtime: Read and write

Syntax

Object.**BlockName**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

STRING

Optional. A value or a constant that specifies the name of the block.

See also

TrendRulerControl (Page 444)

BlockPrecision (RT Professional)

Description

Specifies the number of decimal places in the current block.

Access in runtime: Read and write

Syntax

Object.**BlockPrecisions**[=Int16]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

Int16

Optional. A value or a constant that specifies the number of decimal places.

See also

TrendRulerControl (Page 444)

Blocks (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BlockShowDate (RT Professional)

Description

Specifies that the date is displayed in the current block.

Access in runtime: Read and write

Syntax

Object.**BlockShowDate**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the date is displayed in the current block.

FALSE, if the date is not displayed in the current block.

See also

TrendRulerControl (Page 444)

BlockShowIcon (RT Professional)

Description

Specifies whether the content of the current block is displayed as an icon.

Access in runtime: Read and write

Syntax

Object.**BlockShowIcon**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the content of the current block is displayed as an icon.

FALSE, if the content of the current block is not displayed as an icon.

See also

TrendRulerControl (Page 444)

BlockShowTitleIcon (RT Professional)

Description

Specifies whether the header of the current block is displayed as an icon.

Access in runtime: Read and write

Syntax

Object.**BlockShowTitleIcon**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the header of the current block is displayed as an icon.

FALSE, if the header of the current block is not displayed as an icon.

See also

TrendRulerControl (Page 444)

BlockTimeFormat (RT Professional)

Description

Specifies the format of the time information in the current block.

Access in runtime: Read and write

Syntax

Object.**BlockTimeFormat**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

STRING

A value or a constant that specifies the format of the time information.

See also

TrendRulerControl (Page 444)

BlockUseSourceFormat (RT Professional)**Description**

Specifies whether the format of the connected control is used in the current block.

Access in runtime: Read and write

Syntax

Object.**BlockUseSourceFormat**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the format of the connected control is used in the current block.

FALSE, if the configured format is used in the current block.

See also

TrendRulerControl (Page 444)

BorderBackColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the background color of the broken border line.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- ComboBox
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- Pölygon
- Rectangle
- RoundButton
- TextField
- WindowSlider

You have no access in runtime with the following format:

- AlarmView
- Clock
- DateTimeField
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SymbolicIOField
- SysDiagControl
- TrendView
- UserView

Color

Optional. A value or a constant that specifies the background color of the broken border line.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmView (Page 251)
Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
Clock (Page 287)
ComboBox (Page 290)
DateTimeField (Page 296)
GraphicIOField (Page 324)
GraphicView (Page 327)
Gauge (Page 320)
IOField (Page 332)
Listbox (Page 340)
MultiLineEdit (Page 345)
OptionGroup (Page 372)
RecipeView (Page 390)
RoundButton (Page 399)
Slider (Page 410)
StatusForce (Page 415)
Switch (Page 420)
SymbolicIOField (Page 424)
SysDiagControl (Page 431)
TextField (Page 440)
TrendView (Page 454)
UserView (Page 481)
WindowSlider (Page 485)

BorderBrightColor3D (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the border color of the following border parts for 3D presentation:

- External frame parts at the top and bottom
- Internal frame parts at the top and right

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderBrightColor3D**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- RoundButton
- Slider*

* RT Advanced read and write access

You have no access in runtime with the following format:

- Switch

Color

Optional. A value or a constant that specifies the frame color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Button (Page 266)

RoundButton (Page 399)

Slider (Page 410)

Switch (Page 420)

BorderColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the line color.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle*
- CircleSegment
- ComboBox
- DateTimeField**
- Ellipse*
- EllipseSegment
- FunctionTrendControl
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- Polygon*
- Rectangle*
- RoundButton
- SymbolicIOField*
- TextField*

- TrendRulerControl
 - UserArchiveControl
 - WindowSlider
- * RT Advanced read and write access
- ** RT Advanced read and write access, RT Professional no access

You have no access in runtime with the following format:

- AlarmControl
- AlarmView
- Clock
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

Color

Optional. A value or a constant that specifies the line color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

AlarmView (Page 251)

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

Circle (Page 279)

CircleSegment (Page 282)
Clock (Page 287)
ComboBox (Page 290)
DateTimeField (Page 296)
Ellipse (Page 300)
EllipseSegment (Page 303)
FunctionTrendControl (Page 309)
Gauge (Page 320)
GraphicIOField (Page 324)
GraphicView (Page 327)
IOField (Page 332)
Listbox (Page 340)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)
Polygon (Page 379)
RecipeView (Page 390)
Rectangle (Page 396)
RoundButton (Page 399)
Slider (Page 410)
StatusForce (Page 415)
Switch (Page 420)
SymbolicIOField (Page 424)
SysDiagControl (Page 431)
TextField (Page 440)
TrendRulerControl (Page 444)
TrendView (Page 454)
UserArchiveControl (Page 469)
UserView (Page 481)
WindowSlider (Page 485)

BorderEnabled (RT Professional)

Description

Returns whether the window is displayed with a border in runtime.

Access in runtime: Read and write

Syntax

Object.**BorderEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

Optional. TRUE, if the window is displayed with a border in runtime.

See also

ApplicationWindow (Page 256)

ScreenWindow (Page 405)

BorderEndStyle (RT Professional)

Description

Specifies the type of line ends.

Access in runtime: Read and write

Syntax

Object.**BorderEndStyle**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector
- Line
- Polyline

Int32

Optional. A value or a constant that specifies the type of line end.

See also

Connector (Page 293)

Line (Page 337)

Polyline (Page 382)

BorderFlashingColorOff (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the border line for the flashing status "Off".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderFlashingColorOff**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following format:

- Switch

Color

Optional. A value or a constant that specifies the color of the border line for the flashing status "Off".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

GraphicIOField (Page 324)

IOField (Page 332)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicIOField (Page 424)

TextField (Page 440)

WindowSlider (Page 485)

BorderFlashingColorOn (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the border line for the flashing status "On".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderFlashingColorOn**[=Color]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following format:

- Switch

Color

Optional. A value or a constant that specifies the color of the border line for the flashing status "On".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)
GraphicIOField (Page 324)
IOField (Page 332)
OptionGroup (Page 372)
RoundButton (Page 399)
Switch (Page 420)
SymbolicIOField (Page 424)
WindowSlider (Page 485)

BorderFlashingEnabled (RT Professional)

Description

Specifies whether the limit of the object can flash in runtime.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderFlashingEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon
- Rectangle

- RoundButton
- SymbolicIOField
- TextField

You have no access in runtime with the following format:

- Switch
- WindowSlider

BOOLEAN

Optional.

TRUE if the object limit can flash in runtime.

FALSE if the object limit cannot flash in runtime.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

GraphicIOField (Page 324)

IOField (Page 332)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicIOField (Page 424)

WindowSlider (Page 485)

BorderFlashingRate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the flash rate of the border line.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderFlashingRate**[=FlashingRate]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following format:

- GraphicIOField
- Switch

FlashingRate

Optional. A value or a constant that specifies the flash rate of the border line.

Value	VB constant	Explanation
0	hmiFlashingRateSlow	The length of the flashing interval is 250 ms.
1	hmiFlashingRateMedium	The length of the flashing interval is 500 ms.
2	hmiFlashingRateFast	The length of the flashing interval is 1000 ms.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

GraphicIOField (Page 324)

IOField (Page 332)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicOField (Page 424)

WindowSlider (Page 485)

BorderInnerStyle3D (RT Professional)

Description

Specifies the appearance of the inner part of the object border.

Access in runtime Read and write

Syntax

Object.**BorderInnerStyle3D**[=3DStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge
- Slider

3DStyle

Optional. A value or a constant that defines how the inside part of the object border is displayed.

Value	VB constant	Explanation
0	hmiBorder3DStyleNone	There is no inside part of the object border.
1	hmiBorder3DStyleRecessed	The object border is shown in relief.
2	hmiBorder3DStyleRaised	The object border is shown raised.
3	hmiBorder3DStyleGray	The object border has a uniform gray border.
4	hmiBorder3DStyleColored	The object border has a uniform colored border. The border color corresponds to the background color.

See also

Gauge (Page 320)

Slider (Page 410)

BorderInnerWidth3D (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the width of the inner border for 3D display.

Access in runtime Read and write

Syntax

Object.**BorderInnerWidth3D**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Slider

Int32

Optional. A value or a constant that specifies the width of the inner border in pixels.

See also

Slider (Page 410)

BorderOuterStyle3D (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the appearance of the outer part of the object border.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderOuterStyle3D**[=3DStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

You have no access in runtime with the following format:

- Slider

3DStyle

Optional. A value or a constant that defines how the outside part of the object border is displayed.

Value	VB constant	Explanation
0	hmiBorder3DStyleNone	There is no inside part of the object border.
1	hmiBorder3DStyleRecessed	The object border is shown in relief.
2	hmiBorder3DStyleRaised	The object border is shown raised.
3	hmiBorder3DStyleGray	The object border has a uniform gray border.
4	hmiBorder3DStyleColored	The object border has a uniform colored border. The border color corresponds to the background color.

See also

Gauge (Page 320)

Slider (Page 410)

BorderOuterWidth3D (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the width of the outer border for 3D display.

Access in runtime Read and write

Syntax

Object.**BorderOuterWidth3D**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Slider

Int32

Optional. A value or a constant that specifies the width of the outer border in pixels.

See also

Slider (Page 410)

BorderShadeColor3D (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the border color of the following border parts for 3D presentation:

- Internal frame parts at the top and bottom
- External frame parts at the bottom and right

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderShadeColor3D**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- Slider*

* RT Advanced read and write access

You have no access in runtime with the following format:

- RoundButton
- Switch

Color

Optional. A value or a constant that specifies the color of the shading.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Button (Page 266)

RoundButton (Page 399)

Slider (Page 410)

Switch (Page 420)

BorderStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the type of border lines.

Access in runtime: Read and write

Syntax

Object.**BorderStyle**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField

- TextField
- WindowSlider

You have no access in runtime with the following format:

- OptionGroup

Int32

Optional. A value or a constant that specifies the type of border line.

Value	Description
0	Solid line
1	Dashed line
2	Dotted line
3	Dashed and dotted line
4	Dash-dot-dot line

See also

Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
ComboBox (Page 290)
GraphicIOField (Page 324)
IOField (Page 332)
Listbox (Page 340)
MultiLineEdit (Page 345)
OptionGroup (Page 372)
RoundButton (Page 399)
SymbolicIOField (Page 424)
WindowSlider (Page 485)

BorderWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the line thickness.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Ala rmControl
- Bar
- Button
- Circle *
- CircleSegment
- CheckBox
- ComboBox
- Ellipse *
- EllipseSegment
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- Polygon *
- Rectangle *
- OptionGroup
- RoundButton
- Slider *
- SymbolicIOField *
- TextField *
- TrendRulerControl
- UserArchiveControl
- WindowSlider

* RT Advanced read and write access

You have no access in runtime with the following format:

- AlarmView
- Clock
- DateTimeField
- RecipeView
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

Int32

Optional. A value or a constant that specifies the line weight in pixels.

See also

AlarmControl (Page 236)
AlarmView (Page 251)
Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
Clock (Page 287)
ComboBox (Page 290)
DateTimeField (Page 296)
FunctionTrendControl (Page 309)
Gauge (Page 320)
GraphicIOField (Page 324)
IOField (Page 332)
Listbox (Page 340)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)
RecipeView (Page 390)
RoundButton (Page 399)
Slider (Page 410)
StatusForce (Page 415)

Switch (Page 420)
SymbolicOField (Page 424)
SysDiagControl (Page 431)
TrendRulerControl (Page 444)
TrendView (Page 454)
UserArchiveControl (Page 469)
UIView (Page 481)
WindowSlider (Page 485)

BorderWidth3D (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the width of the inner border for 3D display.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**BorderWidth3D**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- Gauge
- RoundButton

You have no access in runtime with the following format:

- Bar
- IOField
- Switch

Int32

Optional. A value or constant that determines the width of the inside border for 3D display in pixels.

See also

Bar (Page 258)
Button (Page 266)

Gauge (Page 320)

IOField (Page 332)

RoundButton (Page 399)

Switch (Page 420)

BottomMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Bounds (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BrowserTypeUsed (RT Professional)

Description

No access in runtime.

BufferViewColumnOrder (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BufferViewInternalRowOrder (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BusyText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ButtonBackColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ButtonBackFillStyle (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ButtonBarElements (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ButtonBarHeight (RT Advanced)**Description**

No access in runtime.

ButtonBarStyle (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ButtonBorderBackColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ButtonBorderColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ButtonBorderWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ButtonCornerRadius (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ButtonEdgeStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ButtonFirstGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ButtonFirstGradientOffset (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ButtonMiddleGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ButtonPositions (RT Advanced)

Description

No access in runtime.

ButtonSecondGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ButtonSecondGradientOffset (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ColumnWidth_Date (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ColumnWidth_Event (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ColumnWidth_EventSeverity (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ColumnWidth_EventState (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ColumnWidth_Number (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ColumnWidth_Time (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BV_ItemText_Date (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BV_ItemText_Event (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BV_ItemText_EventSeverity (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BV_ItemText_EventState (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BV_ItemText_Number (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BV_ItemText_Time (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

BV_ShowItem_Date (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ShowItem_Event (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ShowItem_EventSeverity (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ShowItem_EventState (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ShowItem_Number (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

BV_ShowItem_Time (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

1.5.5.3 Properties C (Panels, Comfort Panels, RT Advanced, RT Professional)

CameraUrl (RT Professional)

Description

No access in runtime.

CanBeGrouped (RT Professional)

Description

No access in runtime.

Caption (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the text to be displayed in the header.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**Caption**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTrendControl
- Slider *
- TrendRulerControl
- UserArchiveControl

* RT Advanced read and write access

You have no access in runtime with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that contains the text that will be shown in the title line.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

Slider (Page 410)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

CaptionBackColor (RT Professional)**Description**

Specifies the background color of the title bar.

Access in runtime Read and write

Syntax

Object.**CaptionBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicIOField

Color

Optional. A value or a constant that specifies the background color of the title bar.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

SymbolicIOField (Page 424)

CaptionColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the text to be displayed in the title bar.

Access in runtime Read and write

Syntax

Object.**CaptionColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Gauge
- Switch
- SymbolicIOField

Color

Optional. A value or a constant that specifies the text color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

Switch (Page 420)

SymbolicIOField (Page 424)

CaptionFont (RT Professional)

Description

Sets the font.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**CaptionFont**[=Font]

Object

Required. A "ScreenItem" object with the characteristics:

- Gauge

You have no access in runtime in the case of the following characteristic:

- Switch

Font

Optional. A value or a constant that specifies the font.

See also

Gauge (Page 320)

Switch (Page 420)

CaptionText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the text to be displayed in the header.

Access in runtime Read and write

Syntax

Object.**CaptionText**[= STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- Gauge
- ScreenWindow
- Switch

STRING

Optional. A value or a constant that contains the text that will be shown in the title line.

See also

Gauge (Page 320)

ScreenWindow (Page 405)

Switch (Page 420)

CaptionTop (RT Professional)

Description

Specifies the distance between the instrument labels and the upper edge of the object. You can only position the instrument labels in line with the vertical diameter of the scale. The value of the property refers to the height of the object. The height specifies the upper edge of the object and the lower edge of the lettering.

Access in runtime Read and write

Syntax

Object.**CaptionTop**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

DOUBLE

Optional. A value or a constant that specifies the distance between the instrument labels and the upper edge of the object.

Value	Description
0	The lower edge of the lettering is positioned at the upper limit of the object. The text is no longer visible as it is positioned outside the object.
1	The lower edge of the lettering is positioned on the lower limit of the selected object.

See also

Gauge (Page 320)

CellCut (RT Professional)

Description

Specifies whether the contents of the cells are abbreviated if the cells are too narrow.

Access in runtime Read and write

Syntax

Object.**CellCut**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl

You have no access in runtime with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if the contents are abbreviated.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

CellSpaceBottom (RT Professional)**Description**

Specifies the bottom margin of the table cells.

Access in runtime: Read and write

Syntax

Object.**CellSpaceBottom**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or constant that defines the bottom margin used in the table cells.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

CellSpaceLeft (RT Professional)

Description

Specifies the left indent used in the table cells.
Access in runtime: Read and write

Syntax

Object.**CellSpaceLeft**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or parameter that defines the left indent in the table cells.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

CellSpaceRight (RT Professional)

Description

Specifies the right indent used in the table cells.
Access in runtime: Read and write

Syntax

Object.**CellSpaceRight**[=Int32]

Object

Required. A "ScreenItem" object with the following formats:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that defines the right indent used in the table cells.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

CellSpaceTop (RT Professional)

Description

Defines the top margin of the table cells.

Access in runtime: Read and write

Syntax

Object.**CellSpaceTop**[=Int32]

Object

Required. A "ScreenItem" object with the following formats:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or constant that defines the top margin used in the table cells.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

CenterColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the center point.
Access in runtime Read and write

Syntax

Object.**CenterColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the color of the center.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

CenterSize (RT Professional)

Description

Specifies the diameter of the scale center point.
Access in runtime: Read and write

Syntax

Object.**CenterSize**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that specifies the diameter of the round scale center point.

Value range from 0.03 to 1

1: The diameter corresponds to the smaller value of the geometry attribute "Width" or "Height".

See also

Gauge (Page 320)

ChangeMouseCursor (RT Professional)

Description

Specifies how the appearance of the cursor changes in runtime when it is above the icon.

Access in runtime Read and write

Syntax

Object.**ChangeMouseCursor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolLibrary

BOOLEAN

Optional.

TRUE, if the mouse cursor has the appearance of an arrow even when it is positioned above the icon.

FALSE, if the mouse cursor has the appearance of a 3D arrow with a green lightning symbol. This indicates in Runtime that the respective objective can be operated.

See also

SymbolLibrary (Page 429)

CheckMarkAlignment (RT Professional)

Description

Specifies whether the fields are right-aligned.

Access in runtime: Read and write

Syntax

Object.**CheckMarkAlignment**[=CheckMarkAlignment]

Object

Required. An object of the type "ScreenItem" with the format:

- CheckBox
- OptionGroup

CheckMarkAlignment

Optional. A value or a constant that specifies whether the fields are right-aligned.

Value	Explanation
0	The fields are arranged left aligned.
1	The fields are right-aligned.

See also

OptionGroup (Page 372)

CheckBox (Page 275)

CheckMarkCount (RT Professional)

Description

Specifies the number of fields.

Access in runtime: Read and write

Syntax

Object.**CheckMarkCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- CheckBox
- OptionGroup

Int32

Optional. A value or a constant that specifies the number of fields. Value range from 0 to 31

See also

OptionGroup (Page 372)

CheckBox (Page 275)

ClearOnError (RT Professional)**Description**

Specifies whether an invalid input in this object will be deleted automatically.

Access in runtime: Read and write

Syntax

Object.**ClearOnError**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if an invalid input in this object will be deleted automatically.

See also

IOField (Page 332)

ClearOnFocus (RT Professional)**Description**

Specifies whether the field entry will be deleted as soon as the I/O field is activated.

Access in runtime: Read and write

Syntax

Object.**ClearOnFocus**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if the field entry is deleted as soon as the I/O field is activated.

See also

IOField (Page 332)

Closeable (RT Professional)

Description

Specifies whether the object can be closed in runtime.

Access in runtime: Read and write

Syntax

Object.**Closeable**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional. TRUE if the object can be closed in runtime.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

Color (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the line color.

Access in runtime: Read and write

Syntax

Object.**Color**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- CircularArc
- Connector
- EllipticalArc
- Line
- Polyline
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline

Color

Optional. A value or a constant that specifies the line color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

CircularArc (Page 285)

Connector (Page 293)

EllipticalArc (Page 306)
Line (Page 337)
Polyline (Page 382)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)

ColorChangeHysteresis (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies hysteresis as a percentage of the display value.

The "ColorChangeHysteresisEnable" property must have the value TRUE so that the hysteresis can be calculated.

Access in runtime: Read and write

Syntax

Object.**ColorChangeHysteresis**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the hysteresis as a percentage of the display value.

See also

Bar (Page 258)

ColorChangeHysteresisEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Determines whether the object will be displayed with a hysteresis.

Access in runtime: Read and write

Syntax

Object.ColorChangeHysteresisEnabled[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the object will be displayed with a hysteresis.

See also

Bar (Page 258)

ColumnAdd (RT Professional)

Description

Creates a new column. The newly created column is automatically referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.ColumnAdd[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- TrendRulerControl

STRING

Required. A value or a constant that specifies the name of a new column.

See also

ColumnIndex (Page 613)

TrendRulerControl (Page 444)

ColumnAlias (RT Professional)

Description

Returns the display name of the column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnAlias**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- UserArchiveControl

STRING

Optional. A value or a constant that specifies the display name of the column.

See also

UserArchiveControl (Page 469)

ColumnAlignment (RT Professional)

Description

Specifies the alignment of the column content in a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnAlignment**[=HorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the format:

- UserArchiveControl

HorizontalAlignment

Optional. A value or constant that determines the horizontal alignment.

Value	Designation	Description
0	Left	The column text is left-justified.
1	Centered	The column text is centered.
2	Right	The column text is right-justified.

See also

UserArchiveControl (Page 469)

ColumnAutoPrecisions (RT Professional)

Description

Specifies whether the number of decimal places for a column referenced with "ColumnIndex" is set automatically.

Access in runtime Read and write

Syntax

Object.**ColumnAutoPrecisions**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE if the number of decimal places is automatically determined.

FALSE if the value of "ColumnPrecisions" is used.

See also

UserArchiveControl (Page 469)

ColumnCaption (RT Professional)

Description

Specifies the header of the current column.

Access in runtime: Read and write

Syntax

Object.**ColumnCaption**[=STRING]

Object

Required. A "ScreenItem" object with the characteristics:

- UserArchiveControl

STRING

Optional. A value or a constant that specifies the header of the current column.

See also

UserArchiveControl (Page 469)

ColumnCount (RT Professional)

Description

Specifies the number of configured columns.

Access in runtime Read and write

Syntax

Object.**ColumnCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the number of configured columns.

See also

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ColumnDateFormat (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the date format.

Access in runtime Read and write

Syntax

Object.**ColumnDateFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- UserArchiveControl

STRING

Optional. A value or a constant that specifies the date format.

See also

UserArchiveControl (Page 469)

ColumnDMVarName (RT Professional)**Description**

Specifies the name of a tag.

Access in runtime: Read and write

Syntax

Object.**ColumnDMVarName**[=STRING]

Object

Required. A "ScreenItem" object with the characteristics:

- UserArchiveControl

STRING

Optional. A value or a constant that specifies the name of a tag.

See also

UserArchiveControl (Page 469)

ColumnExponentialFormat (RT Professional)**Description**

Specifies whether the values of a column referenced with "ColumnIndex" are displayed in exponential notation.

Access in runtime Read and write

Syntax

Object.**ColumnExponentialFormat**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE if the values are displayed in exponential notation.

FALSE if the values are not displayed in exponential notation.

See also

ColumnIndex (Page 613)

UserArchiveControl (Page 469)

ColumnFlagNotNull (RT Professional)

Description

Specifies whether the user archive field assigned to a column referenced with "ColumnIndex" has to have a value.

Access in runtime Read and write

Syntax

Object.**ColumnFlagNotNull**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if a value is required.

FALSE, if no value is required.

See also

UserArchiveControl (Page 469)

ColumnFlagUnique (RT Professional)

Description

Specifies whether the user archive field assigned to a column referenced with "ColumnIndex" has to have a unique value. The values in this column must differ from one another.

Access in runtime Read and write

Syntax

Object.**ColumnFlagUnique**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if the value must be unique.

FALSE, if the value does not have to be unique.

See also

UserArchiveControl (Page 469)

ColumnHideText (RT Professional)

Description

Specifies whether the text in a column referenced with "ColumnIndex" is hidden.

Access in runtime Read and write

Syntax

Object.**ColumnHideText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if the text in the current column is hidden.

FALSE, if the text in the current column is shown.

See also

UserArchiveControl (Page 469)

ColumnHideTitleText (RT Professional)

Description

Specifies whether the header of a column referenced with "ColumnIndex" is hidden.

Access in runtime Read and write

Syntax

Object.**ColumnHideTitleText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE if the header of the referenced column is hidden.

FALSE if the header of the referenced column is displayed.

See also

UserArchiveControl (Page 469)

ColumnIndex (RT Professional)

Description

References a column. To access the properties of a column, you need to set ""ColumnIndex"".

Values between 0 and "ColumnIndex minus 1 are valid for "ColumnCount". The "ColumnCount" property specifies the number of configured columns.

Access in runtime Read and write

Syntax

Object.**ColumnIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the column to be edited over the index.

See also

ColumnCount (Page 608)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ColumnLeadingZeros (RT Professional)**Description**

Specifies how many leading zeros are displayed for the values of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnLeadingZeros**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that specifies the number of leading zeros. The maximum number is 11.

See also

UserArchiveControl (Page 469)

ColumnLength (RT Professional)**Description**

Specifies the number of characters displayed in a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnLength**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that specifies the number of characters displayed.

See also

UserArchiveControl (Page 469)

ColumnMaxValue (RT Professional)

Description

Returns the maximum value defined in the user archive of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnMaxValue**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

STRING

Optional. A value or a constant that returns the maximum value of the column specified in the user log.

See also

UserArchiveControl (Page 469)

ColumnMinValue (RT Professional)

Description

Returns the minimum value defined in the user archive of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnMinValue**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

STRING

Optional. A value or a constant that returns the minimum value of the column specified in the user log.

See also

UserArchiveControl (Page 469)

ColumnName (RT Professional)

Description

Specifies the name of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that returns the name of the referenced column.

See also

ColumnIndex (Page 612)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ColumnOrder (RT Advanced)

Description

No access in runtime.

ColumnPosition (RT Professional)

Description

Specifies the position of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnPosition**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that specifies the position of the column.

See also

UserArchiveControl (Page 469)

ColumnPrecisions (RT Professional)

Description

Specifies the number of decimal places in a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnPrecisions**[=Int16]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int16

Optional. A value or a constant that specifies the number of decimal places in the column.

See also

UserArchiveControl (Page 469)

ColumnReadAccess (RT Professional)**Description**

Returns the read access authorization defined in the user archive for a column referenced with "ColumnIndex".

The number corresponds to the number that is assigned to the authorization in the "User Administrator" editor.

Access in runtime Read and write

Syntax

Object.**ColumnReadAccess**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that returns the authorization for read access specified in the user log to the column.

See also

UserArchiveControl (Page 469)

ColumnReadonly (RT Professional)**Description**

Specifies whether the values of a column referenced with "ColumnIndex" can only be read.

Access in runtime Read and write

Syntax

Object.**ColumnReadonly**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if the values of the column are read-only values.

FALSE, if the values of the column are not read-only values.

See also

UserArchiveControl (Page 469)

ColumnRemove (RT Professional)**Description**

Removes a column referenced with its name.

Access in runtime Read and write

Syntax

Object.**ColumnRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl

STRING

Optional. A value or a constant that returns the name of the referenced column to be removed.

See also

TrendRulerControl (Page 444)

ColumnRepos (RT Professional)**Description**

For multiple columns, specifies the position of a column referenced with "ColumnIndex".

If you have changed the position of the column with "ColumnRepos", "ColumnIndex" is assigned the value of "ColumnRepos".

Access in runtime Read and write

Syntax

Object.**ColumnRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the position of the referenced column. Value range from 0 to "ColumnCount" minus 1. Values outside this range have no effect.

0: The referenced column is positioned on the left.

See also

ColumnIndex (Page 612)

ColumnCount (Page 608)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ColumnResize (RT Professional)

Description

Specifies whether you can change the width of the columns.

Access in runtime Read and write

Syntax

Object.**ColumnResize**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if you can change the width of the columns.

FALSE if you cannot change the width of the columns.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

Columns (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnScrollbar (RT Professional)

Description

Specifies when the horizontal scroll bar is to be displayed.

Access in runtime Read and write

Syntax

Object.**ColumnScrollbar**[=ScrollbarVisibility]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

ScrollbarVisibility

Optional. A value or a constant that specifies when the horizontal scroll bar is displayed.

Value	Description
0	The horizontal scroll bar is not displayed.
1	If the display area is too small for all columns, the horizontal scroll bar is displayed.
2	The horizontal scroll bar is always displayed.

See also

UserArchiveControl (Page 469)

TrendRulerControl (Page 444)

OnlineTableControl (Page 348)

AlarmControl (Page 236)

ColumnSettings (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnSettingsBufferView (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnShowDate (RT Professional)**Description**

Specifies whether the date is to be displayed in a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnShowDate**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if the date is displayed in the column.

FALSE, if the date is not displayed in the column.

See also

UserArchiveControl (Page 469)

ColumnShowIcon (RT Professional)

Description

Specifies whether a symbol is to be displayed in a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnShowIcon**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if an icon is displayed in the column.

FALSE, if no icon is displayed in the column.

See also

UserArchiveControl (Page 469)

ColumnShowTitleIcon (RT Professional)

Description

Specifies whether a symbol is to be displayed in a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnShowTitleIcon**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if an icon is displayed in the column.

FALSE, if no icon is displayed in the column.

See also

UserArchiveControl (Page 469)

ColumnsMoveable (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnSort (RT Professional)**Description**

Specifies the sorting type of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnSort**[=SortMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl
- UserArchiveControl

SortMode

Optional. A value or a constant that specifies the sorting mode.

Value	Description
0	No sorting
1	Ascending sorting from the smallest to largest value
2	Descending sorting from the largest to smallest value

See also

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ColumnSortIndex (RT Professional)

Description

Specifies the sorting order for a column referenced with "ColumnIndex" in which the columns are sorted one after another.

Access in runtime Read and write

Syntax

Object.**ColumnSortIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the sorting index of the column.

Value	Description
0	The sorting direction defined in the "ColumnSort (Page 623)" property is set to "No sorting".
1	The number 1 is displayed in the header. Sorting by this column takes place first.
2 - 4	The selected number is displayed in the header. The columns are sorted in the corresponding order.

See also

ColumnIndex (Page 612)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ColumnStartValue (RT Professional)

Description

Returns the start value defined in the user log of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnStartValue**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

STRING

Optional. A value or a constant that returns the start value of the column specified in the user log.

See also

UserArchiveControl (Page 469)

ColumnStringLength (RT Professional)**Description**

Returns the string length defined in the user archive for a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnStringLength**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that returns the length of the character string of the column specified in the user log.

See also

UserArchiveControl (Page 469)

ColumnTextAckGroup (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnTextAlarmState (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnTextAlarmText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnTextBit (RT Advanced)

Description

No access in runtime.

ColumnTextClassName (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnTextConnection (RT Advanced)

Description

No access in runtime.

ColumnTextDataType (RT Advanced)

Description

No access in runtime.

ColumnTextDate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnTextDateTime (RT Advanced)**Description**

No access in runtime.

ColumnTextDbNumber (RT Advanced)**Description**

No access in runtime.

ColumnTextDevice (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnTextDiagnosable (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnTextFormat (RT Advanced)**Description**

No access in runtime.

ColumnTextGroup (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnTextLogTime (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnTextNumber (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnTextOffset (RT Advanced)

Description

No access in runtime.

ColumnTextPassword (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnTextRead (RT Advanced)

Description

No access in runtime.

ColumnTextTagConnection (RT Advanced)

Description

No access in runtime.

ColumnTextTime (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ColumnTextTrend (RT Advanced)

Description

No access in runtime.

ColumnTextType (RT Advanced)**Description**

No access in runtime.

ColumnTextUser (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ColumnTextValue (RT Advanced)**Description**

No access in runtime.

ColumnTextWrite (RT Advanced)**Description**

No access in runtime.

ColumnTextXValue (RT Advanced)**Description**

No access in runtime.

ColumnTimeFormat (RT Professional)**Description**

Specifies the time format for a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnTimeFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

STRING

Optional. A value or a constant that specifies the time format for the column.

Value	Description
Automatic	The time format is set automatically.
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.

See also

UserArchiveControl (Page 469)

ColumnTitleAlignment (RT Professional)**Description**

Specifies the alignment of the header in a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnTitleAlignment**[=GridColumnHeaderHorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridColumnHeaderHorizontalAlignment

Optional. A value or a constant that specifies the column header alignment.

Value	Designation	Description
0	Left	The column header is left-aligned.
1	Centered	The column header is centered.
2	Right	The column header is right-aligned.
3	As table	The column header is aligned as defined in the "ColumnAlignment (Page 606)" property for the column content.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

ColumnTitles (RT Professional)**Description**

Specifies whether the column header is displayed.
Access in runtime: Read and write

Syntax

Object.**ColumnTitles**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchive Control

BOOLEAN

Optional.

TRUE if the column header is displayed.

FALSE if the column header is not displayed.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

ColumnType (RT Professional)**Description**

Returns the data type defined in the user archive of a column referenced with "ColumnIndex".

Access in runtime Read and write

Syntax

Object.**ColumnType**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that returns the data type of the column specified in the user log.

See also

UserArchiveControl (Page 469)

ColumnVisible (RT Professional)

Description

Specifies whether the column referenced with "ColumnIndex" is displayed in the object.

Access in runtime Read and write

Syntax

Object.**ColumnVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the referenced column is displayed.

FALSE if the referenced column is not displayed.

See also

ColumnIndex (Page 612)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ColumnWidth (RT Advanced)**Description**

No access in runtime.

ColumnWriteAccess (RT Professional)**Description**

Returns the write access authorization defined in the user archive for a column referenced with "ColumnIndex".

The number corresponds to the number that is assigned to the authorization in the "User Administrator" editor.

Access in runtime Read and write

Syntax

Object.**ColumnWriteAccess**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that returns the authorization for write access specified in the user log to the column.

See also

UserArchiveControl (Page 469)

ComboboxFont (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

CompatibilityMode (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ComplexViewToolbar (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ComplexViewToolbarBounds (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ComponentInfoText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ComputerName (RT Professional)

Description

Returns the name of the computer on which the alarm object was triggered.

ComputerName (readonly)

ConfiguredAlarmClasses (RT Advanced)

Description

No access in runtime.

ConnectionType (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the type of connector. You can select one of two connection types.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ConnectionType**[=ConnectorConnectionType]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector

You have no access in runtime with the following format:

- SmartClientView

ConnectorConnectionType

Optional. A value or a constant that specifies the type of connector.

(0): Automatic: Both objects are connected by a polyline made up of horizontal and vertical parts.

(1): Simple: Both objects are connected by a straight line between the connecting points. The fields are right-aligned.

See also

Connector (Page 293)

SmartClientView (Page 413)

ConnectOnStart (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ConnectTrendWindows (RT Professional)

Description

Specifies whether the configured trend views are connected. The requirement is that you have configured several trend views.

The connected trend views have the following properties:

- A common X axis
- A scrollbar
- A ruler

Access in runtime Read and write

Syntax

Object.**ConnectTrendWindows**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if all configured trend views are connected.

FALSE if the trend views are displayed separately.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

Context (RT Professional)

Description

Reads or sets the alarm object server prefix.

ContinousChange (RT Professional)

Description

Specifies whether the value of the "ProcessValue" property will be transferred when the mouse button is released or as soon as the slider position changes in Runtime.

Access in runtime: Read and write

Syntax

Object.**ContinousChange**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Slider

BOOLEAN

Optional TRUE, if the value of the property "ProcessValue" will be transferred when the mouse button is released or as soon as the slider position changes in Runtime.

See also

Slider (Page 410)

ControlDesignMode (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the control design.

Access in runtime Read and write

Syntax

Object.**ControlDesignMode**[=RTControlModes]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

RTControlModes

Optional. A value or a constant that specifies the control design.

Value	Designation	Description
	Project setting	The design corresponds to the project settings in WinCC Explorer.
0	Simple	Classic WinCC design
1	Standard	New WinCC V7 design

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

CornerRadius (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

CornerStyle (RT Professional)

Description

Specifies the shape of the corners.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**CornerStyle**[=CornerStyle | LineJoinStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line

- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- Tubepolyline
- WindowSlider

You have no access in runtime with the following format:

- Switch
- TubeArcObject

CornerStyle | LineJoinStyle

Optional. A value or a constant that specifies the shape of the corners.

Value	Description
0	Solid line
1	Dashed line
2	Dotted line
3	Dashed and dotted line
4	Dash-dot-dot line

Count (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns the number of elements in the specified list.

Access in Runtime: Read

Syntax

Object.Count

Object

Required A "Collection" object.

CountDivisions (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the number of segments in which the bar will be divided by the large marking lengths of the scale.

Access in runtime: Read and write

Syntax

Object.CountDivisions[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the number of segments in which the bar will be divided by the large marking lengths of the scale.

0-100: An object can be divided into a maximum of 100 segments

= 0: The optimum number of segments will be established automatically.

See also

Bar (Page 258)

CountOfLinesPerAlarms (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

CountOfVisibleAlarms (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

CountSubDivisions (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the number of scale marks between two main marks.

Access in runtime Read and write

Syntax

Object.**CountSubDivisions**[= Int32]

Object

Required. A ScreenItem object with the following format

- Bar

Int32

Optional. A value or a constant that specifies the number of scale segments.

See also

Bar (Page 258)

CountVisibleItems (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies how many lines the selection list will contain. If the number of configured texts is greater than this value, then the selection list will have a vertical scroll bar.

Access in runtime:

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**CountVisibleItems**[=Int32]

Object

Required. A "ScreenItem" object with the format:

- ComboBox
- ListBox
- SymbolicIOField

You have no access in runtime with the following format:

- StatusForce
- TrendView
- UserView

Int32

Optional A value or a constant that specifies how many lines the selection list will contain.

See also

ComboBox (Page 290)

Listbox (Page 340)

SymbolicIOField (Page 424)

StatusForce (Page 415)

TrendView (Page 454)

UserView (Page 481)

CurrentContext (RT Professional)

Description

Returns a character string depending on the use of the function.

If the function is contained in a screen in the screen window, CurrentContext returns the symbolic server name which supplies the screen. Example: "WinCCProject_MyComputer::"

If the function is contained in the main screen, an empty character string is returned.

Access in Runtime: Read

Syntax

Object.**CurrentContext**

Object

Required A "HMIRuntime" object.

See also

HMIRuntime (Page 208)

CursorControl (RT Professional)

Description

Specifies whether the mouse cursor jumps to the next field of the TAB sequence after leaving the field.

Access in runtime: Read and write

Syntax

Object.**CursorControl**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField
- SymbolicIOField

BOOLEAN

Optional TRUE, if the mouse cursor jumps to the next field of the TAB-order after leaving the field.

Comments

For this purpose, the "CursorMode" property must be set to TRUE.

See also

IOField (Page 332)

SymbolicIOField (Page 424)

Curves (RT Advanced)

Description

No access in runtime.

1.5.5.4 Properties D (Panels, Comfort Panels, RT Advanced, RT Professional)

DangerRangeColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the danger range on the scale of the "Gauge" object.

The "DangerRangeVisible" property must have the value TRUE so that the warning range is displayed.

Access in runtime Read and write

Syntax

Object.**DangerRangeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the color of the warning range.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

DangerRangeStart (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies at which scale value the danger range of the "Gauge" object will start.

The "DangerRangeColor" property must have the value TRUE so that the warning range is displayed.

Access in runtime Read and write

Syntax

Object.**DangerRangeStart**[=SINGLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

SINGLE

Optional. A value or a constant that specifies the scale value for the start of the warning range.

Comments

The range extends from the value "DangerRangeStart" through to the end of the scale.

See also

Gauge (Page 320)

DangerRangeVisible (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the warning range in the scale of the "Gauge" object will be displayed.

Access in runtime: Read and write

Syntax

Object.**DangerRangeVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the warning range will be displayed in the scale.

Comments

Specifies the color of the warning range with the "DangerRangeColor" property.

Specifies the start of the warning range with the "DangerRangeStart" property.

See also

Gauge (Page 320)

DataFormat (RT Professional)**Description**

Returns the display format.

Access in runtime: Read and write

Syntax

Object.**DataFormat**[=IOFieldDataFormat]

Object

Required. An object of the type "ScreenItem" with the format:

- IOField

IOFieldDataFormat

Optional. A value or a constant that returns the display format.

Value	Designation	Description
0	Binary	The content is shown in "Binary" data format.
1	Decimal	The content is shown in "Decimal" data format.
2	String	Represents character strings.
3	Hexadecimal	The content is shown in "Hexadecimal" data format.

See also

IOField (Page 332)

DataLogs (RT Professional)**Description**

Returns an object of type "DataLogs".

Access in Runtime: Read

Syntax

Object.**DataLogs**

Object

Required A "Logging" object.

See also

Logging (Page 213)

DataProviderGuid (RT Professional)**Description**

Returns the GUID of the control.

Access in Runtime: Read

Syntax

Object.DataProviderGuid

Object

Required. An object of the "UserArchiveControl" type.

Example

```
Dim objRecipeview
Set objRecipeview= HMIRuntime.Screens("Screen_1").ScreenItems("Recipeview_1")
SmartTags("Tag1") = objRecipeView.DataProviderGuid
```

See also

UserArchiveControl (Page 469)

DataRecordNameCaption (RT Advanced)

Description

No access in runtime.

DataRecordNrCaption (RT Advanced)

Description

No access in runtime.

DataSet (RT Professional)

Description

Returns an object of type "DataSet".

Access in Runtime: Read

Syntax

Object.**DataSet**

Object

Required A "Screen" object.

See also

HMIRuntime (Page 208)

Screen (Page 215)

DataSource (RT Professional)

Description

No access in runtime.

DefaultFilterEom (RT Professional)

Description

No access in runtime.

DefaultHitListFilterEom (RT Professional)

Description

No access in runtime.

DefaultMsgFilterSQL (RT Professional)

Description

Specifies an SQL statement as default for alarm filter.

Access in runtime: Read and write

Syntax

Object.**DefaultMsgFilterSQL**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl

STRING

Optional. A value or a constant that specifies an SQL statement as default for alarm filter.

See also

AlarmControl (Page 236)

DefaultSort (RT Professional)

Description

Specifies the type of sorting.

Access in runtime: Read and write

Syntax

Object.**DefaultSort**[=SortMode]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl

SortMode

Optional. A value or a constant that specifies the type of sorting.

Value	Explanation
0	No sorting
1	The alarms are updated starting from the bottommost line.
2	The alarms are updated starting from the topmost line.

See also

AlarmControl (Page 236)

DefaultSort2 (RT Professional)

Description

Specifies the type of sorting.

Access in runtime: Read and write

Syntax

Object.**DefaultSort2**[=SortMode]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl

SortMode

Optional. A value or a constant that specifies the type of sorting.

Value	Explanation
0	No sorting
1	The alarms are updated starting from the bottommost line.
2	The alarms are updated starting from the topmost line.

See also

AlarmControl (Page 236)

DefaultSort2Column (RT Professional)**Description**

Specifies a column for initial sorting of the table entries.

If no value is assigned, alarms are sorted according to the following column order:

- Date/Time/Number

If you configure the "Alarm text" column as a value, for example, alarms are sorted according to the following column order:

- Alarm text/Date/Time/Number

Access in runtime Read and write

Syntax

Object.**DefaultSort2Column**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the name of the column for initial sorting of the table entries.

See also

AlarmControl (Page 236)

DeviceStyle (RT Professional)**Description**

No access in runtime.

DiagnosticsContext (RT Professional)**Description**

Specifies the diagnostics context.

Access in runtime: Read and write

Syntax

Object.**DiagnosticsContext**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl

STRING

Optional. A value or a constant that specifies the diagnostics context.

See also

AlarmControl (Page 236)

DiagramAreaHeight (RT Advanced)**Description**

No access in runtime.

DiagramAreaLeft (RT Advanced)**Description**

No access in runtime.

DiagramAreaTop (RT Advanced)

Description

No access in runtime.

DiagramAreaWidth (RT Advanced)

Description

No access in runtime.

DialColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the dial for the selected object.

Access in runtime Read and write

Syntax

Object.**DialColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock
- Gauge

Color

Optional. A value or a constant that specifies the color of the dial.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Clock (Page 287)

Gauge (Page 320)

DialFillStyle (RT Professional)**Description**

Specifies the type of background.

Access in runtime Read and write

Syntax

Object.**DialFillStyle**[=GaugeFillStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

GaugeFillStyle

Optional. A value or a constant that specifies the type of background.

Value	VB constant	Explanation
0	hmiBackStyleSolid	The rectangular background of the display is filled with the specified border color. The scale is filled with the specified background color.
1	hmiBackStyleFrameTransparent	The rectangular background is transparent. The scale is filled with the specified background color. As a result, a circular display can be shown.
2	hmiBackStyleTransparent	The rectangular background and the scale are transparent.

See also

Gauge (Page 320)

DialPicture (RT Professional)**Description**

Specifies a graphic for the dial face.

Access in runtime: Read and write

Syntax

Object.**DialPicture**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following characteristics:

- Gauge

HmiObjectHandle

Optional. A value or a constant that specifies the graphic for the dial face.

See also

Gauge (Page 320)

DialSize (RT Professional)

Description

Specifies the diameter of the scale in relation to the smaller value of the geometry attributes "Width" and "Height".

Access in runtime: Read and write

Syntax

Object.**DialSize**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that specifies the diameter of the scale in relation to the smaller value of the geometry attributes "Width" and "Height".

See also

Gauge (Page 320)

Display3D (RT Advanced)

Description

No access in runtime.

DisplayButton2Pic (RT Advanced)

Description

No access in runtime.

DisplayButtonComparison (RT Advanced)**Description**

No access in runtime.

DisplayButtonDelete (RT Advanced)**Description**

No access in runtime.

DisplayButtonFromPlc (RT Advanced)**Description**

No access in runtime.

DisplayButtonHelp (RT Advanced)**Description**

No access in runtime.

DisplayButtonNew (RT Advanced)**Description**

No access in runtime.

DisplayButtonSave (RT Advanced)**Description**

No access in runtime.

DisplayButtonSaveAs (RT Advanced)**Description**

No access in runtime.

DisplayCentury (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

DisplayComboBox (RT Advanced)

Description

No access in runtime.

DisplayGridLines (RT Advanced)

Description

No access in runtime.

DisplayLabeling (RT Advanced)

Description

No access in runtime.

DisplayNumbers (RT Advanced)

Description

No access in runtime.

DisplayOptions (RT Professional)

Description

Specifies whether alarms whose view was suppressed are displayed.

Access in runtime Read and write

Syntax

Object.**DisplayOptions**[=AlarmDisplayOptions]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl

AlarmDisplayOptions

Optional. A value or a constant that specifies whether alarms that were hidden are displayed.

Value	Explanation
0	All messages
1	Only displayed messages
2	Only hidden messages

See also

AlarmControl (Page 236)

DisplaySize (RT Advanced)**Description**

No access in runtime.

DisplayStatusBar (RT Advanced)**Description**

No access in runtime.

DisplaySystemTime (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

DisplayTable (RT Advanced)**Description**

No access in runtime.

DoubleClickAction (RT Professional)**Description**

Specifies the action to be executed in runtime by double-clicking on a message line.

Access in runtime Read and write

Syntax

Object.**DoubleClickAction**[=AlarmControlActions]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

AlarmControlActions

Optional. A value or a constant that specifies the action to be executed in runtime when you double-click on an alarm line.

Value	Description	Explanation
0	None	No action.
1	Loop-in-alarm	Calls the "Loop-in-alarm" function.
2	Open comments dialog	Calls the "Comments dialog" button function.
3	Open Infotext dialog	Calls the "Infotext dialog" button function.
4	Column-dependent	The action is determined by the column in which you double-clicked.

See also

AlarmControl (Page 236)

DrawInsideFrame (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the border line is to be shown with a line thickness of greater than 1 within the border or symmetrically to the border.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**DrawInsideFrame**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- OptionGroup
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

You have no access in runtime with the following format:

- Switch
- TubeArcObject

BOOLEAN

Optional. TRUE if the border line is shown with a line thickness of greater than 1 within the border.

Drive (RT Professional)**Description**

No access in runtime.

1.5.5.5 Properties E-F (Panels, Comfort Panels, RT Advanced, RT Professional)

EdgeStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the line style.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**EdgeStyle**[=LineStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle *
- CircleSegment
- ComboBox
- Ellipse *
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField *
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon *
- Rectangle *
- RoundButton
- SymbolicIOField
- TextField *
- WindowSlider

* RT Advanced read and write access

You have no access in runtime with the following format:

- AlarmView
- Clock
- DateTimeField
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

LineStyle

Optional. A value or a constant that specifies the line style. Value range from -1 to 4.

The objects "Ellipse", "Circle", "Rectangle" and "Polygon" support the line styles:

Value	VB constant	Explanation
0	hmiLineStyleSolid	Solid line
1	hmiLineStyleDash	Dashed line
2	hmiLineStyleDot	Dotted line
3	hmiLineStyleDashDot	Dash-dot line
4	hmiLineStyleDashDot-Dot	Dash-dot-dot line

The objects "TextField" and "IOField" support only the line styles:

Value	VB constant	Explanation
-1	hmiLineStyleNone	No line
0	hmiLineStyleSolid	Solid line

See also

[Bar \(Page 258\)](#)
[Button \(Page 266\)](#)
[CheckBox \(Page 275\)](#)
[Circle \(Page 279\)](#)
[CircleSegment \(Page 282\)](#)
[ComboBox \(Page 290\)](#)
[Ellipse \(Page 300\)](#)
[EllipseSegment \(Page 303\)](#)

GraphicIOField (Page 324)
GraphicView (Page 327)
IOField (Page 332)
Listbox (Page 340)
MultiLineEdit (Page 345)
OptionGroup (Page 372)
Polygon (Page 379)
Rectangle (Page 396)
RoundButton (Page 399)
SymbolicIOField (Page 424)
TextField (Page 440)
WindowSlider (Page 485)
AlarmView (Page 251)
DateTimeField (Page 296)
Gauge (Page 320)
RecipeView (Page 390)
Slider (Page 410)
StatusForce (Page 415)
Switch (Page 420)
SysDiagControl (Page 431)
TrendView (Page 454)
UIView (Page 481)

EditOnFocus (RT Professional)

Description

Specifies whether the data input is immediately possible if the input field is selected using the <Tab> key.

Access in runtime: Read and write

Syntax

Object.**EditOnFocus**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField
- SymbolicIOField

BOOLEAN

Optional TRUE, if the data input is immediately possible and the input field has been selected using the <Tab> key.

See also

IOField (Page 332)

SymbolicIOField (Page 424)

Enabled (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the selected object can be operated in runtime.

Access in runtime: Read and write

Syntax

Object.**Enabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl *
- AlarmView **
- Bar
- Button
- ChannelDiagnose *
- CheckBox *
- Circle
- CircleSegment *
- CircularArc *
- Clock
- ComboBox *
- Connector *
- DateTimeField **

- DiscSpaceView *
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl **
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox *
- MediaPlayer
- MultiLineEdit *
- OptionGroup *
- PLCCodeViewer **
- PdfView **
- Polygon
- Polyline
- ProDiagOverview ***
- RecipeView **
- Rectangle
- RoundButton *
- S7GraphOverview **
- Slider
- SmartClientView **
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField
- SysDiagControl ****
- TextField
- TrendRulerControl *
- TrendView **
- TubeArcObject *

- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserView
- WindowSlider *

* RT Advanced No access

** RT Professional No access

*** RT Advanced Read access

**** Read access only

You have no access in runtime with the following format:

- OnlineTableControl
- OnlineTrendControl
- UserArchiveControl

BOOLEAN

Optional. TRUE, if the specified object can be operated in Runtime.

See also

AlarmControl (Page 236)

AlarmView (Page 251)

Bar (Page 258)

Button (Page 266)

ChannelDiagnose (Page 274)

CheckBox (Page 275)

Circle (Page 279)

CircleSegment (Page 282)

CircularArc (Page 285)

Clock (Page 287)

ComboBox (Page 290)

DateTimeField (Page 296)

DiskSpaceView (Page 298)

Ellipse (Page 300)

EllipseSegment (Page 303)

EllipticalArc (Page 306)

FunctionTrendControl (Page 309)

Gauge (Page 320)
GraphicIOField (Page 324)
GraphicView (Page 327)
HTMLBrowser (Page 330)
IOField (Page 332)
Line (Page 337)
Listbox (Page 340)
MediaPlayer (Page 343)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)
PLCCodeViewer (Page 377)
Polygon (Page 379)
Polyline (Page 382)
RecipeView (Page 390)
Rectangle (Page 396)
RoundButton (Page 399)
S7GraphOverview (Page 403)
Slider (Page 410)
SmartClientView (Page 413)
StatusForce (Page 415)
Switch (Page 420)
SymbolicIOField (Page 424)
SymbolLibrary (Page 429)
SysDiagControl (Page 431)
TextField (Page 440)
TrendRulerControl (Page 444)
TrendView (Page 454)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)
UserArchiveControl (Page 469)

UserView (Page 481)

WindowSlider (Page 485)

EnableDelete (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether data can be deleted in runtime.

Access in runtime: Read and write

Syntax

Object.**EnableDelete**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if data can be deleted in runtime.

FALSE, if data cannot be deleted in runtime.

See also

UserArchiveControl (Page 469)

EnableEdit (RT Professional)

Description

Specifies whether the data shown can be edited in runtime.

Access in runtime Read and write

Syntax

Object.**EnableEdit**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the data can be changed in runtime.

FALSE if the data cannot be changed in runtime.

See also

OnlineTableControl (Page 348)

UserArchiveControl (Page 469)

EnableInsert (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether data can be inserted in runtime.

Access in runtime: Read and write

Syntax

Object.**EnableInsert**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if data can be inserted in runtime.

FALSE, if data cannot be inserted in runtime.

See also

UserArchiveControl (Page 469)

EnableNavigateButtons (RT Advanced)

Description

No access in runtime.

EnableNavigateKeys (RT Advanced)**Description**

No access in runtime.

EncryptCommunication (RT Advanced)**Description**

No access in runtime.

EndAngle (RT Professional)**Description**

Specifies the angle by which the end point deviates from the zero position (0°).

Access in runtime Read and write

Syntax

Object.**EndAngle**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- CircleSegment
- CircularArc
- EllipseSegment
- EllipticalArc
- TubeArcObject

Int32

Optional. A value or a constant that specifies the angle by which the end point deviates from the zero position (0°).

See also

CircleSegment (Page 282)

CircularArc (Page 285)

EllipseSegment (Page 303)

EllipticalArc (Page 306)

TubeArcObject (Page 460)

EndLeft (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

EndStyle (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies how the line end is displayed.

Access in runtime: Read and write

Syntax

Object.**EndStyle**[=LineEndStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector
- Line
- Polyline

LineEndStyle

Optional. A value or a constant that specifies the line-end shapes.

Value	VB constant	Description
0	hmiLineEndStyleNone	The line has no end symbol.
65536*	hmiLineEndStyleArrow	The line ends with an empty arrowhead.
131072	hmiLineEndStyleFilledArrow	The line ends with a filled arrowhead.
196608*	hmiLineEndStyleFilledArrowReversed	The line ends with a reversed arrowhead.
262144*	hmiLineEndStyleLine	The line ends with a vertical line.
327680*	hmiLineEndStyleCircle	The line ends with a circle.
393216*	hmiLineEndStyleFilledCircle	The line ends with a filled circle.

* Only RT Professional

See also

Line (Page 337)

Polyline (Page 382)

Connector (Page 293)

EndTop (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

EnterButtonVisible (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

EntryNameCaption (RT Advanced)**Description**

No access in runtime.

EntryNameColumnWidth (RT Advanced)**Description**

No access in runtime.

EntryValueColFirst (RT Advanced)**Description**

No access in runtime.

EntryValueColumnWidth (RT Advanced)**Description**

No access in runtime.

EntryValueFieldLength (RT Advanced)**Description**

No access in runtime.

EntryValuePos (RT Advanced)

Description

No access in runtime.

ErrorColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of errors in the GRAPH overview.

Access in runtime: Read and write

Syntax

Object.ErrorColor[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- S7GraphOverview

Color

Optional. A value or a constant that specifies the color for errors.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

The background color is not visible if the "BorderStyle" property has the value "0".

See also

S7GraphOverview (Page 403)

ErrorDescription (RT Professional)

Description

Returns one of the following error descriptions in English:

Output	Description
" "	OK
"Operation Failed"	Execution error
"Variable not found"	Tag error
"Server down"	Server not available.
"An error occurred for one or several tags"	Multi Tag Error (Error in one or several tags)

To obtain an error description, first of all carry out the Read method.

Note

If the error occurs when accessing via the TagSet object, evaluate the ErrorDescription property for each tag of the TagSet object.

Access in Runtime: Read

Syntax

Object.**ErrorDescription**

Object

Required A "Tag" object.

Example

The following example shows the error description for the "Tag1" tag:

```
'VBS72
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objtag.Read
MsgBox objTag.ErrorDescription
```

The following example adds two tags to the TagSet list and outputs the ErrorDescription property as Trace:

```
'VBS179
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "ErrorDescription: " & group.ErrorDescription & vbNewLine
```

The ErrorDescription property of a tag contained in the list may be accessed as follows:

```
HMIRuntime.Trace "ErrorDescription: " & group("Motor1").ErrorDescription & vbNewLine
```

See also

Tag (Page 228)

TagSet (list) (Page 232)

Errorflag (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether an error description is displayed in the device / detail view.

Access in runtime Read and write

Syntax

Object.**Errorflag**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- S7GraphOverview

BOOLEAN

Optional.

TRUE, if an error description is displayed in the device/detail view.

FALSE, if no error description is displayed in the device/detail view.

See also

S7GraphOverview (Page 403)

ES2RT_ButtonPositions (RT Advanced)**Description**

No access in runtime.

ES2RT_ColumnOrder (RT Advanced)**Description**

No access in runtime.

ES2RT_ColumnWidth (RT Advanced)**Description**

No access in runtime.

ES2RT_EntryNameColumnWidth (RT Advanced)**Description**

No access in runtime.

ES2RT_EntryValueColumnWidth (RT Advanced)**Description**

No access in runtime.

ES2RT_ListAreaHeight (RT Advanced)**Description**

No access in runtime.

ES2RT_ListAreaWidth (RT Advanced)**Description**

No access in runtime.

ES2RT_MessageAreaHeight (RT Advanced)

Description

No access in runtime.

ES2RT_MessageAreaWidth (RT Advanced)

Description

No access in runtime.

ES2RT_StoreAsCheckBack (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Es2rtButtonPositions (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Es2rtTableBounds (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

EscButtonVisible (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

EsPreviewType (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

EvenRowBackColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ExportDelimiter (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ExportDirectoryChangeable (RT Professional)**Description**

Specifies whether the data export directory can be changed in runtime.

Access in runtime: Read and write

Syntax

Object.**ExportDirectoryChangeable**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the directory for data export can be changed in runtime.

FALSE if the directory for data export cannot be changed in runtime.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportDirectoryname (RT Professional)

Description

Defines the directory to which the exported Runtime data is written.

Access in runtime: Read and write

Syntax

Object.**ExportDirectoryname**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant which specifies the directory.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportFileExtension (RT Professional)

Description

Specifies the file extension of the export file. Only the "csv" file extension is currently supported.

Access in runtime: Read and write

Syntax

Object.**ExportFileExtension**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. Specifies the file extension of the export file.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportFilename (RT Professional)

Description

Specifies the name of the target file for the exported runtime data.

Access in runtime: Read and write

Syntax

Object.**ExportFilename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. Specifies the name of the target file for the exported runtime data.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportFilenameChangeable (RT Professional)**Description**

Specifies whether the export file name can be changed in runtime.

Access in runtime: Read and write

Syntax

Object.**ExportFilenameChangeable**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the export file can be renamed in runtime.

FALSE if the export file cannot be renamed in runtime.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportFormat (RT Professional)**Description**

No access in runtime.

ExportFormatGuid (RT Professional)**Description**

Specifies the assignment of ID number and export provider.

Access in runtime: Read and write

Syntax

Object.**ExportFormatGuid**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or constant that specifies the assignment of ID number and export provider.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportFormatName (RT Professional)

Description

Defines the export file format. Only the "csv" file format is currently available for the export.

Access in runtime: Read and write

Syntax

Object.**ExportFormatName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that defines the file format for export.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportParameters (RT Professional)

Description

Specifies the parameters of the selected format by means of the properties dialog.

Access in runtime Read and write

Syntax

Object.**ExportParameters**

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

ExportParameters

Optional. A value or constant that specifies the parameters of the selected format in the Properties dialog.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportSelection (RT Professional)

Description

Specifies which runtime data of the control is exported.

Access in runtime: Read and write

Syntax

Object.**ExportSelection**[=ExportRange]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

ExportRange

Optional. A value or a constant that specifies which runtime data of the control can be exported.

Value	Description	Explanation
0	all	All Runtime data of the control is exported.
1	Selection	Selected Runtime data of the control is exported.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExportShowDialog (RT Professional)

Description

Enables the display of the export dialog during runtime.

Access in runtime: Read and write

Syntax

Object.**ExportShowDialog**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional TRUE, if the dialog box is shown in Runtime.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ExtendedZoomingEnable (RT Professional)**Description**

Specifies whether the operator can zoom the screen in or out in runtime by turning the mouse wheel.

Access during runtime: Read and write

Syntax

Object.**ExtendedZoomingEnable**[=BOOLEAN]

Object

Necessary. A "Screen" object.

BOOLEAN

Optional TRUE, if the operator can zoom the screen in and out in runtime.

Example

The following example shows how the extended zoom can be enabled for the NewPDL1 screen:

```
'VBS155
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
objScreen.ExtendedZoomingEnable = 1
```

See also

Screen (Page 215)

ExtraHeightOffset (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FieldLength (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FileName (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the name of the file to be loaded.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FileName**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

You have no access in runtime with the following format:

- PdfView

STRING

Optional. A value or a constant that specifies the name of the file to be loaded.

See also

MediaPlayer (Page 343)

FillColorMode (RT Professional)

Description

Specifies the type of foreground for the selected object.

Access in runtime Read and write

Syntax

Object.**FillColorMode**[=SymbolLibraryFillColorMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolLibrary

SymbolLibraryFillColorMode

Optional. A value or a constant that specifies the type of foreground for the object.

Value	VB constant	Explanation
0	hmiSymbolLibraryAppearanceOriginal	The surface is gray.
1	hmiSymbolLibraryAppearanceShaded	The display is shaded.
2	hmiSymbolLibraryAppearanceSolid	The display is solid.
3	hmiSymbolLibraryAppearanceTransparent	The display is gray.

See also

SymbolLibrary (Page 429)

FillPattern (RT Professional)

Description

No access in runtime.

FillColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the fill pattern.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FillColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- IOField
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField
- WindowSlider

*: Read access only

You have no access in runtime with the following format:

- AlarmControl
- ComboBox
- DateTimeField
- ListBox
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color of the fill pattern.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
Circle (Page 279)
CircleSegment (Page 282)
ComboBox (Page 290)
Ellipse (Page 300)
EllipseSegment (Page 303)
GraphicView (Page 327)
IOField (Page 332)
Listbox (Page 340)
OptionGroup (Page 372)
Polygon (Page 379)
Rectangle (Page 396)
RoundButton (Page 399)
SymbolicIOField (Page 424)

TextField (Page 440)
WindowSlider (Page 485)
AlarmControl (Page 236)
DateTimeField (Page 296)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

FillStyle (RT Advanced)

Description

Specifies whether the background color is displayed with a broken line.
Access in runtime Read and write

Syntax

Object.**FillStyle**[=LineFillStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Line
- Polyline

LineFillStyle

Optional. A value or a constant that specifies the fill style.

Value	VB constant	Description
0	hmiLineFillStyleTransparent	Transparent background
1	hmiLineFillStyleSolid	Object background is filled with the specified color

See also

Line (Page 337)
Polyline (Page 382)

FillingDirection (RT Professional)

Description

Specifies the fill direction.

Access in Runtime: Read and write

Syntax

Object.**FillingDirection**[=FillDirection]

Object

Required. An object of the type "ScreenItem" with the following format:

- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- TextField
- WindowSlider

FillDirection

Optional. A value or a constant which specifies the fill direction.

Filter (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FilterSQL (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Sets an SQL statement for the filter criterion.

Access in runtime: Read and write

Syntax

Object.**FilterSQL**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- UserArchiveControl

STRING

Optional. A value or constant that specifies the SQL statement for the filter criterion.

See also

UserArchiveControl (Page 469)

FilterTag (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FilterText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FirstConnectedObject (RT Professional)

Description

No access in runtime.

FirstConnectedObjectIndex (RT Professional)

Description

Specifies the index number of the upper connector point.

Access in runtime: Read and write

Syntax

Object.**FirstConnectedObjectIndex**[=Int]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector

Int

Optional. A value or a constant that specifies the index number of the upper connector point.

See also

Connector (Page 293)

FirstConnectedObjectName (RT Professional)

Description

Specifies the name of the object that is docked to the upper connector point.

Access in runtime: Read and write

Syntax

Object.**FirstConnectedObjectName**[=String]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector

String

Optional. A value or a constant that specifies the name of the object that is docked to the upper connector point.

See also

Connector (Page 293)

FirstGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FirstGradientOffset (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FitToLargest (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FitToSize (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FitToSizeLowerRows (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FitToSizeUpperRows (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FixedAspectRatio (RT Professional)

Description

Specifies whether the aspect ratio should be maintained or changed when the symbol size is altered.

Access in runtime: Read and write

Syntax

Object.**FixedAspectRatio**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- SymbolLibrary

BOOLEAN

Optional TRUE, if the aspect ratio should be maintained when the symbol size is altered.

See also

SymbolLibrary (Page 429)

Flashing (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the selected object flashes in runtime.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**Flashing**[=FlashingType]

Object

Required. An object of the type "ScreenItem" with the format:

- Clock
- Gauge
- Slider

You have no access in runtime with the following format:

- AlarmView
- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField

- Line
- OptionGroup
- Polygon
- Polyline
- RecipeView
- Rectangle
- RoundButton
- ScreenWindow
- SmartClientView
- StatusForce
- Switch
- SymbolicIOField
- SymbolLibrary
- SysDiagControl
- TextField
- TrendView
- TubeArcObject
- UserView
- WindowSlider

FlashingType

Optional. A value or a constant that specifies whether the object flashes in runtime.

Designation	Description
None	The object does not flash in runtime.
Standard	The object flashes in runtime.

FlashingColorOff (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color of the border line for the flashing status "Off".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FlashingColorOff**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only

You have no access in runtime with the following format:

- Bar
- Switch
- TubeArcObject

Color

Optional. A value or a constant that specifies the color of the border line for the flashing status "Off".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Button (Page 266)
Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
IOField (Page 332)
Line (Page 337)
OptionGroup (Page 372)
Polygon (Page 379)
Polyline (Page 382)
Rectangle (Page 396)
RoundButton (Page 399)
SymbolicIOField (Page 424)
TextField (Page 440)
Bar (Page 258)
CheckBox (Page 275)
Switch (Page 420)
TubeArcObject (Page 460)

FlashingColorOn (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the border line for the flashing status "On".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FlashingColorOn**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only

You have no access in runtime with the following format:

- Bar
- Switch
- TubeArcObject

Color

Optional. A value or a constant that specifies the color of the border line for the flashing status "On".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Button (Page 266)
Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
IOField (Page 332)
Line (Page 337)
OptionGroup (Page 372)
Polygon (Page 379)
Polyline (Page 382)
Rectangle (Page 396)
RoundButton (Page 399)
SymbolicIOField (Page 424)
TextField (Page 440)
Bar (Page 258)
CheckBox (Page 275)
Switch (Page 420)
TubeArcObject (Page 460)

FlashingEnabled (RT Professional)

Description

Specifies whether flashing is activated for the object in runtime.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FlashingEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only

You have no access in runtime with the following format:

- Bar
- Switch
- TubeArcObject

BOOLEAN

Optional.

TRUE if flashing is activated for the object in runtime.

FALSE if flashing is not activated for the object in runtime.

See also

Button (Page 266)

Circle (Page 279)

CircleSegment (Page 282)
CircularArc (Page 285)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
IOField (Page 332)
Line (Page 337)
OptionGroup (Page 372)
Polygon (Page 379)
Polyline (Page 382)
Rectangle (Page 396)
RoundButton (Page 399)
SymbolicIOField (Page 424)
TextField (Page 440)
Bar (Page 258)
CheckBox (Page 275)
Switch (Page 420)
TubeArcObject (Page 460)

FlashingOnLimitViolation (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FlashingRate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the flash rate of the border line.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FlashingRate**[=FlashingRate]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField

* Read access only

You have no access in runtime with the following format:

- Bar
- Switch
- TubeArcObject

FlashingRate

Optional. A value or a constant that specifies the flash rate of the border line.

Value	VB constant	Explanation
0	hmiFlashingRateSlow	The length of the flashing interval is 1000 ms.
1	hmiFlashingRateMedium	The length of the flashing interval is 500 ms.
2	hmiFlashingRateFast	The length of the flashing interval is 250 ms.

See also

Button (Page 266)
Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
IOField (Page 332)
Line (Page 337)
OptionGroup (Page 372)
Polygon (Page 379)
Polyline (Page 382)
Rectangle (Page 396)
RoundButton (Page 399)
SymbolicIOField (Page 424)
TextField (Page 440)
Bar (Page 258)
CheckBox (Page 275)
Switch (Page 420)
TubeArcObject (Page 460)

FlashTransparentColor (RT Professional)

Description

Specifies the color of the bitmap object of a flashing graphic that is set to "transparent".

Access in runtime: Read and write

Syntax

Object.**FlashTransparentColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- GraphicIOField

Color

Optional. A value or a constant that specifies the color of the bitmap object of a flashing graphic that is set to "transparent".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

The background color is not visible if the "BorderStyle" property has the value "0".

See also

GraphicIOField (Page 324)

Flip (RT Professional)**Description**

Specifies whether the symbol is flipped on the vertical and / or horizontal center axis of the symbol.

Access in runtime Read and write

Syntax

Object.**Flip**[=SymbolLibraryFlip]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolLibrary

SymbolLibraryFlip

Optional. A value or a constant that specifies whether the symbol is flipped on the vertical and / or horizontal center axis of the symbol.

Value	VB constant	Explanation
0	<code>hmiSymbolLibraryFlipNone</code>	The symbol is not flipped.
1	<code>hmiSymbolLibraryFlipHorizontal</code>	The symbol is flipped horizontally.
2	<code>hmiSymbolLibraryFlipVertical</code>	The symbol is flipped vertically.
3	<code>hmiSymbolLibraryFlipBoth</code>	The symbol is flipped horizontally and vertically.

See also

SymbolLibrary (Page 429)

FocusColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the focus border when the object has the focus.

Access in runtime:

- RT Advanced: Read and write
- RT Professional: No access

Syntax

Object.**FocusColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmView
- Button
- GraphicIOField
- RecipeView
- Slider *
- StatusForce
- Switch
- TrendView

* RT Professional read and write access

You have no access in runtime with the following format:

- SymbolicIOField

Color

Optional. A value or a constant that specifies the color of the focus border.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmView (Page 251)
RecipeView (Page 390)
Slider (Page 410)
StatusForce (Page 415)
Switch (Page 420)
TrendView (Page 454)
Button (Page 266)
GraphicIOField (Page 324)
SymbolicIOField (Page 424)

FocusWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the border width when the object has the focus.

Access in runtime:

- RT Advanced: Read and write
- RT Professional: No access

Syntax

Object.**FocusWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmView
- Button
- GraphicIOField
- RecipeView
- Slider *
- StatusForce
- Switch
- TrendView

* RT Professional read and write access

You have no access in runtime with the following format:

- SymbolicIOField

Int32

Optional. A value or a constant that specifies the border width in pixels. Value range from 1 to 10.

See also

AlarmView (Page 251)

RecipeView (Page 390)

Slider (Page 410)

StatusForce (Page 415)

Switch (Page 420)

TrendView (Page 454)

Button (Page 266)

GraphicIOField (Page 324)

SymbolicIOField (Page 424)

Font (RT Professional)**Description**

Specifies the font.

The font object has the following sub-properties

- Size (Font Size)
- Bold (yes/no)
- Name (font name)
- Italic (yes/no)
- Underline (underline yes/no)
- StrikeThrough (yes/no)

If two font properties are directly assigned, only the default property "Name" is assumed.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**Font**.**[Size|Bold|Name|Italic|Underline|StrikeThrough]**[=Value]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- Bar
- Clock
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- Slider
- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- Button
- CheckBox
- ComboBox
- DateTimeField
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- ProtectedAreaNameView
- RangeLabelView
- RecipeView
- RoundButton
- SmartClientView
- Switch
- SymbolicIOField
- TextField
- TrendView
- ZoneLabelView

Value

Optional. A value or a constant which specifies the selected subproperty.

Example

```
'VBS74
Dim objControl1
Dim objControl2
Set objControl1 = ScreenItems("Control1")
Set objControl2 = ScreenItems("Control2")
objControl2.Font.Name = objControl1.Font.Name ' take over only the type of font
```

See also

[AlarmControl \(Page 236\)](#)
[Clock \(Page 287\)](#)
[FunctionTrendControl \(Page 309\)](#)
[OnlineTableControl \(Page 348\)](#)
[OnlineTrendControl \(Page 360\)](#)
[Slider \(Page 410\)](#)
[TrendRulerControl \(Page 444\)](#)
[UserArchiveControl \(Page 469\)](#)
[Bar \(Page 258\)](#)
[Button \(Page 266\)](#)
[CheckBox \(Page 275\)](#)
[ComboBox \(Page 290\)](#)
[DateTimeField \(Page 296\)](#)
[IOField \(Page 332\)](#)
[Listbox \(Page 340\)](#)
[MultiLineEdit \(Page 345\)](#)
[OptionGroup \(Page 372\)](#)
[ProtectedAreaNameView \(Page 386\)](#)
[RangeLabelView \(Page 387\)](#)
[RecipeView \(Page 390\)](#)
[RoundButton \(Page 399\)](#)
[SmartClientView \(Page 413\)](#)
[Switch \(Page 420\)](#)
[SymbolicIOField \(Page 424\)](#)
[TextField \(Page 440\)](#)
[TrendView \(Page 454\)](#)
[ZoneLabelView \(Page 490\)](#)

FontBold (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the text is shown in bold.

Access in runtime: Read and write

Syntax

Object.**FontBold**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only.

BOOLEAN

Optional. TRUE if the text is shown in bold.

FontItalic (RT Professional)

Description

Specifies whether the text is shown in italics.

Access in runtime: Read and write

Syntax

Object.**FontItalic**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only

BOOLEAN

Optional. TRUE if the text is shown in italics.

See also

TextField (Page 440)

Button (Page 266)

CheckBox (Page 275)

ComboBox (Page 290)

IOField (Page 332)

Listbox (Page 340)

MultiLineEdit (Page 345)

OptionGroup (Page 372)

RoundButton (Page 399)

SymbolicIOField (Page 424)

FontName (RT Professional)**Description**

Specifies the font.

Access in runtime: Read and write

Syntax

Object.**FontName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only

You have no access in runtime with the following format:

- MultiLineEdit

STRING

Optional. A value or a constant that specifies the font.

See also

TextField (Page 440)
Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
ComboBox (Page 290)
IOField (Page 332)
Listbox (Page 340)
MultiLineEdit (Page 345)
OptionGroup (Page 372)
RoundButton (Page 399)
SymbolicIOField (Page 424)

FontSize (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the font size of the text.

Access in runtime: Read and write

Syntax

Object.**FontSize**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only.

Int32

Optional. A value or a constant that specifies the font size of the text.

FontUnderline (RT Professional)

Description

Specifies whether the text is shown underlined.

Access in runtime: Read and write

Syntax

Object.**FontUnderline**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*: Read access only

BOOLEAN

Optional. TRUE if the text is shown underlined.

ForeColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the font color of the text.

Access in runtime Read and write

Syntax

Object.**ForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox **
- ComboBox **
- DateTimeField *
- IOField
- ListBox **
- MultiLineEdit **
- OptionGroup **

- RecipeView *
- RoundButton ***
- Slider
- Switch *
- SymbolLibrary
- SymbolicIOField ****
- TextField

* RT Professional No access

** RT Advanced No access

*** RT Advanced No access, RT Professional read access only

**** RT Professional Read access only

You have no access in runtime with the following format:

- AlarmView

Color

Optional. A value or a constant that specifies the font color of the text.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Button (Page 266)

CheckBox (Page 275)

ComboBox (Page 290)

DateTimeField (Page 296)

IOField (Page 332)

Listbox (Page 340)

MultiLineEdit (Page 345)

OptionGroup (Page 372)

RecipeView (Page 390)

RoundButton (Page 399)

Slider (Page 410)
Switch (Page 420)
SymbolicIOField (Page 424)
SymbolLibrary (Page 429)
TextField (Page 440)
AlarmView (Page 251)

ForeColorTransparency (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Format (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

FormatPattern (RT Professional)

Description

Specifies the format of the output value.

Access in runtime: Read and write

Syntax

Object.**FormatPattern**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- IOField

STRING

Optional. A value or a constant that specifies the format of the output value.

See also

IOField (Page 332)

FrameColor (RT Professional)

Description

Specifies the background color of the gauge.

Also set "background graphic" to "none".

Access in Runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**FrameColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the background color of the gauge.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

Free (RT Professional)

Description

No access in runtime.

FreePercent (RT Professional)

Description

Returns the measured values for the free disk space as a percentage. The values can be queried in Runtime. The values cannot be predefined.

Access in runtime: Read

Syntax

Object.**FreePercent**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that returns the measured values for the free disk space as a percentage.

See also

DiskSpaceView (Page 298)

1.5.5.6 Properties G-H (Panels, Comfort Panels, RT Advanced, RT Professional)

Gradation (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the value difference between two main marking lengths of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**Gradation**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that specifies the value difference.

See also

Gauge (Page 320)

GraphDirection (RT Professional)**Description**

Specifies at which border of the trend view the current values are displayed.

Access in runtime Read and write

Syntax

Object.**GraphDirection**[=GraphDirection]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

GraphDirection

Optional. A value or a constant that specifies at which margin of the trend view the current values are displayed.

Value	Explanation
0	Positive values run to the right and upwards.
-1	Positive values run to the left and upwards.
-2	Positive values run to the right and upwards.
-3	Positive values run to the right and downwards.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

GridlineAxis (RT Advanced)**Description**

No access in runtime.

GridLineColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color for the grid lines.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Read and write

Syntax

Object.**GridLineColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- AlarmView*
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

* RT Advanced read and write access, RT Professional no access

You have no access in runtime with the following format:

- TrendView

Color

Optional. A value or a constant that specifies the color of the grid lines.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

AlarmView (Page 251)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TrendView (Page 454)

GridlineEnabled (RT Advanced)

Description

No access in runtime.

GridlineFillColor (RT Advanced)

Description

No access in runtime.

GridlineStyle (RT Advanced)

Description

No access in runtime.

GridLineWidth (RT Professional)

Description

Defines the line weight of the row/column dividers in pixels.

Access in runtime: Read and write

Syntax

Object.**GridLineWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant which specifies the width of the grid line.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

GSCRuntimeAllowed (RT Professional)

Description

No access in runtime.

HeaderFont (RT Advanced)

Description

No access in runtime.

Height (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the height.

Access in runtime: Read and write

Syntax

Object.**Height**[=Int32]

Object

Required. An object of the type "ScreenItem". This property is a standard property of the ScreenItem object and is therefore available for all formats.

You have no access in "Runtime Advanced" and "Panel Runtime" with the following formats:

- AlarmView
- Bar
- BatteryView
- Button
- CameraControl
- Clock
- DateTimeField
- FunctionTrendControl
- Gauge
- GraphicIOField

- GraphicView
- HTMLBrowser
- IOField
- Line
- MediaPlayer
- PDFView
- PLCCodeViewer
- Polygon
- Polyline
- ProDiagOverview
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- S7GraphOverview
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolicIOField
- SymbolLibrary
- SysDiagControl
- TextField
- TrendView
- UserView
- WlanQualityView
- ZoneLabelView
- ZoneQualityView

Int32

Optional. A value or a constant that specifies the height in pixels.

Example

The following example halves the height of all objects of the image "NewPDL1", whose name starts with "Circle":

```
'VBS75
Dim objScreen
Dim objCircle
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
'
'Searching all circles
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
If "Circle" = Left(strName, 6) Then
'
'to halve the height of the circles
Set objCircle = objScreen.ScreenItems(strName)
objCircle.Height = objCircle.Height / 2
End If
Next
```

See also

- [Line \(Page 337\)](#)
- [Polyline \(Page 382\)](#)
- [Ellipse \(Page 300\)](#)
- [Circle \(Page 279\)](#)
- [EllipseSegment \(Page 303\)](#)
- [CircleSegment \(Page 282\)](#)
- [EllipticalArc \(Page 306\)](#)
- [CircularArc \(Page 285\)](#)
- [Rectangle \(Page 396\)](#)
- [Polygon \(Page 379\)](#)
- [TextField \(Page 440\)](#)
- [IOField \(Page 332\)](#)
- [SymbolicIOField \(Page 424\)](#)
- [Button \(Page 266\)](#)
- [Switch \(Page 420\)](#)
- [GraphicView \(Page 327\)](#)
- [GraphicIOField \(Page 324\)](#)
- [Bar \(Page 258\)](#)

Clock (Page 287)
Gauge (Page 320)
Slider (Page 410)
SymbolLibrary (Page 429)
OnlineTrendControl (Page 360)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
AlarmControl (Page 236)
HTMLBrowser (Page 330)
CheckBox (Page 275)
OptionGroup (Page 372)
WindowSlider (Page 485)
Connector (Page 293)
ScreenWindow (Page 405)
DiskSpaceView (Page 298)
ChannelDiagnose (Page 274)
ScriptDiagnostics (Page 408)
ProtectedAreaNameView (Page 386)
UserView (Page 481)
TubeTeeObject (Page 467)
TubePolyline (Page 464)
TubeDoubleTeeObject (Page 462)
TubeArcObject (Page 460)
MultiLineEdit (Page 345)
MediaPlayer (Page 343)
Listbox (Page 340)
DateTimeField (Page 296)
UserArchiveControl (Page 469)
TrendRulerControl (Page 444)
AlarmView (Page 251)
BatteryView (Page 265)
RangeLabelView (Page 387)
ZoneQualityView (Page 491)
ZoneLabelView (Page 490)
WLANQualityView (Page 488)

TrendView (Page 454)
SysDiagControl (Page 431)
StatusForce (Page 415)
SmartClientView (Page 413)
RecipeView (Page 390)
RangeQualityView (Page 389)
ApplicationWindow (Page 256)
ComboBox (Page 290)
PLCCodeViewer (Page 377)
RoundButton (Page 399)
S7GraphOverview (Page 403)

HelpText (Panels, Comfort Panels, RT Advanced)

Description

Returns the tooltip that is shown in Runtime as an operating aid for the specified object.

Access in runtime: Read and write

Syntax

Object.**HelpText**[=STRING]

Object

Required. An object of the "ScreenItem" type with the format:

- Button
- DateTimeField
- GraphicIOField
- IOField
- Switch
- SymbolicIOField
- TrendView

STRING

Optional. A value or constant that specifies the contents of the tooltip that is shown in runtime as an operator aid for the specified object.

See also

Button (Page 266)
DateTimeField (Page 296)
GraphicIOField (Page 324)
IOField (Page 332)
Switch (Page 420)
SymbolicIOField (Page 424)
TrendView (Page 454)

HiddenInput (RT Professional)

Description

Specifies whether the input value or a * for each character will be shown during the input.
Access in runtime: Read and write

Syntax

Object.**HiddenInput**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- IOField

BOOLEAN

Optional TRUE, if the input value is not shown during the input. An * is shown for each character.

See also

IOField (Page 332)

HighlightColor (RT Professional)

Description

Specifies the highlight color in the GRAPH overview.
Access in runtime Read and write

Syntax

Object.**HighlightColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- S7GraphOverview

Color

Optional. A value or a constant that specifies the highlight color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

S7GraphOverview (Page 403)

HighLimitColor (RT Professional)**Description**

Specifies the color of the top or right scroll button in a scroll bar.

Access in runtime Read and write

Syntax

Object.HighLimitColor[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- WindowSlider

Color

Optional. A value or a constant that specifies the color of the top or right scroll button in a scroll bar.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

WindowSlider (Page 485)

Hitlist (RT Professional)

Description

No access in runtime.

HitlistColumnAdd (RT Professional)

Description

Creates a new alarm text block in alarm statistics. The newly created alarm text block is automatically referenced with "HitlistColumnIndex".

Access in runtime Read and write

Syntax

`Object.HitlistColumnAdd[=STRING]`

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Required. A value or a constant that specifies the name of the new alarm text block in alarm statistics with "HitlistColumnName".

See also

HitlistColumnIndex (Page 732)

AlarmControl (Page 236)

HitlistColumnCount (RT Professional)

Description

Specifies the number of configured alarm text blocks in alarm statistics.

Access in runtime Read and write

Syntax

Object.**HitlistColumnCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the number of configured alarm text blocks in alarm statistics.

See also

AlarmControl (Page 236)

HitlistColumnIndex (RT Professional)

Description

References an alarm text block selected for alarm statistics. To access the properties of a alarm text block, you need to set "HitlistColumnIndex".

Values between 0 and "HitlistColumnIndex minus 1 are valid for "HitlistColumnCount". The "HitlistColumnCount" property specifies the number of configured alarm text blocks.

Access in runtime Read and write

Syntax

Object.**HitlistColumnIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies, with the index, the alarm text block to be edited.

See also

HitlistColumnCount (Page 730)

AlarmControl (Page 236)

HitlistColumnName (RT Professional)

Description

Specifies the name of an alarm statistics alarm text block referenced with "HitlistColumnIndex". You cannot edit this name.

Access in runtime Read and write

Syntax

Object.HitlistColumnName[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the name of the referenced alarm text block in alarm statistics.

See also

HitlistColumnIndex (Page 731)

AlarmControl (Page 236)

HitlistColumnRemove (RT Professional)

Description

Removes an alarm statistics alarm text block referenced with its name.

Access in runtime Read and write

Syntax

Object.HitlistColumnRemove[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the name of the referenced alarm statistics alarm text block to be removed.

See also

AlarmControl (Page 236)

HitlistColumnRepos (RT Professional)**Description**

Specifies the position of an alarm text block referenced with "HitlistColumnIndex" in alarm statistics.

If you have changed the position of the alarm text block with "HitlistColumnRepos", "HitlistColumnIndex" is assigned the value of "HitlistColumnRepos".

Access in runtime Read and write

Syntax

Object.**HitlistColumnRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the position of the referenced alarm text block in alarm statistics. Value range from 0 to "HitlistColumnCount" minus 1. Values outside this range have no effect.

0: The referenced alarm text block is positioned on the left.

See also

HitlistColumnIndex (Page 731)

HitlistColumnCount (Page 730)

AlarmControl (Page 236)

HitlistColumnSort (RT Professional)**Description**

Specifies the sorting order in the hit list for the alarm text block that is referenced in "HitlistColumnIndex".

Access in runtime Read and write

Syntax

Object.**HitlistColumnSort**[=SortMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

SortMode

Optional. A value or a constant that specifies which runtime data of the control can be exported.

Value	Explanation
0	No sorting
1	Ascending sorting from the smallest to largest value
2	Descending sorting from the largest to smallest value

See also

AlarmControl (Page 236)

HitlistColumnSortIndex (RT Professional)

Description

Specifies the sorting order of the alarm statistics alarm text block referenced with "HitlistColumnIndex".

If you set the value to "0", the sorting criterion in "HitlistColumnSort" is removed.

Access in runtime Read and write

Syntax

Object.**HitlistColumnSortIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. Specifies the sorting order of the referenced alarm statistics alarm text block.

See also

AlarmControl (Page 236)

HitlistColumnVisible (RT Professional)

Description

Specifies whether the alarm text block referenced with "HitlistColumnIndex" is displayed in alarm statistics.

Access in runtime Read and write

Syntax

Object.**HitlistColumnVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the referenced alarm statistics alarm text block is displayed.

FALSE if the referenced alarm statistics alarm text block is not displayed.

See also

HitlistColumnIndex (Page 731)

AlarmControl (Page 236)

HitlistDefaultSort (RT Professional)

Description

Specifies the default sorting order in the table columns of alarm statistics.

Access in runtime Read and write

Syntax

Object.**HitlistDefaultSort**[=SortMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

SortMode

Optional. A value or a constant that specifies the default sorting order in the table columns of alarm statistics.

Value	Explanation
0	The list is sorted in ascending order of frequency.
1	The list is sorted in descending order of frequency.

See also

AlarmControl (Page 236)

HitlistFilter (RT Professional)**Description**

No access in runtime.

HitlistMaxSourceItems (RT Professional)**Description**

Specifies the maximum number of data records that can be used from the alarm log to create alarm statistics. The value can be anything from "0 - 10000".

Access in runtime Read and write

Syntax

Object.**HitlistMaxSourceItems**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the maximum number of data records that can be used from the alarm log to create alarm statistics.

See also

AlarmControl (Page 236)

HitlistMaxSourceItemsWarn (RT Professional)

Description

Specifies that a warning is output when the maximum number of data records in the alarm view is reached.

Access in runtime Read and write

Syntax

Object.**HitlistMaxSourceItemsWarn**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

TRUE if an alarm is output as soon as the maximum number of data records in the alarm view is reached.

FALSE if no alarm is output when the maximum number of data records in the alarm view is reached.

See also

AlarmControl (Page 236)

HitListRelTime (RT Professional)

Description

Specifies whether the calculation period for alarm statistics is limited.

Access in runtime Read and write

Syntax

Object.**HitListRelTime** [=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

TRUE if the specified time range is used for statistics.

FALSE if the specified time range is not used.

See also

AlarmControl (Page 236)

HitListRelTimeFactor (RT Professional)**Description**

Specifies the time factor which, alongside the time type "HitlistRelTimeFactorType (Page 739)", determines the time period for which the alarm statistics are created.

Access in runtime: Read and write

Syntax

Object.**HitListRelTimeFactor**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the time factor.

See also

AlarmControl (Page 236)

HitlistRelTimeFactorType (RT Professional)**Description**

Specifies the time unit that, together with the time factor "HitlistRelTimeFactor", defines the time range displayed for alarm statistics.

Access in runtime Read and write

Syntax

Object.**HitListRelTimeFactorType**[=AlarmControlTimeUnit)

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

AlarmControlTimeUnit

Optional. A value or a constant that specifies the time unit.

Value	Designation
0	Minute
1	Hour
2	Day
3	Week
4	Month

See also

AlarmControl (Page 236)

HomeButtonVisible (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

HorizontalAlignment (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the horizontal alignment of the text.

Access in runtime Read and write

Syntax

Object.**HorizontalAlignment**[=HorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox *
- ComboBox *
- DateTimeField **
- IOField
- ListBox *
- MultiLineEdit *
- OptionGroup *

- RoundButton ***
- Switch **
- SymbolicIOField ****
- TextField

* RT Advanced No access

** RT Professional No access

*** RT Advanced No access, RT Professional read access only

**** RT Professional Read access only

HorizontalAlignment

Optional. A value or a constant that specifies the horizontal alignment of the text.

Value	VB constant	Explanation
0	hmiAlignmentLeft	The text is left-justified.
1	hmiAlignmentCentered	The text is centered horizontally in the object.
2	hmiAlignmentRight	The text is right-justified in the object.

See also

Button (Page 266)

CheckBox (Page 275)

ComboBox (Page 290)

DateTimeField (Page 296)

IOField (Page 332)

Listbox (Page 340)

MultiLineEdit (Page 345)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicIOField (Page 424)

TextField (Page 440)

HorizontalGridLines (RT Professional)

Description

Specifies whether horizontal separating lines are displayed.

Access in runtime Read and write

Syntax

Object.**HorizontalGridLines**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if horizontal separation lines are displayed.

FALSE if horizontal separation lines are not displayed.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

HorizontalPictureAlignment (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

HorizontalScrollBarPosition (RT Professional)

Description

Specifies the position of the slider in the horizontal scroll bar. The picture is displayed by moving the scroll bar slider horizontally.

To display the screen as a cutout where the scroll bars are located at the left and top edge of the screen, use the LeftOffset (Page 779)" and "TopOffset (Page 1089)" properties as the source of this cutout.

Access in runtime Read and write

Syntax

Object.**HorizontalScrollBarPosition**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- ScreenWindow

Int32

Optional. A value or a constant that specifies the horizontal position of the slide in the scroll bar.

See also

ScreenWindow (Page 405)

HorizontalScrollingEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Hotkey (RT Professional)

Description

No access in runtime.

HourNeedleHeight (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the length of the hour hand.

Access in runtime Read and write

Syntax

Object.HourNeedleHeight[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

Int32

Optional. A value or a constant that specifies the length of the hour hand.

Specify the length of the hour hand as a percentage relating to the radius of the clock face.

See also

Clock (Page 287)

HourNeedleWidth (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the width of the hour hand.

Access in runtime Read and write

Syntax

Object.**HourNeedleWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

Int32

Optional. A value or a constant that specifies the width of the hour hand.

Specify the width of the hour hand as a percentage relating to double the length of the clock face.

See also

Clock (Page 287)

1.5.5.7 Properties I-J (Panels, Comfort Panels, RT Advanced, RT Professional)**IconSpace (RT Professional)****Description**

Defines the spacing between the icons and text in the table cells. The value is active if and icon and text are displayed.

Access in runtime: Read and write

Syntax

Object.**IconSpace**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional Value that specifies the spacing.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

Index (RT Professional)**Description**

Specifies the index for the selected text field.

Access in runtime: Read and write

Syntax

Object.**Index**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- CheckBox
- ComboBox
- ListBox
- OptionGroup

Int32

Optional A value or a constant that specifies the index of the selected text field.

See also

CheckBox (Page 275)
ComboBox (Page 290)
Listbox (Page 340)
OptionGroup (Page 372)

IndipendentWindow (RT Professional)**Description**

No access in runtime.

InfoArea_BackgroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InfoArea_ColumnsMovable (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InfoArea_DefaultTextColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InfoArea_ErrorTextBackgroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InfoArea_ErrorTextColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InfoArea_FocusFrameColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InfoArea_FocusFrameWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InfoArea_Font (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InfoArea_RootNodeText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InfoArea_SelectionBackgroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InfoArea_SelectionForegroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InfoArea_ShowGridLines (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InfoArea_TableHeaderBackgroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InfoArea_TableHeaderTextColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InnerBackColorOff (RT Advanced)**Description**

Specifies the color under the slider of the "Switch" object if the object is in OFF status.

Access in runtime Read and write

Syntax

Object.InnerBackColorOff[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Switch

Color

Optional. A value or a constant that specifies the color for the OFF status.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Switch (Page 420)

InnerBackColorOn (RT Advanced)

Description

Specifies the color under the slider of the "Switch" object if the object is in ON status.

Access in runtime Read and write

Syntax

Object.InnerBackColorOn[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Switch

Color

Optional. A value or a constant that specifies the color for the ON status.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Switch (Page 420)

InnerDialColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InnerDialInnerDistance (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

InnerDialOuterDistance (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InnerHeight (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies ###

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.InnerHeight[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- IOField
- RoundButton*
- SymbolicIOField*

*: Read access only

Int32

Optional. A value or a constant, which ###

InnerWidth (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies ###

Access in Runtime: Read and write

Syntax

Object.InnerWidth[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- RoundButton*

*: Read access only

Int32

Optional. A value or a constant, which ###

InputAddressText (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InputValue (RT Professional)**Description**

Specifies the input value. The value is not displayed in the object when the property is set.

To show the value entered in the object, configure the system function "SetPropertyByProperty" to the event "Object changed":

Relevant parameters	Value
Property name	Process value
Name of the property	Input value

This system function is meaningful if no tag connection is configured at the output value but the user still wants to query the specified value, for example using a script.

Access in runtime Read and write

Syntax

Object.**InputValue**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- IOField
- SymbolicIOField*

*: Read access only

Int32

Optional. A value or a constant that specifies the input value.

InspectorViewInternalColumnOrder (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

InspectorViewRowOrder (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

Instance (RT Professional)**Description**

Returns an instance of the alarm object.

IntegerDigits (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Establishes the number of integer digits (0 to 20).

Access in runtime: Read and write

Syntax

Object.**IntegerDigits**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the number of integer digits (0 to 20).

See also

Bar (Page 258)

Interval (RT Professional)

Description

Specifies the time interval for updating the displayed measured value. You enter the value in minutes.

Access in runtime: Read

Syntax

Object.Interval[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that specifies the time interval for updating the measured values.

See also

DiskSpaceView (Page 298)

IsActive (RT Professional)

Description

No access in runtime.

IsImageMiddleAligned (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IsMinPasswordValueSet (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IsRunningUnderCE (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IsVerticalScrollBarEnabled (RT Advanced)**Description**

No access in runtime.

ItemBorderStyle (RT Professional)**Description**

Specifies the line type of the separation lines in the selection list.

Access in runtime Read

Syntax

Object.**ItemBorderStyle**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicIOField

Int32

Optional. A value or a constant that specifies the type of separation lines in the selection list.

Value	VB constant	Explanation
-1	hmiLineStyleNone	The selection list has no separation lines.
0	hmiLineStyleSolid	The selection list has solid separation lines.
1	hmiLineStyleDash	The selection list has dashed separation lines.
2	hmiLineStyleDot	The selection list has dotted separation lines.
3	hmiLineStyleDashDot	The selection list has dash-dot separation lines.
4	hmiLineStyleDashDotDot	The selection list has dash-dot-dot separation lines.

See also

SymbolicIOField (Page 424)

ItemText_AKZ (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_Descriptor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_ErrorText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_HardwareRevision (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_IMDataVersion (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_InstallationDate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_LADDR (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_ManufacturerID (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_Name (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_OKZ (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_OperationState (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_OrderID (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_ProfileID (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_Rack (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_RevisionCounter (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_SerialNumber (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_Slot (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_SoftwareRevision (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_SpecificProfileData (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_State (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_Station (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ItemText_SubAddress (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_SubSlot (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_SubSystem (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ItemText_Type (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_AKZ (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_Descriptor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_ErrorText (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_HardwareRevision (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_IMDataVersion (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_InstallationDate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_LADDR (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_ManufacturerID (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_Name (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_OKZ (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_OperationState (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_OrderID (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_ProfileID (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_Rack (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_RevisionCounter (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_SerialNumber (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_Slot (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

IV_ShowItem_SoftwareRevision (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_SpecificProfileData (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_State (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_Station (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_SubAddress (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_SubSlot (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_SubSystem (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

IV_ShowItem_Type (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

JumpToLimitsAfterMouseClicked (RT Professional)**Description**

Specifies whether the slider will be set to the associated end value. The end value is the minimum or maximum value. To set the slider to the end value, users click on the area outside of the current slider setting.

Access in runtime: Read and write

Syntax

Object.**JumpToLimitsAfterMouseClicked**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- WindowSlider

BOOLEAN

Optional. TRUE, if the slider is set to the associated end value.

See also

WindowSlider (Page 485)

1.5.5.8 Properties K-L (Panels, Comfort Panels, RT Advanced, RT Professional)**KeyboardOnline (RT Advanced)****Description**

No access in runtime.

LabelColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color of the scale label.

Access in runtime Read and write

Syntax

Object.**LabelColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Slider

Color

Optional. A value or a constant that specifies the color of the scale label.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Slider (Page 410)

Language (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Determines the current Runtime language.

Access in Runtime: Read and write

Syntax

Object.**Language**[= LONG]

Object

Required A "HMIRuntime" object.

LONG

Optional A value or a constant that specifies the national code.

Comments

Specify the Runtime language in VBS with a national code, e.g. 1031 for German, 2057 for English etc.. Refer to the VBScript basics under the topic "Current local ID (LCID) diagram" for an overview of all the national codes.

See also

HMIRuntime (Page 208)

LargeTickLabelingStep (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns the interval at which scale sections are labeled.

Access in runtime: Read and write

Syntax

Object.**LargeTickLabelingStep**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Int32

Optional. A value or constant that returns the interval at which scale sections are labeled.

See also

Bar (Page 258)

LargeTicksBold (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the long marking lengths of a scale are shown in bold.

Access in runtime: Read and write

Syntax

Object.**LargeTicksBold**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the long marking lengths of a scale are shown in bold.

See also

Bar (Page 258)

LargeTicksSize (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the length of the long marking lengths of a scale.

Access in runtime: Read and write

Syntax

Object.**LargeTicksSize**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the length of the long marking lengths of a scale.

See also

Bar (Page 258)

LastConnectedObject (RT Professional)

Description

No access in runtime.

LastConnectedObjectIndex (RT Professional)

Description

Specifies the index number of the connection point to the last connected object.

Access in runtime: Read and write

Syntax

Object.**LastConnectedObjectIndex**[=Int]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector

Int

Optional. A value or a constant that specifies the index number of the connection point to the last connected object.

See also

Connector (Page 293)

LastConnectedObjectName (RT Professional)

Description

Specifies the name of the object that is docked to the lower connector point.

Access in runtime: Read and write

Syntax

Object.**LastConnectedObjectName**[=String]

Object

Required. An object of the type "ScreenItem" with the following format:

- Connector

String

Optional. A value or a constant that specifies the name of the object that is docked to the lower connector point.

See also

Connector (Page 293)

LastError (RT Professional)

Description

Returns an error code regarding the success of the last operation, e.g. information on a tag write or read process:

Code in hexadecimal notation	Description
0x00000000	OK
0x80040001	Execution error
0x80040002	Tag error
0x80040003	Server not available.
0x80040004	Multi Tag Error; error in one or several tags

To obtain an error description, first of all carry out the Read method.

Note

If the error occurs when accessing via the TagSet object, evaluate the LastError property for each tag of the TagSet object.

To obtain a statement about the quality of the supplied value, use the "QualityCode" property. To obtain a description of the error, use the "ErrorDescription" property.

Access during runtime: Read

Syntax

Object.**LastError**

Object

Necessary. A "Tag" object.

Example

The following example shows the error code for the "Tag1" tag:

```
'VBS77
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.LastError
```

The following example adds two tags to the TagSet list and outputs the LastError property as Trace:

```
'VBS178
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "LastError: " & group.LastError & vbNewLine
```

The LastError property of a tag contained in the list may be accessed as follows:

```
HMIRuntime.Trace "LastError: " & group("Motor1").LastError & vbNewLine
```

See also

Tag (Page 228)

TagSet (list) (Page 232)

Layer (RT Advanced, RT Professional)

Description

Returns the layer in the screen in which an object is located. There is a total of 32 layers available, whereby Layer "0" is the bottom layer and Layer "31" the top layer.

The configured objects are initially in the background of a layer.

Access in runtime: Read and write

Syntax

Object.**Layer**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl *
- AlarmView **
- ApplicationWindow *
- Bar
- BatteryView ***
- Button
- CameraControl ***
- ChannelDiagnose *

- CheckBox *
- Circle
- CircleSegment *
- CircularArc *
- Clock
- ComboBox *
- Connector *
- DateTimeField **
- DiscSpaceView *
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl **
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox *
- MediaPlayer
- MultiLineEdit *
- OptionGroup *
- PLCCodeViewer **
- Polygon
- Polyline
- ProDiagOverview ****
- ProtectedAreaNameView ***
- RangeLabelView ***
- RangeQualityView ***
- RecipeView **
- Rectangle
- RoundButton *****
- S7GraphOverview
- ScreenWindow *

- Slider
- SmartClientView **
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField *****
- SysDiagControl *****
- TextField
- TrendRulerControl *
- TrendView **
- TubeArcObject *
- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserView
- WlanQualityView ***
- WindowSlider *
- ZoneLabelView ***
- ZoneQualityView ***

* RT Advanced No access

** RT Professional No access

*** Only Panel RT read and write access

**** RT Advanced read access, RT Professional read and write access

***** RT Advanced No access, RT Professional read access

***** RT Advanced read and write access, RT Professional read access

***** Read access only

You have no access in runtime with the following format:

- OnlineTableControl
- OnlineTrendControl
- UserArchiveControl

Int32

Optional. A value or a constant returns the layer in the screen in which an object is located.

Comments

Note

The Layer property specifies the layer in which the object is located. The layer "0" is output as layer "0".

When accessed, the layers are counted up from 1 in VBS. Therefore, address the level "1" with layers(2).

Example

The following example displays the name and layer of all objects in the screen "NewPDL1":

```
'VBS78
Dim objScreen
Dim objScrItem
Dim lngAnswer
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
Set objScrItem = objScreen.ScreenItems(strName)
lngAnswer = MsgBox(strName & " is in layer " & objScrItem.Layer,vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
```

See also

[AlarmControl \(Page 236\)](#)
[AlarmView \(Page 251\)](#)
[ApplicationWindow \(Page 256\)](#)
[Bar \(Page 258\)](#)
[BatteryView \(Page 265\)](#)
[Button \(Page 266\)](#)
[CameraControl \(Page 272\)](#)
[ChannelDiagnose \(Page 274\)](#)
[CheckBox \(Page 275\)](#)
[Circle \(Page 279\)](#)
[CircleSegment \(Page 282\)](#)
[CircularArc \(Page 285\)](#)
[Clock \(Page 287\)](#)

ComboBox (Page 290)
Connector (Page 293)
DateTimeField (Page 296)
DiskSpaceView (Page 298)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
FunctionTrendControl (Page 309)
Gauge (Page 320)
GraphicIOField (Page 324)
GraphicView (Page 327)
HTMLBrowser (Page 330)
IOField (Page 332)
Line (Page 337)
Listbox (Page 340)
MediaPlayer (Page 343)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)
PLCCodeViewer (Page 377)
Polygon (Page 379)
Polyline (Page 382)
ProtectedAreaNameView (Page 386)
RangeLabelView (Page 387)
RangeQualityView (Page 389)
RecipeView (Page 390)
Rectangle (Page 396)
RoundButton (Page 399)
S7GraphOverview (Page 403)
ScreenWindow (Page 405)
Slider (Page 410)
SmartClientView (Page 413)
StatusForce (Page 415)
Switch (Page 420)

SymbolicIOField (Page 424)
SymbolLibrary (Page 429)
SysDiagControl (Page 431)
TextField (Page 440)
TrendRulerControl (Page 444)
TrendView (Page 454)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)
UserArchiveControl (Page 469)
UserView (Page 481)
WindowSlider (Page 485)
WLANQualityView (Page 488)
ZoneLabelView (Page 490)
ZoneQualityView (Page 491)

LayerDeclutteringEnable (RT Professional)

Description

Indicates whether the layers of a screen can be shown or hidden dependent on the set minimum and maximum zoom.

Access during runtime: Read

Syntax

Object.**LayerDeclutteringEnable**

Object

Necessary. A "Screen" object.

Example:

The example outputs the LayerDecluttering property of the "NewPDL1" screen as Trace.

```
'VBS156
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 215)

Layers (RT Professional)**Description**

Returns an object of type "Layers".

Access in Runtime: Read

Syntax

Object.**Layers**

Object

Required A "Screen" object.

See also

Screen (Page 215)

Left (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the value of the X coordinate.

Access in runtime: Read and write

Syntax

Object.**Left**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl *
- AlarmView **
- ApplicationWindow *
- Bar
- BatteryView ***
- Button
- ChannelDiagnose *
- CheckBox *

- Circle
- CircleSegment *
- CircularArc *
- Clock
- ComboBox *
- Connector *
- DateTimeField **
- DiscSpaceView *
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox *
- MediaPlayer
- MultiLineEdit *
- OnlineTableControl *
- OnlineTrendControl *
- OptionGroup *
- PLCCodeViewer **
- Polygon
- Polyline
- ProDiagOverview ***
- ProtectedAreaNameView ***
- RangeLabelView ***
- RangeQualityView ***
- RecipeView**
- Rectangle
- RoundButton *****
- S7GraphOverview **

- ScreenWindow *
- Slider
- SmartClientView **
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField *****
- SysDiagControl *****
- TextField
- TrendRulerControl *
- TrendView **
- TubeArcObject *
- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserArchiveControl *
- UserView
- WlanQualityView ***
- WindowSlider *
- ZoneLabelView ***
- ZoneQualityView ***

* RT Advanced No access

** RT Professional No access

*** Only Panel RT read and write access

**** RT Advanced read access, RT Professional read and write access

***** RT Advanced No access, RT Professional read access

***** RT Advanced read and write access, RT Professional read access

Int32

Optional. A value or a constant that contains the value of the X coordinate in pixels (measured from the top left edge of the screen).

Comments

The X coordinate refers to the top left corner of the rectangle that surrounds the object. The screen limits are also monitored in runtime. If the assigned coordinate value exceeds the display size, the user-defined function is interrupted with an error message.

See also

Bar (Page 258)
BorderWidth (Page 580)
CameraControl (Page 272)
Connector (Page 293)
AlarmControl (Page 236)
AlarmView (Page 251)
ApplicationWindow (Page 256)
BatteryView (Page 265)
Button (Page 266)
ChannelDiagnose (Page 274)
CheckBox (Page 275)
Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
Clock (Page 287)
ComboBox (Page 290)
DateTimeField (Page 296)
DiskSpaceView (Page 298)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
FunctionTrendControl (Page 309)
Gauge (Page 320)
GraphicIOField (Page 324)
GraphicView (Page 327)
HTMLBrowser (Page 330)
IOField (Page 332)
Line (Page 337)
Listbox (Page 340)
MediaPlayer (Page 343)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)

PLCCodeViewer (Page 377)
Polygon (Page 379)
Polyline (Page 382)
ProtectedAreaNameView (Page 386)
RangeLabelView (Page 387)
RangeQualityView (Page 389)
RecipeView (Page 390)
Rectangle (Page 396)
RoundButton (Page 399)
S7GraphOverview (Page 403)
ScreenWindow (Page 405)
Slider (Page 410)
SmartClientView (Page 413)
StatusForce (Page 415)
Switch (Page 420)
SymbolicOField (Page 424)
SymbolLibrary (Page 429)
SysDiagControl (Page 431)
TextField (Page 440)
TrendRulerControl (Page 444)
TrendView (Page 454)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)
UserArchiveControl (Page 469)
UserView (Page 481)
WindowSlider (Page 485)
WLanQualityView (Page 488)
ZoneLabelView (Page 490)
ZoneQualityView (Page 491)

LeftMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

LeftOffset (RT Professional)

Description

Specifies the horizontal shift of zero point of the display of a screen that is larger than the screen window. The shift refers to the left edge of the screen window.

The screen is displayed cut. The picture scroll bars are located at the left and upper edge of the picture.

If you wish to display the screen in the screen window with horizontal and vertical offset of the screen scroll bars, use the "HorizontalScrollBarPosition (Page 741)" and "VerticalScrollbarPosition (Page 1206)" properties for the offset.

Access in Runtime: Read and write

Syntax

Object.**LeftOffset**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Screenwindow

Int32

Optional. A value or a constant that specifies the horizontal shift of the screen display's zero point from the left margin of the screen window.

See also

ScreenWindow (Page 405)

Limit4LowerLimit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the lower limit for ""Reserve4"".

The "Limit4LowerLimitEnabled" property must be set to TRUE so that the limit "Reserve4"" can be monitored.

Access in runtime: Read and write

Syntax

Object.Limit4LowerLimit[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the low limit for "Reserve4".

Comments

The "Limit4LowerLimitRelative" property specifies whether the object is evaluated as a percentage or absolutely.

See also

Bar (Page 258)

Limit4LowerLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the low limit "Reserve4".

The "Limit4LowerLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime Read and write

Syntax

Object.Limit4LowerLimitColor[=Color]

Object

Required. An object of the type ""ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the lower limit ""Reserve4".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Limit4LowerLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the lower limit `""Reserve4""` is to be monitored.

Access in runtime: Read and write

Syntax

`Object.Limit4LowerLimitEnabled[=BOOLEAN]`

Object

Required. A `"ScreenItem"` object with the following format:

- Bar

BOOLEAN

Optional `TRUE`, if the lower limit `""Reserve4""` is to be monitored.

Comments

The following values will be specified with the properties `""Limit4LowerLimit""`, `""Limit4LowerLimitColor""` and `""Limit4LowerLimitRelative""`:

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

Limit4LowerLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the low limit `""Reserve4""` is evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.Limit4LowerLimitRelative[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BOOLEAN

Optional.

TRUE if the low limit "Reserve4" is to be evaluated as a percentage.

FALSE if the low limit "Reserve4" is to be evaluated as an absolute value.

See also

Bar (Page 258)

Limit4UpperLimit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the lower limit for "Reserve4".

The "Limit4UpperLimitEnabled"" property must be set to TRUE so that the limit ""Reserve4" can be monitored.

Access in runtime: Read and write

Syntax

Object.Limit4UpperLimit[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the upper limit for "Reserve4".

Comments

The "Limit4UpperLimitRelative" property specifies whether the object is evaluated as a percentage or absolutely.

See also

Bar (Page 258)

Limit4UpperLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the upper limit "Reserve4".

The "Limit4UpperLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime Read and write

Syntax

Object.Limit4UpperLimitColor[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the upper limit "Reserve4".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Limit4UpperLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the upper limit "Reserve4" is to be monitored.

Access in runtime: Read and write

Syntax

Object.Limit4UpperLimitEnabled[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the upper limit ""Reserve4" is to be monitored.

Comments

The following values are defined via the properties ""Limit4UpperLimit", "Limit4UpperLimitColor" and "Limit4UpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

Limit4UpperLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the high limit "Reserve4" is to be evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.Limit4UpperLimitRelative[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BOOLEAN

Optional.

TRUE if the low limit "Reserve4" is to be evaluated as a percentage.

FALSE if the low limit "Reserve4" is to be evaluated as an absolute value.

See also

Bar (Page 258)

Limit5LowerLimit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the lower limit for "Reserve5".

The "Limit5LowerLimitEnabled" property must be set to TRUE so that the limit ""Reserve5" can be monitored.

Access in runtime: Read and write

Syntax

Object.Limit5LowerLimit[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the lower limit for "Reserve5".

Comments

The "Limit5LowerLimitRelative" property specifies whether the object is evaluated as a percentage or absolutely.

See also

Bar (Page 258)

Limit5LowerLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the lower limit ""Reserve5".

The "Limit5LowerLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime Read and write

Syntax

Object.Limit5LowerLimitColor[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the lower limit ""Reserve5".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Limit5LowerLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the lower limit "Reserve5" is to be monitored.

Access in runtime: Read and write

Syntax

Object.Limit5LowerLimitEnabled[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the lower limit ""Reserve5" is to be monitored.

Comments

The following values will be defined via the properties "Limit5LowerLimit", "Limit5LowerLimitColor" and "Limit5LowerLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

Limit5LowerLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the lower limit ""Reserve5" is to be evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.Limit5LowerLimitRelative[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- Bar

BOOLEAN

Optional.

TRUE if the low limit ""Reserve5" is to be evaluated as a percentage.

FALSE if the low limit ""Reserve5" is to be evaluated as an absolute value.

See also

Bar (Page 258)

Limit5UpperLimit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the lower limit for "Reserve5".

The ""Limit5UpperLimitEnabled" property must be set to TRUE so that the limit ""Reserve5" can be monitored.

Access in runtime: Read and write

Syntax

Object.Limit5UpperLimit[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the upper limit for ""Reserve5".

Comments

The "TypeLimitHigh5" property specifies whether the object is evaluated as a percentage or absolutely.

See also

Bar (Page 258)

Limit5UpperLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the upper limit "Reserve5".

The "Limit5UpperLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime Read and write

Syntax

Object.Limit5UpperLimitColor[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the upper limit ""Reserve5".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

Limit5UpperLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the upper limit "Reserve5" is to be monitored.

Access in runtime: Read and write

Syntax

Object.Limit5UpperLimitEnabled[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the upper limit ""Reserve5"" is to be monitored.

See also

Bar (Page 258)

Limit5UpperLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the high limit "Reserve5" is to be evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.Limit5UpperLimitRelative[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BOOLEAN

Optional.

TRUE if the high limit "Reserve5" is to be evaluated as a percentage.

FALSE if the high limit "Reserve5" is to be evaluated as an absolute value.

See also

Bar (Page 258)

LimitRangeCollection (RT Professional)**Description**

No access in runtime.

LineAlarmView (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

LineBackgroundColor (RT Professional)**Description**

No access in runtime.

LineColor (RT Professional)**Description**

Specifies the color of the window separation lines.

Access in runtime Read and write

Syntax

Object.LineColor[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color of the window separation lines.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

LineEndShapeStyle (RT Professional)**Description**

Specifies the shape of the line ends.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**LineEndShapeStyle**[=LineEndShapeStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton *
- SymbolicIOField *
- TextField
- WindowSlider

* Read access only

You have no access in runtime with the following format:

- Switch
- TubeArcObject

LineEndShapeStyle

Optional. A value or a constant that specifies the form of the line ends.

Value	Description
0	Rounded
1	Flush
2	Rectangular

See also

Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
ComboBox (Page 290)
Connector (Page 293)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
GraphicIOField (Page 324)
GraphicView (Page 327)
IOField (Page 332)
Line (Page 337)
Listbox (Page 340)
MultiLineEdit (Page 345)
OptionGroup (Page 372)
Polygon (Page 379)
Polyline (Page 382)
Rectangle (Page 396)
RoundButton (Page 399)
Switch (Page 420)
SymbolicIOField (Page 424)
TextField (Page 440)
TubeArcObject (Page 460)
WindowSlider (Page 485)

LinesPerDiagEntry (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

LineStyle (RT Professional)**Description**

No access in runtime.

LineWidth (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the line thickness.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**LineWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- CircularArc
- Connector
- EllipticalArc
- FunctionTrendControl
- Line *
- OnlineTableControl
- OnlineTrendControl
- Polyline *
- TrendRulerControl
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject

- Tubepolyline
 - UserArchiveControl
- * RT Advanced read and write access

You have no access in runtime with the following format:

- Polygon

Int32

Optional. A value or a constant that specifies the line weight in pixels.

See also

AlarmControl (Page 236)
CircularArc (Page 285)
EllipticalArc (Page 306)
FunctionTrendControl (Page 309)
Line (Page 337)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
Polyline (Page 382)
TrendRulerControl (Page 444)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)
UserArchiveControl (Page 469)
Polygon (Page 379)

LineWrap (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ListAreaHeight (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ListAreaLeft (RT Advanced)**Description**

No access in runtime.

ListAreaTop (RT Advanced)**Description**

No access in runtime.

ListAreaWidth (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

LoadDataImmediately (RT Professional)**Description**

Specifies whether in the event of a screen refresh the tag values should be downloaded from the archives for the time range that is to be shown.

Access in runtime: Read and write

Syntax

Object.**LoadDataImmediately**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

BOOLEAN

Optional TRUE, if in the event of a screen the tag values should be downloaded from the archives for the time range that is to be shown.

See also

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

LocalCursor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Location (RT Professional)

Description

No access in runtime.

LockSquaredExtent (RT Professional)

Description

Specifies whether the size of the clock can be adjusted with the mouse.

Access in runtime: Read and write

Syntax

Object.**LockSquaredExtent**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Clock
- Gauge

BOOLEAN

Optional TRUE, if the size of the clock can be adjusted by dragging the mouse on the selection points to the desired aspect ratio.

See also

Clock (Page 287)

Gauge (Page 320)

Logging (RT Professional)

Description

Returns an object of type "Logging".

Access in Runtime: Read

Syntax

Object.**Logging**

Object

Required A "HMIRuntime" object.

See also

HMIRuntime (Page 208)

LogOperation (RT Professional)

Description

Specifies whether, after operating this object, an alarm is output on the alarm system.

Access in runtime: Read and write

Syntax

Object.**LogOperation**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- CheckBox
- ComboBox
- IOField
- ListBox
- OptionGroup
- SymbolicIOField *
- WindowSlider

*: Read access only

BOOLEAN

Optional TRUE, if, after operating this object, an alarm is output on the alarm system.

LongDateTimeFormat (RT Advanced)

Description

No access in runtime.

LongTermArchiveConsistency (RT Professional)

Description

Specifies how alarms are displayed in the alarm view when the "Historical alarm list (long-term)" is configured.

Access in runtime Read and write

Syntax

Object.**LongTermArchiveConsistency**[= BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the 1000 most recent alarms are displayed on the client of all servers or the redundant server pair in the "Historical alarm list" (long-term) display.

FALSE if 1000 alarms are displayed in the "Historical alarm list" (long-term) display on the single-user system, server or client for each server, or for each redundant server pair.

See also

AlarmControl (Page 236)

Look3D (RT Advanced)

Description

No access in runtime.

LowerLimit (RT Professional)

Description

Specifies the lower limit for input values.

Access in runtime: Read and write

Syntax

Object.**LowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- IOField

DOUBLE

Optional A value or a constant that specifies the lower limit for input values.

See also

IOField (Page 332)

LowLimitColor (RT Professional)

Description

Specifies the color of the bottom or left scroll button in a scroll bar.

Access in runtime Read and write

Syntax

Object.**LowLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- WindowSlider

Color

Optional. A value or a constant that specifies the color of the bottom or left scroll button in a scroll bar.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

WindowSlider (Page 485)

1.5.5.9 Properties M-N (Panels, Comfort Panels, RT Advanced, RT Professional)

Machine (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

MachineName (RT Advanced)

Description

Specifies the network code of the device that is to be monitored.

Enter the name or the port of the device as the network code.

Access in runtime: Read and write

Syntax

Object.**MachineName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- SmartClientView

STRING

Optional A value or a constant that contains the network code.

See also

SmartClientView (Page 413)

MaintainAspectRatio (RT Professional)

Description

No access in runtime.

MaintainOriginalSize (RT Professional)**Description**

No access in runtime.

MarginToBorder (RT Professional)**Description**

Specifies the width of the 3D border in pixels. The value of the width is dependent on the size of the object.

Access in runtime: Read and write

Syntax

Object.**MarginToBorder**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- WindowSlider

Int32

Optional A value or a constant that specifies the width of the 3D border in pixels.

See also

WindowSlider (Page 485)

MaximumNumberOfTimeAxes (RT Professional)**Description**

No access in runtime.

MaximumNumberOfTimeColumns (RT Professional)**Description**

No access in runtime.

MaximumNumberOfValueAxes (RT Professional)

Description

No access in runtime.

MaximumNumberOfValueColumns (RT Professional)

Description

No access in runtime.

MaximumValue (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the maximum value of the scale in the selected object.

Access in runtime: Read and write

Syntax

Object.**MaximumValue**[=DOUBLE | Int32 | SINGLE]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Gauge
- Slider
- WindowSlider*

*: Read access only

DOUBLE | Int32 | SINGLE

Optional A value or a constant that specifies the maximum value. The data type depends on the format.

- DOUBLE: Bar
- Int32: Slider, WindowsSlider
- SINGLE: Gauge

MaxNrOfCurves (RT Advanced)**Description**

No access in runtime.

MaxNumberOfComboBoxCharacters (RT Advanced)**Description**

No access in runtime.

MaxToolBarRows (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

MenuButtonVisible (RT Advanced)**Description**

No access in runtime.

MenuToolBarConfig (RT Professional)**Description**

Determines the configuration file with user-defined menu and toolbars.

Access in Runtime: Read and write

Syntax

Object.**MenuToolBarConfig**[=HmiObjectHandle]

Object

Required. An object of the type "ScreenItem" with the format:

- ScreenWindow

HmiObjectHandle

Optional. A value or a constant that specifies the configuration file with user-defined menu and toolbars.

MessageAreaHeight (RT Advanced)

Description

No access in runtime.

MessageAreaLeft (RT Advanced)

Description

No access in runtime.

MessageAreaTop (RT Advanced)

Description

No access in runtime.

MessageAreaWidth (RT Advanced)

Description

No access in runtime.

MessageBlockAlignment (RT Professional)

Description

Specifies the designation of an alarm text block referenced with "MessageBlockIndex" in the column header.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockAlignment** [=HorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

HorizontalAlignment

Optional. A value or constant that determines the alignment.

Value	Designation	Description
0	Left	The designation of the block in the column header is left-justified.
1	Centered	The designation of the block in the column header is centered.
2	Right	The designation of the block in the column header is right-justified.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockAutoPrecisions (RT Professional)**Description**

Specifies whether the number of decimal places for an alarm text block referenced with "MessageBlockIndex" is automatically determined.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockAutoPrecisions**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the number of decimal places is automatically determined.

FALSE if the value of "MessageBlockPrecisions" is used.

See also

MessageBlockIndex (Page 812)

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockCaption (RT Professional)

Description

Specifies the designation in the column header for an alarm text block referenced with "MessageBlockIndex". The designation entered is effective in all runtime languages.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockCaption**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies the designation in the column header for the selected alarm text block.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockCount (RT Professional)

Description

Specifies the number of configured alarm text blocks.

Access in runtime Read and write

Syntax

Object.**MessageBlockCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the number of configured alarm text blocks.

See also

AlarmControl (Page 236)

MessageBlockDateFormat (RT Professional)**Description**

Specifies the date format for displaying the alarms.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockDateFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the date format used for displaying alarms.

Value	Explanation
Automatic	The date format is set automatically.
dd.MM.yy	Day.Month.Year, e.g. 24.12.10.
dd.MM.yyyy	Day.Month.Year, e.g. 24.12.2010.
dd/MM/yy	Day/Month/Year, e.g. 24/12/10.
dd/MM/yyyy	Day/Month/Year, e.g. 24/12/2010.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockExponentialFormat (RT Professional)**Description**

Specifies whether the values of an alarm text block referenced with "MessageBlockIndex" are displayed in exponential notation.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockExponentialFormat**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the values are displayed in exponential notation.

FALSE if the values are displayed in decimal notation.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockFlashOn (RT Professional)

Description

Specifies whether the content of an alarm text block referenced with "MessageBlockIndex" flashes in runtime when an alarm appears.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockFlashOn**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the content of the alarm text block flashes.

FALSE if the content of the alarm text block does not flash.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockHideText (RT Professional)**Description**

Specifies whether the content of an alarm text block referenced with "MessageBlockIndex" is displayed as text.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockHideText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the content is not displayed as text.

FALSE if the content is displayed as text.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockHideTitleText (RT Professional)**Description**

Specifies whether the header of an alarm text block referenced with "MessageBlockIndex" is displayed as text.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockHideTitleText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the header is not displayed as text.

FALSE if the header is displayed as text.

See also

AlarmControl (Page 236)

MessageBlockId (RT Professional)**Description**

Specifies the assignment of ID number and alarm text block in the alarm view.

Access in runtime: Read and write

Syntax

Object.**MessageBlockId**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Specifies the assignment of ID number and alarm text block in the alarm view.

See also

AlarmControl (Page 236)

MessageBlockIndex (RT Professional)

Description

References an alarm text block. To access the properties of an alarm text block, you need to set "MessageBlockIndex".

Values between 0 and "MessageBlockIndex" minus 1 are valid for "MessageBlockCount". The "MessageBlockCount" property specifies the number of configured alarm text blocks.

Access in runtime Read and write

Syntax

Object.**MessageBlockIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies, with the index, the alarm text block to be edited.

See also

MessageBlockCount (Page 806)

AlarmControl (Page 236)

MessageBlockLeadingZeros (RT Professional)

Description

Specifies whether the entries in an alarm text block referenced with "MessageBlockIndex" are displayed with leading zeros.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockLeadingZeros**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional.

1 if the entries in the alarm text block are displayed with leading zeros.

0 if the entries in the alarm text block are displayed without leading zeros.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockLength (RT Professional)

Description

Specifies the length in characters of the content of an alarm text block referenced with "MessageBlockIndex".

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockLength**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. Value which specifies the length of the alarm text block.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockName (RT Professional)

Description

Specifies the name of an alarm text block referenced with "MessageBlockIndex".

Access in runtime: Read and write

Syntax

Object.**MessageBlockName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the name of the referenced alarm text block.

MessageBlockPrecisions (RT Professional)

Description

Specifies the number of decimal places for values in an alarm text block referenced with "MessageBlockIndex".

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockPrecisions**[=Int16]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int16

Optional. Specifies the number of decimal places.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockSelected (RT Professional)

Description

Specifies whether an alarm text block referenced with "MessageBlockIndex" is selected.

Also set "ApplyProjectSettings (Page 515)" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockSelected**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional. TRUE if the alarm text block is selected.

See also

AlarmControl (Page 236)

MessageBlockShowDate (RT Professional)

Description

Specifies whether to display both the date and time in the "Time" alarm text block.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockShowDate**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if both date and time are displayed.

FALSE if the time is displayed.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockShowIcon (RT Professional)

Description

Specifies whether the content of an alarm text block referenced with "MessageBlockIndex" is displayed as an icon.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockShowIcon**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the content is displayed as an icon.

FALSE if the content is not displayed as an icon.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockShowTitleIcon (RT Professional)

Description

Specifies whether the header of an alarm text block referenced with "MessageBlockIndex" is displayed as text.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockShowTitleIcon**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the header is shown as an icon.

FALSE if the header is not shown as an icon.

See also

[ApplyProjectSettings](#) (Page 515)

[AlarmControl](#) (Page 236)

MessageBlockTextId (RT Professional)**Description**

Specifies the designation of an alarm text block referenced with "MessageBlockIndex" using a text ID from the text library. The designation is adapted automatically if the runtime language is changed.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime: Read and write

Syntax

Object.**MessageBlockTextId**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- [AlarmControl](#)

Int32

Optional. Specifies the name of the selected alarm text block using a text ID number.

MessageBlockTimeFormat (RT Professional)**Description**

Specifies which time format or duration format is used for displaying alarms.

Also set "ApplyProjectSettings" to "FALSE".

Access in runtime Read and write

Syntax

Object.**MessageBlockTimeFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the time format or duration format used for displaying alarms.

The following time formats are available:

Value	Explanation
Automatic	The time format is set automatically.
HH:mm:ss	Hours:Minutes:Seconds, e.g. 15:35:44
HH:mm:ss.ms	Hours:Minutes:Seconds.Milliseconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.

The following duration formats are available:

Value	Explanation
Automatic	The duration format is determined automatically.
d H:mm:ss	Day Hours:Minutes:Seconds, e.g. 1 2:03:55.
H:mm:ss.	Hours:Minutes:Seconds, e.g. 26:03:55.
m:ss	Minutes:Seconds, Example: 1563:55.
s	Seconds, e.g. 93835.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageBlockType (RT Professional)**Description**

Returns the type of an alarm text block referenced with "MessageBlockIndex".

Access in runtime Read and write

Syntax

Object.**MessageBlockType**[=AlarmBlockType]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

AlarmBlockType

Optional. A value or a constant that returns the type of alarm text block.

Value	Explanation
0	The alarm text block is a system block.
1	The alarm text block is a user text block.
2	The alarm text block is a parameter block.
3	The alarm text block belongs to the alarm text blocks of the alarm statistics.

See also

ApplyProjectSettings (Page 515)

AlarmControl (Page 236)

MessageColumnAdd (RT Professional)**Description**

Creates a new alarm text block. The newly created alarm text block is automatically referenced with "MessageColumnIndex".

Access in runtime Read and write

Syntax

Object.**MessageColumnAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Required. A value or a constant that specifies the name of the new alarm text block with "MessageColumnName".

See also

MessageColumnIndex (Page 820)

AlarmControl (Page 236)

MessageColumnCount (RT Professional)**Description**

Specifies the number of configured alarm text blocks.

Access in runtime Read and write

Syntax

Object.**MessageColumnCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the number of configured alarm text blocks.

See also

AlarmControl (Page 236)

MessageColumnIndex (RT Professional)

Description

References an alarm text block selected for the alarm list. To access the properties of an alarm text block, you need to set "MessageColumnIndex".

Values between 0 and "MessageColumnIndex" minus 1 are valid for "MessageColumnCount". The "MessageColumnCount" property specifies the number of configured alarm text blocks.

Access in runtime Read and write

Syntax

Object.**MessageColumnIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies, with the index, the alarm text block selected for the alarm list.

See also

MessageColumnCount (Page 818)

AlarmControl (Page 236)

MessageColumnName (RT Professional)

Description

Specifies the name of an alarm text block referenced with "MessageColumnIndex".

Access in runtime Read and write

Syntax

Object.**MessageColumnName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the name of the referenced alarm text block.

See also

MessageColumnIndex (Page 819)

AlarmControl (Page 236)

MessageColumnRemove (RT Professional)

Description

Removes an alarm text block referenced with its name.

Access in runtime Read and write

Syntax

Object.**MessageColumnRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the name of the referenced alarm text block to be removed.

See also

AlarmControl (Page 236)

MessageColumnRepos (RT Professional)**Description**

For multiple alarm text blocks, specifies the position of an alarm text block referenced with "MessageColumnIndex".

If you have changed the position of the alarm text block with "MessageColumnRepos", "MessageColumnIndex" is assigned the value of "MessageColumnRepos".

Access in runtime Read and write

Syntax

Object.**MessageColumnRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the position of the referenced alarm text block. Value range from 0 to "MessageColumnCount" minus 1. Values outside this range have no effect.

0: The referenced alarm text block is positioned on the left.

See also

MessageColumnIndex (Page 819)

MessageColumnCount (Page 818)

AlarmControl (Page 236)

MessageColumnSort (RT Professional)**Description**

Specifies how the content of an alarm text block referenced with "MessageColumnIndex" is sorted.

Access in runtime Read and write

Syntax

Object.**MessageColumnSort**[=SortMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

SortMode

Optional. A value or a constant that specifies sorting for the referenced alarm text block.

Value	Explanation
0	No sorting
1	Ascending sorting from the smallest to largest value
2	Descending sorting from the largest to smallest value

See also

AlarmControl (Page 236)

MessageColumnSortIndex (RT Professional)

Description

Defines the sorting order of the alarm text block referenced in "MessageColumnIndex". If you set the value to "0", the sorting criterion is removed in "MessageColumnSort".

Access in runtime: Read and write

Syntax

Object.**MessageColumnSortIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional Defines the sorting order of the alarm text block referenced in "MessageColumnIndex".

See also

AlarmControl (Page 236)

MessageColumnVisible (RT Professional)

Description

Specifies whether the alarm text block referenced with "MessageColumnIndex" is displayed.

Access in runtime Read and write

Syntax

Object.**MessageColumnVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the referenced alarm text block is displayed.

FALSE if the referenced alarm text block is not displayed.

See also

MessageColumnIndex (Page 819)

AlarmControl (Page 236)

MessageListType (RT Professional)

Description

Specifies the content to be displayed.

Access in runtime Read and write

Syntax

Object.**MessageListType**[=AlarmListType]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

AlarmListType

Optional. A value or a constant that specifies the content.

Value	Explanation
0	The alarms currently active are displayed upon screen refresh.
1	The "Historical alarm list" (short-term) display is activated upon screen refresh. The alarm view shows logged alarms and alarms currently pending. The display is updated immediately on activation of new messages.
2	The "Historical alarm list" (long-term) display is activated after a screen change. The alarm view only displays logged alarms, which the user can comment.
3	Only the currently locked alarms are displayed after a screen change.
4	The configured statistical calculations for logged alarms are displayed after a screen change.
5	Only previously hidden alarms are displayed after a screen change.

See also

AlarmControl (Page 236)

MiddleGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

MinimumNumberOfTimeAxes (RT Professional)**Description**

No access in runtime.

MinimumNumberOfTimeColumns (RT Professional)**Description**

No access in runtime.

MinimumNumberOfValueAxes (RT Professional)**Description**

No access in runtime.

MinimumNumberOfValueColumns (RT Professional)**Description**

No access in runtime.

MinimumValue (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the minimum value of the scale in the selected object.

Access in runtime: Read and write

Syntax

Object.**MinimumValue**[=DOUBLE | Int32 | SINGLE]

Object

Required. A "ScreenItem" object with the format:

- Bar
- Gauge
- Slider
- WindowSlider*

*: Read access only

DOUBLE | Int32 | SINGLE

Optional A value or a constant that specifies the minimum value. The data type depends on the format.

- DOUBLE: Bar
- Int32: Slider, WindowsSlider
- SINGLE: Gauge

MinNrOfCurves (RT Advanced)**Description**

No access in runtime.

MinPasswordValue (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

MinuteNeedleHeight (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the length of the minute hand.

Access in runtime Read and write

Syntax

Object.**MinuteNeedleHeight**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

Int32

Optional. A value or a constant that specifies the length of the minute hand.

Specify the length of the minute hand as a percentage relating to the radius of the clock face.

See also

Clock (Page 287)

MinuteNeedleWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the width of the minute hand.

Access in runtime Read and write

Syntax

Object.**MinuteNeedleWidth**[=Int32]

Object

Required. An object of the type ""ScreenItem" with the following format:

- Clock

Int32

Optional. A value or a constant that specifies the width of the minute hand.

Specify the width as a percentage relating to double the length of the minute hand.

See also

Clock (Page 287)

Mode (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the mode for the behavior of a text object in runtime.

Access in runtime

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**Mode**[=Type]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- IOField
- RoundButton*
- SymbolicIOField*

*: Read access only

You have no access in runtime with the following format:

- DateTimeField
- GraphicIOField
- Switch

Type

Optional. A value or a constant that specifies the mode for the behavior of a text object in runtime.

Value	VB constant	Description
0	hmilOFieldOutput	Output field
1	hmilOFieldInput	Input field
2	hmilOFieldInOut	Input and output field

MonitorNumber (RT Professional)**Description**

Returns the number of the monitor whose contents are shown in the screen window.
The monitor number refers to the number of monitors set in Microsoft Windows: "Control Panel > Display > Settings > Display"

Access in runtime: Read and write

Syntax

Object.**MonitorNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- Screenwindow

Int32

Optional. A value or a constant that specifies the number of the displayed monitor.

See also

ScreenWindow (Page 405)

Moveable (RT Professional)**Description**

Specifies whether the object can be moved in runtime.

Access in runtime Read and write

Syntax

Object.**Moveable**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the window can be moved in runtime.

FALSE if the window is fixed in runtime.

See also

AlarmControl (Page 236)

UserArchiveControl (Page 469)

TrendRulerControl (Page 444)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

MsgFilterSQL (RT Professional)**Description**

Specifies one or several SQL statements for user-defined selection of the alarms. Several customized selections are connected with "OR". If you have configured a fixed selection "DefaultMsgFilterSQL", the SQL statements of "DefaultMsgFilterSQL" and "MsgFilterSQL" are connected with "AND".

Access in runtime: Read and write

Syntax

Object.**MsgFilterSQL**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or constant that specifies the SQL statements for user-defined selection of messages.

See also

AlarmControl (Page 236)

Name (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Returns the object name as STRING. The returned value is dependent upon the used object.

Access in runtime: Read and write

Syntax

Object.**Name**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- Clock
- DiscSpaceView *
- Gauge
- Slider

* Read access only

You have no access in runtime with the following format:

- AlarmControl
- AlarmView
- ApplicationWindow
- BatteryView
- Bar
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox

- Connector
- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- ProDiagOverview
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- S7GraphOverview
- ScreenWindow
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl

- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowSlider
- ZoneLabelView
- ZoneQualityView

String

Optional. A value or a constant that returns the object name.

Comments

Dependent on the specified object, the following object names will be returned:

- Tag: Name of the tag without server or tag prefix.
- Project: Name of the current runtime project.
- Dataltem: Name of the Dataltem object.
- Layer: Name of the layer.
- FunctionTrendControl: Name of the trend referenced by the "Index" property.

Note

Use the "Name" property to address a tag in the Tags" list. Tag names are structured in WinCC according to the following scheme:

<Tag prefix><Name of tag>

If you only specify the tag name, the tag prefix is applied from the screen window shortcut.

Example

The following example returns the name of the current Runtime project as Trace:

Table 1-129

```
'VBS160
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
```

NameColumnWidth (RT Advanced)**Description**

No access in runtime.

NavigateTo (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the module to be jumped to.

Access in runtime: Read and write

Syntax

Object.**NavigateTo**[=String]

Object

Required. An object of the type "ScreenItem" with the following format:

- PLCCodeViewer
- SysDiagControl *

* RT Advanced No access, RT Professional read access only

String

Optional. A value or a constant that specifies the module.

NavigationButtons (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

NavigationPath_Font (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

NavigationPath_RootText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

NavigationPath_TextColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

NavigationpathDiagbufferDetailText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

NavigationpathDiagbufferText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

NeedleBorderColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the line color of the pointer.

Access in runtime Read and write

Syntax

Object.**NeedleBorderColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

Color

Optional. A value or a constant that specifies the line color of the pointer.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Clock (Page 287)

NeedleColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the pointer color.

Also specify with ""NeedleFillStyle" that the pointer color is to be displayed.

Access in runtime Read and write

Syntax

Object.**NeedleColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

Color

Optional. A value or a constant that specifies the pointer color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Clock (Page 287)

NeedleFillStyle (RT Professional)

Description

Specifies whether the pointers are displayed as filled or transparent.

Access in runtime Read and write

Syntax

Object.**NeedleFillStyle**[=THmiFillStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

THmiFillStyle

Optional. A value or a constant that specifies whether the pointers are filled or transparent.

Value	VB constant	Explanation
0	<code>hmiFillStyleSolid</code>	The pointers appear transparent and are indicated by a margin in the foreground color.
65536	<code>hmiFillStyleTransparent</code>	The pointers are filled in the pointer fill color and indicated by a margin in the foreground color.

See also

Clock (Page 287)

NeedleHeight (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

NoHitTest (RT Professional)

Description

No access in runtime.

NormalColor (RT Professional)

Description

Specifies the color of the utilized memory in the memory space view when the memory space utilization is in the non-critical range.

Access in runtime: Read

Syntax

Object.**NormalColor**[=Color]

Object

Required. A "ScreenItem" object with the following characteristics:

- DiscSpaceView

Color

Optional. A value or a constant that specifies the color of the memory display in the normal range.

Comments

You use the "RGB" function to define the color in RGB format (red, green, blue). To do this, you specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255). The "Red" color is displayed as follows, for example:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

DiskSpaceView (Page 298)

NormalRangeColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the normal range on the scale of the "Gauge" object.

The "NormalRangeVisible" property must have the value TRUE for the normal range to be displayed.

Access in runtime Read and write

Syntax

Object.**NormalRangeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the color of the normal range.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

NormalRangeVisible (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the normal range in the scale of the "Gauge" object is displayed.

Access in runtime: Read and write

Syntax

Object.**NormalRangeVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the normal range will be displayed in the scale.

Comments

Specify the color of the normal range with the "NormalRangeColor" property.

See also

Gauge (Page 320)

NumberOfButtons (RT Professional)**Description**

No access in runtime.

NumberOfLines (RT Advanced)**Description**

No access in runtime.

NumberOfVisibleLines (RT Professional)**Description**

Specifies the number of visible lines in the PLC code display.

Access in runtime: Read and write

Syntax

Object.**NumberOfVisibleLines**[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- **PLCCodeViewer**

Int32

Optional. A value or a constant that specifies the number of lines in the PLC code display.

See also

PLCCodeViewer (Page 377)

NumberStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

1.5.5.10 Properties O-P (Panels, Comfort Panels, RT Advanced, RT Professional)

Object (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ObjectSizeDeclutteringEnable (RT Professional)

Description

Specifies whether only the objects within a set size range are displayed.

Access during runtime: Read

Syntax

Object.ObjectSizeDeclutteringEnable

Object

Necessary. A "Screen" object.

Example

The example outputs the Decluttering properties of the screen "NewPDL1" as Trace.

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 215)

ObjectSizeDeclutteringMax (RT Professional)

Description

Returns the upper size range for the display suppression of objects of the specified screen as LONG.

Access during runtime: Read

Syntax

Object.**ObjectSizeDeclutteringMax**

Object

Necessary. A "Screen" object.

Example

The example outputs the Decluttering properties of the screen "NewPDL1" as Trace.

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 215)

ObjectSizeDeclutteringMin (RT Professional)

Description

Returns the lower size range for the display suppression of objects of the specified screen as LONG.

Access during runtime: Read

Syntax

Object.**ObjectSizeDeclutteringMin**

Object

Necessary. A "Screen" object.

Example

The example outputs the Decluttering properties of the screen "NewPDL1" as Trace.

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

See also

Screen (Page 215)

OcxGuid (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

OCXState (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

OcxStateForEs2Rt (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Online (RT Professional)

Description

Specifies start and stop of the updating.

Access in runtime: Read and write

Syntax

Object.**Online**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the updated display is to be stopped. The values are buffered and updated when the button is clicked again.

FALSE if the updated display is to be continued.

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnValue (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

OperationSteps (RT Professional)**Description**

Specifies by how many steps the slider of the scrollbar is moved with one mouseclick.

Access in runtime: Read and write

Syntax

Object.**OperationSteps**[=Int32]

Object

Required. An object of the type ""ScreenItem" with the format:

- WindowSlider

Int32

Optional. A value or a constant which specifies by how many steps the slider of the scrollbar is moved with one mouse-click.

See also

WindowSlider (Page 485)

OperatorAlarms (RT Professional)**Description**

No access in runtime.

OperatorMessageId (RT Professional)**Description**

Specifies the assignment of ID number and trigger event in the alarm view.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageId**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or constant that specifies the assignment of ID number and trigger event in the alarm view.

Value	Description	Explanation
0	Lock	Trigger event "Lock"
1	Unlock	"Unlock" Trigger Event
2	Hide	Trigger event "Hide"
3	Unhide	Trigger event "Unhide"
4	Ackn	Trigger event "Acknowledge"

See also

AlarmControl (Page 236)

OperatorMessageIndex (RT Professional)

Description

References an operator message event. You can assign the values of other properties to a specific operator message by using the property.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. A value or a constant that specifies the alarm event for an operator input alarm.

Value	Explanation
0	Message event "Lock"
1	"Unlock" Message Event
2	Message event "Hide"
3	Message event "Unhide"
4	Message event "Acknowledge"

See also

AlarmControl (Page 236)

OperatorMessageName (RT Professional)

Description

Specifies the name which is referenced with the "OperatorMessageIndex" event in message events for operator messages.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the name that is referenced with the "OperatorMessageIndex" event in alarm events for operator input alarms.

Value	Explanation
Lock	Message event "Lock"
Unlock	Message event "Enable"
Hide	Message event "Hide"
Unhide	Message event "Unhide"
Quit	Message event "Ackn."

See also

AlarmControl (Page 236)

OperatorMessageNumber (RT Professional)**Description**

Specifies a message number for the operator message of the selected message event if you do *not* use the operator message of WinCC.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageNumber**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

Int32

Optional. Specifies a message number for the operator message of the selected message event.

See also

AlarmControl (Page 236)

OperatorMessageSelected (RT Professional)**Description**

Activates the message events in the list that trigger operator messages.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSelected**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the alarm events in the list that trigger operator input alarms are activated.

See also

AlarmControl (Page 236)

OperatorMessageSource1 (RT Professional)

Description

Specifies a alarm text block of the operated message to be added to "Process value block 1" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource1**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 1" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "1" at process value as the alarm text block lock of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource2 (RT Professional)

Description

Specifies a alarm text block of the operated message to be added to "Process value block 2" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource2**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 2" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "2" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource3 (RT Professional)

Description

Specifies a alarm text block of the operated message to be added to "Process value block 3" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource3**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 3" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "3" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource4 (RT Professional)**Description**

Specifies a alarm text block of the operated message to be added to "Process value block 4" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.OperatorMessageSource4[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 4" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "4" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource5 (RT Professional)

Description

Specifies a alarm text block of the operated message to be added to "Process value block 5" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource5**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 5" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "5" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource6 (RT Professional)

Description

Specifies a alarm text block of the operated message to be added to "Process value block 6" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource6**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 6" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "6" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource7 (RT Professional)**Description**

Specifies a alarm text block of the operated message to be added to "Process value block 7" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.OperatorMessageSource7[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 7" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "7" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource8 (RT Professional)

Description

Specifies a alarm text block of the operated message to be added to "Process value block 8" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource8**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 8" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "8" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource9 (RT Professional)

Description

Specifies a alarm text block of the operated message to be added to "Process value block 9" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.**OperatorMessageSource9**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 9" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "9" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSource10 (RT Professional)**Description**

Specifies a alarm text block of the operated message to be added to "Process value block 10" of the operator message configured here.

Access in runtime: Read and write

Syntax

Object.OperatorMessageSource10[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. Specifies a alarm text block of the operated message to be added to "Process value block 10" of the operator message configured here.

Example

An operator message is to be generated to indicate that a message was locked. The content of "User text block 1" of the locked message, e.g. "Motor faulty" should be displayed in "Process value block 1" of the operator message. Select "10" at process value as the alarm text block of the operated alarm "User text block 1".

See also

AlarmControl (Page 236)

OperatorMessageSourceType1 (RT Professional)**Description**

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType1**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType2 (RT Professional)**Description**

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType2**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType3 (RT Professional)**Description**

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as value.

Access in runtime: Read and write

Syntax

Object.OperatorMessageSourceType3[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

OperatorMessageSourceType4 (RT Professional)

Description

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType4**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType5 (RT Professional)

Description

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType5**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType6 (RT Professional)**Description**

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType6**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType7 (RT Professional)

Description

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType7**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType8 (RT Professional)

Description

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType8**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType9 (RT Professional)**Description**

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType9**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OperatorMessageSourceType10 (RT Professional)**Description**

Specifies whether the content of the alarm text block in the operator input alarm is to be transferred as text or as a value.

Access in runtime Read and write

Syntax

Object.**OperatorMessageSourceType10**[=TransferAs]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

TransferAs

Optional. A value or a constant that specifies how the alarm text block is displayed in the operator input alarm.

The following settings are available:

Value	Designation	Description
0	Text	The alarm text block is shown as text.
1	Value	The alarm text block is shown as a value.

See also

AlarmControl (Page 236)

OutputAddressText (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

PaddingBottom (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

PaddingLeft (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

PaddingRight (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

PaddingTop (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

PageMode (RT Professional)**Description**

Enables paging is in the long-term archive list. Allows you to display all messages of the short-term archive in the long-term archive list. Use the "PageModeMessageNumber" property to determine the number of messages displayed per page. The page up/down buttons of the toolbar can be used if paging is enabled.

Access in runtime: Read and write

Syntax

Object.**PageMode**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if scrolling in the historical alarm list (long-term) is possible.

FALSE if scrolling in the historical alarm list (long-term) is not possible.

See also

AlarmControl (Page 236)

PageModeMessageNumber (RT Professional)

Description

Defines the number of messages shown per page when paging the long-term archive list.

Access in runtime: Read and write

Syntax

Object.**PageModeMessageNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl

Int32

Optional A value or constant that specifies the number of messages per page.

See also

AlarmControl (Page 236)

Password (RT Advanced)

Description

Determines the password for the loading of the remote monitor.

Access in runtime: Read and write

Syntax

Object.**Password**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- SmartClientView

STRING

Optional A value or constant that contains the password for establishing remote monitoring.

See also

SmartClientView (Page 413)

PasswordsMustBeEncrypted (RT Advanced)**Description**

No access in runtime.

Path (RT Professional)**Description**

Returns the path of the current project without the file names as STRING. For a WinCC client without its own path, the path is returned in UNC format, otherwise the local path is returned.

Access during runtime: Read

Syntax

Object.**Path**

Object

Necessary. A "Project" object.

Example

The following example returns the project path as Trace:

```
'VBS161
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbNewLine
```

See also

Project (Page 214)

PathHeaderBackColor (RT Professional)**Description**

No access in runtime.

PathHeaderFont (RT Professional)**Description**

Specifies the header font.

Access in runtime Read and write

Syntax

Object.**PathHeaderFont**[=Font]

Object

Required. An object of the type "ScreenItem" with the following format:

- PLCCodeViewer
- S7GraphOverview

Font

Optional. A value or a constant that specifies the font of the header.

See also

PLCCodeViewer (Page 377)

S7GraphOverview (Page 403)

PathHeaderTextColor (RT Professional)

Description

Specifies the font color in the header of the GRAPH overview.

Access in runtime Read and write

Syntax

Object.**PathHeaderTextColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- S7GraphOverview

Color

Optional. A value or a constant that specifies the font color in the header of the S7-GRAPH overview.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

S7GraphOverview (Page 403)

PercentageAxis (RT Professional)

Description

Specifies that an additional axis with percentage scaling is shown in the trend view.

Access in runtime: Read and write

Syntax

Object.**PercentageAxis**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE, if an axis with percentage scaling is shown.

FALSE, if only value and time axes are shown.

See also

OnlineTrendControl (Page 360)

PercentageAxisAlignment (RT Professional)

Description

Specifies the alignment of the percentage axis.

Access in runtime: Read and write

Syntax

Object.**PercentageAxisAlignment**[=AxisAlignment]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

AxisAlignment

Optional. Value or constant that specifies the alignment of the percentage axis.

Value	Designation	Description
0	Left	The axis with percentage scaling is left-aligned.
1	Right	The axis with percentage scaling is right-aligned

See also

OnlineTrendControl (Page 360)

PercentageAxisColor (RT Professional)**Description**

Specifies the font color and line color of the percentage axis.

Access in runtime: Read and write

Syntax

Object.**PercentageAxisColor**[=Color]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the percentage axis.

Comments

You use the "RGB" function to define the color in RGB format (red, green, blue). To do this, you specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255). The "Red" color is displayed as follows, for example:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTrendControl (Page 360)

Picture (RT Professional)

Description

Specifies the screen from the WinCC project graphics to be displayed in the object in runtime.

Access in runtime: Read and write

Syntax

Object.**Picture**[=HmiObjectHandle]

Object

Required. An object of the type "ScreenItem" with the format:

- Clock
- GraphicView

HmiObjectHandle

Optional. A value or a constant that specifies the screen from the WinCC project graphics to be displayed in the object in runtime.

PictureAlignment (RT Professional)

Description

Specifies the display type for the background screen in the process image.

Access in runtime: Read and write

Syntax

Object.**PictureAlignment**[=PictureAlignment]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton*

*: Read access only

PictureAlignment

Optional A value or a constant that specifies the display type for the background screen in the process image.

PictureAreaBottomMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PictureAreaLeftMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PictureAreaRightMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PictureAreaTopMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PictureAutoSizing (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PictureDeactivated (RT Professional)

Description

Specifies the graphic from the WinCC project graphics that is displayed in the "Disabled" status.
Access in runtime Read

Syntax

Object.**PictureDeactivated**[=HmiObjectHandle]

Object

Required. An object of the type ""ScreenItem" with the following format:

- Roundbutton

HmiObjectHandle

Optional. A value or a constant that specifies the picture that will be displayed in the "Deactivated" status.

See also

RoundButton (Page 399)

PictureList (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

PictureOff (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the picture that is displayed in the "Off" status.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**PictureOff**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the format:

- Button
- GraphicIOField
- RoundButton*

*: Read access only

You have no access in runtime with the following format:

- Switch

HmiObjectHandle

Optional A value or a constant that specifies the picture that will be displayed in the "Off" status.

Comments

The picture (*.BMP or *.DIB) must be in the "GraCS" folder of the current project so that you can integrate it.

PictureOn (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the screen that will be displayed in the "on" state.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**PictureOn**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the format:

- Button
- GraphicIOField
- RoundButton*

*: Read access only

You have no access in runtime with the following format:

- Switch

HmiObjectHandle

Optional A value or a constant that specifies the screen that will be displayed in the "on" state.

Comments

The screen (*.BMP or *.DIB) must be in the "GraCS" folder of the current project in order to be able to integrate it.

PictureRotation (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PictureSizeMode (RT Professional)

Description

Specifies the size adaptation between Media Player and the contents to be displayed.

Access in runtime: Read and write

Syntax

Object.**PictureSizeMode**[=PictureSizeMode]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

PictureSizeMode

Optional. Value or constant that specifies the size adaptation between Media Player and the contents to be displayed.

Value	Designation	Description
0	Fit object size to content	Specifies that the size of the Media Player is adapted to the size of the contents to be displayed.
1	Fit content to object size	Specifies that the contents to be displayed are adapted to the size of the Media Player.

See also

MediaPlayer (Page 343)

PlayCount (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PlayEndless (RT Professional)

Description

Specifies that the media file is repeated in an endless loop.

Access in runtime: Read and write

Syntax

Object.**PlayEndless**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the media file is repeated in an endless loop.

FALSE, if the media file is played only once.

See also

MediaPlayer (Page 343)

PLCFilter (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PlcUDTFilter (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PointerColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the pointer color of the "Gauge" object.

Access in runtime Read and write

Syntax

Object.**PointerColor**[=Color]

Object

Required. An object of the type ""ScreenItem" with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the pointer color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

Points (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

PointsCount (RT Professional)

Description

Specifies the number of corner points of the polyline or of the polygon.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**PointsCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Polygon
- Polyline
- Tubepolyline

You have no access in runtime with the following format:

- Connector
- Line

Int32

Optional. A value or a constant that specifies the number of corner points of the polyline.

PopupMenuEnabled (RT Professional)

Description

No access in runtime.

PositionFont (RT Professional)

Description

Sets the font for the label of the slider.

Access in runtime: Read and write

Syntax

Object.**PositionFont**[=Font]

Object

Required. An object of the type "ScreenItem" with the format:

- Slider

Font

Optional. Value or constant that specifies the font.

Precision (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the number of decimal places (0 to 20).

Access in runtime: Read and write

Syntax

Object.**Precision**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional A value or a constant that specifies the number of decimal places (0 to 20).

See also

Bar (Page 258)

PreferredUseOnAck (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

Pressed (RT Professional)**Description**

Specifies whether the selected object is displayed in a pressed state.

Access in runtime: Read and write

Syntax

Object.**Pressed**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton*

*: Read access only

BOOLEAN

Optional. TRUE, if the selected object is displayed in a pressed state.

PrintJob (RT Professional)**Description**

Specifies a print job that is created in the "Reports" editor.

Access in runtime: Read and write

Syntax

Object.**PrintJob**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

HmiObjectHandle

Optional. A value or a constant that specifies the layout of the printout.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ProcessTag (RT Professional)

Description

No access in runtime.

ProcessValue (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the default for the value to be displayed.

This value will be used in runtime if the associated tag is not connected or has not been updated when the screen starts.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ProcessValue**[=DOUBLE | Int32 | Object | SINGLE]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- CheckBox
- Gauge
- GraphicIOField
- IOField
- OptionGroup
- Slider
- SymbolicIOField*
- WindowsSlider*

*: Read access only

You have no access in runtime with the following format:

- Button
- DateTimeField
- Switch
- SymbolLibrary

DOUBLE | Int32 | SINGLE

Optional. A value or a constant that includes the default value. The data type depends on the format.

- DOUBLE: Bar
- Int32: CheckBox, GraphicIOField, OptionGroup, Slider, SymbolicIOField, WindowsSlider
- Object: IOField
- SINGLE: Gauge

Comments

If you want to assign the "ProcessValue" SmartTags property, then you must formulate the assignment as follows:

'Examples for the assignment of SmartTags

```
'Example 1
IOField.ProcessValue = SmartTags("TagName").Value
'Example 2
HmiRuntime.Screens("Screen_1").ScreenItems("IOField_1").ProcessValue =
SmartTags("Tag_1").Value
```

ProgID (RT Professional)

Description

In the case of non-WinCC controls, the version-independent ProgID is returned as the type.

ProhibitDataRecordTagInOnlySimpleView (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

1.5.5.11 Properties Q-R (Panels, Comfort Panels, RT Advanced, RT Professional)

QualityCode (RT Professional)

Description

Returns the quality of a tag value after reading the tag as SHORT. After a tag has been written, the value is invalid.

Access during runtime: Read

Syntax

Object.**QualityCode**

Object

Necessary. An object of the "HMIRuntime" type.

Example

The following example indicates the quality of the read value when no errors have occurred during the reading process:

```
'VBS83
Dim objTag
Dim lngLastErr
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
lngLastErr = objTag.LastError
If 0 = lngLastErr Then
MsgBox objTag.QualityCode
End If
```

See also

Tag (Page 228)

Radius (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the radius.

Access in runtime: Read and write

Syntax

Object.**Radius**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Circle
- CircleSegment
- CircularArc
- RoundButton

Int32

Optional. A value or constant which determines the radius in pixels.

RadiusHeight (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the secondary axis.

Access in runtime: Read and write

Syntax

Object.**RadiusHeight**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Ellipse
- EllipseSegment
- EllipticalArc
- TubeArcObject

You have no access in runtime with the following format:

- Circle

Int32

Optional. A value or constant which determines the secondary axis in pixels.

See also

[Ellipse \(Page 300\)](#)

[EllipseSegment \(Page 303\)](#)

[EllipticalArc \(Page 306\)](#)

[TubeArcObject \(Page 460\)](#)

[Circle \(Page 279\)](#)

RadiusWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the main axis.

Access in runtime: Read and write

Syntax

Object.**RadiusWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Ellipse
- EllipseSegment
- EllipticalArc
- TubeArcObject

You have no access in runtime with the following format:

- Circle

Int32

Optional. A value or constant which determines the main axis in pixels.

See also

[Ellipse \(Page 300\)](#)

[EllipseSegment \(Page 303\)](#)

EllipticalArc (Page 306)

TubeArcObject (Page 460)

Circle (Page 279)

Recipe (RT Advanced)

Description

No access in runtime.

RecipeName (RT Advanced)

Description

Returns the name of the recipe that is currently being displayed in the "Recipe view".

Access in runtime: Read and write

Syntax

Object.**RecipeName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

STRING

Optional A value or a constant that returns the name of the recipe.

See also

RecipeView (Page 390)

RecipeNameCaption (RT Advanced)

Description

No access in runtime.

RecipeNrCaption (RT Advanced)

Description

No access in runtime.

RecipeNrColFirst (RT Advanced)

Description

No access in runtime.

RecipeNumber (RT Advanced)

Description

Returns the number of the recipe that is currently being displayed in the "Recipe view".

Access in runtime: Read and write

Syntax

Object.**RecipeNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

Int32

Optional A value or a constant that returns the number of the recipe.

See also

RecipeView (Page 390)

RecordName (RT Advanced)

Description

Returns the name of the recipe data record that is currently being displayed in the "Recipe view".

Access in runtime: Read and write

Syntax

Object.**RecordName**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

STRING

Optional A value or a constant that returns the name of the recipe data record.

See also

RecipeView (Page 390)

RecordNrColFirst (RT Advanced)**Description**

No access in runtime.

RecordNumber (RT Advanced)**Description**

Returns the number of the recipe data record that is currently being displayed in the "Recipe view".

Access in runtime: Read

Syntax

Object.**RecordNumber**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- RecipeView

Int32

Optional A value or a constant that returns the number of the recipe data record.

See also

RecipeView (Page 390)

RelativeFillLevel (RT Professional)**Description**

Specifies the fill percentage for the object.

Access in runtime: Read and write

Syntax

Object.**RelativeFillLevel**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- Circle
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon
- Rectangle
- RoundButton*
- TextField
- WindowsSlider*

*: Read access only

You have no access in runtime with the following format:

- CircleSegment

Int32

Optional. A value or a constant that specifies the fill percentage of the object.

RenameButtonVisible (RT Advanced)

Description

No access in runtime.

ReSizeable (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

RightMargin (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

Rotation (RT Professional)**Description**

Specifies the angle of rotation in degrees. The angle of rotation is measured counterclockwise.

Access in runtime: Read and write

Syntax

Object.**Rotation**[=SymbolLibraryRotation]

Object

Required. An object of the type "ScreenItem" with the format:

- SymbolLibrary

SymbolLibraryRotation

Optional. A value or a constant that specifies the angle of rotation in degrees.

Value	VB constant	Explanation
0	hmiSymbolLibraryRotationNone	The object rotates by 0 degrees.
90	hmiSymbolLibraryRotation90Degree	The object rotates by 90 degrees.
180	hmiSymbolLibraryRotation180Degree	The object rotates by 180 degrees.
270	hmiSymbolLibraryRotation270Degree	The object rotates by 270 degrees.

RotationAngle (RT Professional)**Description**

Specifies the angle of rotation in degrees.

Access in runtime: Read and write

Syntax

Object.**RotationAngle**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Line
- Polygon
- Polyline
- TextField
- TubeTeeObject

Int32

Optional A value or a constant that specifies the angle of rotation in degrees.

Comments

In Runtime the object rotates clockwise on the reference point.

See also

Line (Page 337)

Polygon (Page 379)

Polyline (Page 382)

TextField (Page 440)

TubeTeeObject (Page 467)

RotationCenterLeft (RT Professional)**Description**

Specifies the X coordinate of the pivot point for the object in runtime.

Access in runtime: Read and write

Syntax

Object.**RotationCenterLeft**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Line
- Polygon
- Polyline
- TextField*

*: Read access only

Int32

Optional. A value or a constant that specifies the X coordinates of the pivot point to rotate the object in runtime.

Comments

The value of the X coordinate is relative to the object width. Specify the value as a percentage from the left edge of the rectangle that surrounds the object.

RotationCenterTop (RT Professional)**Description**

Specifies the Y coordinate of the pivot point for the object in runtime.

Access in runtime: Read and write

Syntax

Object.**RotationCenterTop**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Line
- Polygon
- Polyline
- TextField*

*: Read access only

Int32

Optional. A value or a constant that specifies the Y coordinates of the pivot point to rotate the object in Runtime.

Comments

The value of the Y coordinate is relative to the object width. Specify the value as a percentage from the upper edge of the rectangle that surrounds the object.

RoundCornerHeight (RT Professional)**Description**

Specifies the corner radius. Enter the value as a percentage of half the height of the object.

Access in runtime: Read and write

Syntax

Object.**RoundCornerHeight**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Rectangle

Int32

Optional A value or a constant that specifies the corner radius.

RoundCornerWidth (RT Professional)

Description

Specifies the corner radius. Enter the value as a percentage of half the width of the object.

Access in runtime: Read and write

Syntax

Object.**RoundCornerWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Rectangle

Int32

Optional A value or a constant that specifies the corner radius.

RowScrollbar (RT Professional)

Description

Specifies when the vertical scroll bar is displayed.

Access in runtime Read and write

Syntax

Object.**RowScrollbar**[=ScrollbarVisibility]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

ScrollbarVisibility

Optional. A value or a constant that specifies when the vertical scroll bar is displayed.

Value	Explanation
0	The vertical scroll bar is not displayed.
1	The vertical scroll bar is displayed if the display area is too small for all rows.
2	The vertical scroll bar is always displayed.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

RowTitleAlignment (RT Professional)

Description

Specifies the type of row header alignment.

Access in runtime Read and write

Syntax

Object.**RowTitleAlignment**[=HorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

HorizontalAlignment

Optional. A value or a constant that specifies the row header alignment.

Value	Description	Explanation
0	Left	The row headers are aligned left.
1	Centered	The row headers are aligned to center.
2	Right	The row headers are aligned right.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

RowTitles (RT Professional)

Description

Specifies whether numbered column headers are displayed.
Access in runtime Read and write

Syntax

Object.**RowTitles**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional. TRUE if numbered column headers are displayed.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

RTPersistence (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether online configurations are retained after a screen change.
Also set "RTPersistenceType" to 1 or 2.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**RTPersistence**[=RTPersistence]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- SysDiagControl
- UserView

RTPersistence

Optional. A value or a constant that specifies whether online configurations are retained after a screen change.

Value	Explanation
0	The current online configurations are discarded at the next picture change.
1	The current online configurations are retained at the next picture change.
2	All online configurations made are lost. The screen is set to the content originally configured.

RTPersistenceAuthorization (RT Professional)

Description

Specifies the authorization required for online configuration in runtime.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**RTPersistenceAuthorization**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

You have no access in runtime in the case of the following characteristic:

- SysDiagControl
- UserView

HmiObjectHandle

Optional. A value or a constant that specifies the user rights in runtime.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SysDiagControl (Page 431)

UserView (Page 481)

RTPersistenceType (RT Professional)**Description**

Specifies how online configurations of WinCC are retained.

Access in Runtime:

- RT Advanced: No access
- RT Professional: Read

Syntax

Object.**RTPersistenceType**

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- SysDiagControl
- UserView

RTPersistenceType

Optional. A value or a constant that specifies how online configurations are retained by WinCC.

Value	Explanation
0	Online configurations are discarded. These are lost at the next picture change.
1	Online configurations are retained during Runtime. These are lost on exiting.
2	Online configurations are retained permanently. These are also available after restart.

RulerColor (RT Advanced)**Description**

Specifies the color of the scale gradation (help line) of the axis label.

Access in runtime Read and write

Syntax

Object.**RulerColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendView

Color

Optional. A value or a constant that specifies the color of the scale gradation.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

TrendView (Page 454)

RulerColumns (RT Professional)**Description**

No access in runtime.

RulerType (RT Professional)**Description**

Specifies the display mode of the value table.

Access in runtime: Read and write

Syntax

Object.**RulerType**[=TrendRulerControlType]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

TrendRulerControlType

Optional. A value or a constant that specifies the display mode of the value table.

Value	Designation	Explanation
0	Ruler	Specifies that a ruler window is displayed. The ruler window shows the coordinate values of the trends on the ruler or values of a selected line in the table.
1	Extreme values	Specifies that the statistics area window is displayed. The statistics area window shows the values at the low limit and high limit of the trends between two rulers or of the selected area in the table.
2	Statistics	Specifies that a statistics window is displayed. The statistics window shows the statistical evaluation of the trend between two rulers or of the selected values in the table.

See also

TrendRulerControl (Page 444)

1.5.5.12 Properties S (Panels, Comfort Panels, RT Advanced, RT Professional)**S7Device (RT Advanced)****Description**

No access in runtime.

ScaleColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color of the scale.

Access in runtime: Read and write

Syntax

Object.**ScaleColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- TrendView

Color

Optional. A value or a constant that specifies the color of the scale.

Comments

ScreenItem object with the format "Bar": For the color to be displayed, the property ""ShowScale" must have the value TRUE.

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

ScaleDenominator (Panels, Comfort Panels, RT Advanced)

Description

Defines the counter for scaling on the client.

Syntax

Object.ScaleDenominator=[Int]

Object

Required. A "ScreenItem" object with the format "SmartClientView".

Int

Optional. A value or a constant that specifies the value.

See also

SmartClientView (Page 413)

ScaleDenominator (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ScaleGradation (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the distance between two large marking lengths of the scale.

Access in runtime: Read and write

Syntax

Object.ScaleGradation[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the distance between two large marking lengths of the scale.

See also

Bar (Page 258)

ScaleLabelColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color of the label for the scale marks of the "Gauge" object.

Access in runtime Read and write

Syntax

Object.**ScaleLabelColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the label color of the scale gradation.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

ScaleLabelFieldLength (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ScaleLabelFont (RT Professional)

Description

Specifies the font for the scale label.

Access in runtime: Read and write

Syntax

Object.**ScaleLabelFont**[=Font]

Object

Required. An object of the type "ScreenItem" with the format:

- Gauge

Font

Optional. A value or a constant that specifies the font for scale labeling.

ScaleLabelingDoubleLined (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ScaleNumerator (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ScalePosition (RT Professional)

Description

Specifies the position of the scale.

Also set "ShowScale" to "TRUE".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ScalePosition**[=ScalePosition]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

You have no access in runtime with the following format:

- Slider

ScalePosition

Optional. A value or a constant that specifies the position of the scale.

Value	VB constant	Explanation
0	hmiScalePositionLeftUp	For a vertical bar, the scale is displayed at the top. For a horizontal bar the scale will be shown at the left.
1	hmiScalePositionRightDown	For a vertical bar, the scale is displayed at the bottom. For a horizontal bar the scale will be shown at the right.

ScaleStart (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ScaleTickColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color of the scale gradation of the "Gauge" object.

Access in runtime: Read and write

Syntax

Object.**ScaleTickColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Gauge

Color

Optional. A value or a constant that specifies the color of the scale gradation.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

ScaleTickLabelPosition (RT Professional)

Description

Specifies the diameter of the assumed circle on which the label of the scale divisions is located.

Access in runtime: Read and write

Syntax

Object.**ScaleTickLabelPosition**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the diameter of the assumed circle on which the label of the scale division is located.

Value range from 0 to 1

0: The label is located in the center of the scale.

1: The diameter of the assumed circle for the label is the smaller value of the geometry properties "Width" and "Height". A part of the label can lie outside the object limits and is therefore invisible.

See also

Gauge (Page 320)

ScaleTickLength (RT Professional)

Description

Specifies the length of the main markings for the scale division. The value refers to half the smaller value of the geometry properties "Width" and "Height".

The length of the marking lengths for the fine divisions is 0.5* scale width.

Access in runtime: Read and write

Syntax

Object.**ScaleTickLength**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the length of the main markings of the scale.

Value range 0 to scale end value

0: There are no scale divisions. The divisions of the scale in the areas are not visible.

Scale end value: The scale division ranges from the mid-point of the scale disc to the value specified by the scale end value.

See also

Gauge (Page 320)

ScaleTickPosition (RT Professional)**Description**

Specifies the diameter of the assumed circle on which the scale divisions are located.

The main markings of the scale divisions lie with their outward-facing ends on this circle.

Access in runtime: Read and write

Syntax

Object.**ScaleTickPosition**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

DOUBLE

Optional A value or a constant that specifies the diameter of the assumed circle on which the scale divisions are located.

Value range from 0 to 1

0: The scale divisions are located in the center of the scale.

1: The diameter of the assumed circle for the label is the scale divisions is the smaller value of the geometry properties ""Width" and "Height".

See also

Gauge (Page 320)

Scaling (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ScalingType (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the type of bar scaling.

Also set "ShowScale" to "TRUE".

Access in runtime Read and write

Syntax

Object.**ScalingType**[=BarScalingType]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BarScalingType

Optional. A value or a constant that specifies the type of bar scaling.

Value	VB constant	Description	Explanation
0	hmiBarScalingLinear	Linear	The large marks are distributed evenly over the scale. The distance between the large marks corresponds to the value of "LargeTicksSize".
1	hmiBarScalingLogarithmic	Logarithmic	The distribution of the large marks on the scale follows a logarithmic function. The representation of low values is very strongly highlighted.
2	hmiBarScalingNegativeLogarithmic	Negative logarithmic	The distribution of the large marks on the scale follows a negative logarithmic function. The representation of high values is very strongly highlighted.
3	hmiBarScalingAutomatic	Automatic	The large marks are distributed evenly over the scale. The distance between the large marks is specified automatically.
4	hmiBarScalingTangent	Tangents	The distribution of the large marks on the scale highlights the representation of the low and high values.
5	hmiBarScalingQuadratic	Square	The distribution of the large marks follows a square function. The representation of high values is highlighted.
6	hmiBarScalingCubic	Cubic	The distribution of the large marks on the scale follows a cubic function. This highlights the representation of large values.

See also

Bar (Page 258)

ScreenItems (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Returns the ScreenItems list.

Access in Runtime: Read

Syntax

Object.ScreenItems

Object

Required A "ScreenItems"." object.

See also

ScreenItem (Page 218)

ScreenItems (list) (Page 220)

ScreenName (RT Professional)**Description**

Specifies the screen to be displayed in the screen window in runtime.

Access in runtime: Read and write

Syntax

Object.**ScreenName**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following format:

- ScreenWindow

HmiObjectHandle

Optional A value or a constant that specifies the screen to be displayed in the screen window in runtime.

See also

ScreenWindow (Page 405)

Screens (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns the Screens listing. The Screens list contains two elements: The first element with the index 0 represents the permanent area. The second element with the index 1 represents the root screen. Alternatively, the two elements can be addressed by means of their names. Use "Overview" for the permanent area and "Base" for the root screen.

Note

The alarm window and the alarm indicator are not contained in the Screens list, even if they have the focus in Runtime.

Access in runtime: Read

Syntax

Object.Screens

Object

Required. An object of the type "Screens".

See also

Screen object (list) (Page 222)

ScreenScaleMode (RT Professional)

Description

Specifies the scaling mode of the displayed screen.

Access in Runtime: Read and write

Syntax

Object.ScreenScaleMode[=ScreenScaleModeType]

Object

Required. An object of the type "ScreenItem" with the format:

- ScreenWindow

ScreenScaleModeType

Optional. A value or a constant that specifies the scaling mode of the displayed screen.

ScrollBarOrientation (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

SecondGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

SecondGradientOffset (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

SecondNeedleHeight (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the length of the second hand.

Access in runtime Read and write

Syntax

Object.**SecondNeedleHeight**[=Int32]

Object

Required. An object of the type ""ScreenItem" with the following format:

- Clock

Int32

Optional. A value or a constant that specifies the length of the second hand.

Specify the length of the second hand as a percentage relating to the radius of the clock face.

See also

Clock (Page 287)

SecondNeedleWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the width of the second hand.

Access in runtime Read and write

Syntax

Object.**SecondNeedleWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

Int32

Optional. A value or a constant that specifies the width of the second hand. Specify the width as a percentage relating to double the length of the second hand.

See also

Clock (Page 287)

SecurityForSimpleViewEnabled (RT Advanced)

Description

No access in runtime.

SegmentColoring (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the type of color change to be displayed if the limits are exceeded.

Access in runtime Read and write

Syntax

Object.**SegmentColoring**[=THmiBarColorType]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

THmiBarColorType

Optional. A value or a constant that specifies the type of color change. Value range from 0 to 1.

Value	VB constant	Description
0	hmiBarColorEntire	Color change applied to the entire bar.
1	hmiBarColorSegmented	Color change applied to segments.

See also

Bar (Page 258)

SelectArchiveName (RT Professional)**Description**

Specifies that the selection dialog for the data source of the recipe view is displayed initially in runtime.

Access in runtime: Read and write

Syntax

Object.**SelectArchiveName**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- UserArchiveControl

BOOLEAN

Optional.

TRUE, if the selection dialog for the data source of the recipe view is displayed in the screen.

FALSE, if no selection dialog for the data source of the recipe view is displayed in the screen.

See also

UserArchiveControl (Page 469)

SelectBackColor (RT Professional)

Description

Specifies the background color of the selected text entry.

Access in runtime Read

Syntax

Object.**SelectBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicIOField

Color

Optional. A value or a constant that specifies the background color of the selected text entry.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

SymbolicIOField (Page 424)

SelectedCellColor (RT Professional)

Description

Specifies the background color of the selected cell.

Access in runtime Read and write

Syntax

Object.**SelectedCellColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant which specifies the background color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectedCellForeColor (RT Professional)**Description**

Specifies the font color of the selected cell.

Access in runtime Read and write

Syntax

Object.**SelectedCellForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant which specifies the font color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectedID (RT Professional)**Description**

Returns the ID of the data record selected in the recipe view.

Access in runtime Read and write

Syntax

Object.**SelectedID**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- UserArchiveControl

Int32

Optional. A value or a constant that returns the ID of the selected data record.

Value	Description
0	None valid data record selected, e.g. in the event of a connection error.
-1	The edit line is selected.

See also

UserArchiveControl (Page 469)

SelectedIndex (RT Professional)**Description**

Defines the index of which the associated text is highlighted in the combo box or list box.

The maximum value is the number of lines (NumberLines) of the object.

Access in runtime: Read and write

Syntax

Object.**SelectedIndex**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- ComboBox
- ListBox

Int32

Optional A value or a constant that specifies the index of the text highlighted.

See also

ComboBox (Page 290)

Listbox (Page 340)

SelectedRowColor (RT Professional)**Description**

Specifies the background color of the selected row.

Access in runtime Read and write

Syntax

Object.**SelectedRowColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant which specifies the background color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectedRowForeColor (RT Professional)**Description**

Specifies the font color of the selected row.

Access in runtime Read and write

Syntax

Object.**SelectedRowForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant which specifies the font color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectedText (RT Professional)**Description**

Specifies the text for the entry referenced with "SelectedIndex".

Access in runtime Read and write

Syntax

Object.**SelectedText**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- ComboBox
- ListBox

STRING

Optional. A value or a constant that specifies the text for an entry.

See also

ComboBox (Page 290)

Listbox (Page 340)

SelectedTitleColor (RT Professional)

Description

Specifies the background color of the selected table header.

Also set "UseSelectedTitleColor" to "TRUE".

Access in runtime Read and write

Syntax

Object.**SelectedTitleColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant which specifies the background color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectedTitleForeColor (RT Professional)

Description

Specifies the font color of the selected table header.

Also set "UseSelectedTitleColor" to "TRUE".

Access in runtime Read and write

Syntax

Object.**SelectedTitleForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant which specifies the font color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

UserArchiveControl (Page 469)

TrendRulerControl (Page 444)

OnlineTableControl (Page 348)

AlarmControl (Page 236)

SelectForeColor (RT Professional)

Description

Specifies the color of the selected text entry.

Access in runtime Read

Syntax

Object.**SelectForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicIOField

Color

Optional. A value or a constant that specifies the color of the selected text entry.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

SymbolicIOField (Page 424)

SelectionBackColor (RT Advanced)

Description

Specifies the background color of the selected cells.

Access in runtime Read and write

Syntax

Object.**SelectionBackColor**[=Color]

Object

Required. A "ScreenItem" object with the format:

- AlarmView
- RecipeView*
- StatusForce
- UserView

*: Read access only

You have no access in runtime with the following format:

- TrendView

Color

Optional. A value or constant that specifies the background color of the selected row.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

SelectionColoring (RT Professional)

Description

Specifies whether selection colors are used for cells or rows.

Access in runtime Read and write

Syntax

Object.**SelectionColoring**[=GridSelectionColoring]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionColoring

Optional. A value or a constant that specifies whether highlight colors are used for cells or rows.

Value	Description	Explanation
0	None	No selection colors for cells and rows.
1	Cell	Selection color for cell.
2	Row	Selection color for row.
3	Cell and row	Selection colors for cell and row.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectionForeColor (RT Advanced)**Description**

Specifies the foreground color of the selected cells.

Access in runtime Read and write

Syntax

Object.**SelectionForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmView
- RecipeView*
- StatusForce
- UserView

*: Read access only

You have no access in runtime with the following format:

- TrendView

Color

Optional. A value or constant that specifies the background color of the selected row.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

SelectionRect (RT Professional)**Description**

Specifies whether a selection frame is used for the selected cells or rows.

Access in runtime Read and write

Syntax

Object.**SelectionRect**[=GridSelectionBorder]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionBorder

Optional. A value or a constant that specifies whether a selection border is used for the selected cell or rows.

Value	Description	Explanation
0	None	No selection border is drawn for selected cells or rows.
1	Cell	A selection border is drawn for the selected cell.
2	Row	A selection border is drawn for the selected row.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectionRectColor (RT Professional)**Description**

Specifies the color of the selection rectangle in the alarm window.

Also set "SelectionType" to "1" or "2".

Access in runtime Read and write

Syntax

Object.**SelectionRectColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant which specifies the color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectionRectWidth (RT Professional)

Description

Specifies the line width of the selection rectangle in the alarm window if SelectionType equals "1".

Access in runtime: Read and write

Syntax

Object.**SelectionRectWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional A value or a constant which specifies the line width.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SelectionType (RT Professional)**Description**

Specifies the number of lines you can mark.

Access in runtime: Read and write

Syntax

Object.**SelectionType**[=GridSelectionType]

Object

Required. A "ScreenItem" object with the format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionType

Optional A value or a constant that specifies the number of lines that you can select.

The following settings are available:

Value	Description	Explanation
0	None	No line is selected.
1	Single selection	One line can be selected.
2	Multiple selection	Several lines can be selected.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

SeparateLineForAlarmText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

SeparatorBackColor (RT Professional)

Description

Specifies the background color of the broken separation lines in the selection list.

Access in runtime Read

Syntax

Object.**SeparatorBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicOField

Color

Optional. A value or a constant that specifies the background color of the broken separation lines in the selection list.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

[SymbolicIOField](#) (Page 424)

SeparatorColor (RT Professional)

Description

Specifies the color of the separation lines in the selection list.

Access in runtime: Read and write

Syntax

Object.**SeparatorColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- S7GraphOverview
- SymbolicIOField *

* Read access only

Color

Optional. A value or a constant that specifies the color of the separation lines in the selection list.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

SeparatorCornerStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns the shape of the corners for the separation line.

Access in runtime: Read

Syntax

Object.**SeparatorCornerStyle**[=CornerStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- SymbolicOField

CornerStyle

Optional. A value or a constant that returns the shape of the corners of the separation line.

Value	Description
0	Solid line
1	Dashed line
2	Dotted line
3	Dashed and dotted line
4	Dash-dot-dot line

SeparatorLineEndShapeStyle (RT Professional)**Description**

Returns the shape of the line ends for the separation line.

Access in runtime: Read

Syntax

Object.**SeparatorLineEndShapeStyle**[=LineEndShapeStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- SymbolicOField

LineEndShapeStyle

Optional. A value or a constant that returns the shape of the line ends for the separation line.

Value	Designation
-1	Flush
0	Rounded
1	Rectangular

SeparatorStyle (RT Professional)**Description**

Specifies the line type of the separation lines in the selection list.

Access in runtime: Read

Syntax

Object.**SeparatorStyle**[=LineStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- SymbolicIOField

LineStyle

Optional. A value or a constant that specifies the type of separation lines in the selection list.

Value	VB constant	Description
-1	hmiLineStyleNone	The selection list has no separation lines.
0	hmiLineStyleSolid	The selection list has solid separation lines.
1	hmiLineStyleDash	The selection list has dashed separation lines.
2	hmiLineStyleDot	The selection list has dotted separation lines.
3	hmiLineStyleDashDot	The selection list has dash-dot separation lines.
4	hmiLineStyleDashDotDot	The selection list has dash-dot-dot separation lines.

SeparatorWidth (RT Professional)**Description**

Specifies the width of the separation lines in the selection list.

Access in runtime Read

Syntax

Object.**SeparatorWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicIOField

Int32

Optional. A value or a constant that specifies the width of separation lines in the selection list.

See also

SymbolicIOField (Page 424)

ServerNames (RT Professional)

Description

Specifies the servers of a distributed system from which the object receives data. The information is given in the following form: NameServer1;NameServer2;NameServer3.

Access in runtime Read and write

Syntax

Object.**ServerNames**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

STRING

Optional. A value or a constant that specifies the servers of a distributed system from which the object receives data.

See also

AlarmControl (Page 236)

ServerPrefix (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the server on which the screen is located that is displayed in the screen window in runtime or returns the server name.

Enter the server name followed by two colons: "<Server name>:". There is no check whether the server actually exists.

Access in Runtime: Read and write

Syntax

Object.**ServerPrefix**[=String]

Object

Required. An object of the type "ScreenItem" with the following format:

- ScreenWindow

String

Optional. A value or a constant which specifies the server name.

ServerScale (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

SetOfVisibleColumns (RT Advanced)**Description**

No access in runtime.

Shared (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShareSpaceWithSourceControl (RT Professional)**Description**

Specifies that the display area of the data source is adapted to the size of the value table.
Access in runtime Read and write

Syntax

Object.**ShareSpaceWithSourceControl**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- TrendRulerControl

BOOLEAN

Optional.

TRUE if the size of the data source is adjusted to the size of the value table.

FALSE; if the size of the data source is not adapted to the size of the value table.

See also

TrendRulerControl (Page 444)

ShiftDecimalPoint (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowAcknowledgeButton (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowAlarmsFromDate (RT Advanced)

Description

Specifies that only those message events are displayed that are saved in this tag.

Access in runtime: Read and write

Syntax

Object.**ShowAlarmsFromDate**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the format:

- AlarmView

HmiObjectHandle

Optional A value or a constant which specifies that only those message events are displayed that are saved in this tag.

See also

AlarmView (Page 251)

ShowAlarmsToAcknowledge (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowBadTagState (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Defines whether the object is grayed out when a bad quality code or tag status is detected.

Access in runtime: Read and write

Syntax

Object.**ShowBadTagState**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- CheckBox
- IOField
- OptionGroup
- SymbolicIOField *
- WindowSlider *

* Read access only

BOOLEAN

Optional.

TRUE if the object is grayed out or the settings for the grid color are used when the quality code or the the tag status is poor.

ShowBar (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the displayed process value is also shown by a filled bar.

Access in runtime Read and write

Syntax

Object.**ShowBar**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Slider

BOOLEAN

Optional. TRUE if the process value is also shown by a filled bar.

See also

Slider (Page 410)

ShowCaption (RT Professional)

Description

Specifies whether the title bar is shown.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ShowCaption**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- ApplicationWindow
- ScreenWindow

You have no access in runtime with the following format:

- Switch

BOOLEAN

Optional. TRUE if the header is shown.

ShowColumnHeaders (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowControls (RT Professional)

Description

Specifies that a toolbar is displayed.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ShowControls**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- MediaPlayer

You have no access in runtime with the following format:

- SmartClientView

BOOLEAN

Optional.

TRUE, if a toolbar is displayed.

FALS E, if no toolbar will be displayed.

ShowDate (RT Advanced)

Description

No access in runtime.

ShowDecimalPoint (RT Professional)

Description

Specifies whether the scale is identified with decimal figures (decimal point and one decimal place) or with whole integers.

Access in runtime: Read and write

Syntax

Object.**ShowDecimalPoint**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the scale is identified with decimal figures (decimal point and one decimal place).

See also

Gauge (Page 320)

ShowDropDownButton (RT Professional)

Description

Returns whether a button for the selection list is displayed.

Access in runtime Read

Syntax

Object.**ShowDropDownButton**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicIOField

BOOLEAN

Optional.

TRUE if a button is displayed for the selection list.

FALSE, if no button for the selection list is displayed.

See also

SymbolicIOField (Page 424)

ShowDropDownList (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowFeatureBackward (RT Professional)

Description

Specifies that the "Backward" button is to be displayed in runtime.

Access in runtime Read and write

Syntax

Object.**ShowFeatureBackward**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the "Backward" button is displayed.

FALSE if no "Backward" button is displayed.

See also

MediaPlayer (Page 343)

ShowFeatureForward (RT Professional)

Description

Specifies that the "Forward" button is displayed in runtime.

Access in runtime Read and write

Syntax

Object.**ShowFeatureForward**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the "Forward" button is displayed.

FALSE if no "Forward" button is displayed.

See also

MediaPlayer (Page 343)

ShowFeatureFullScreen (RT Professional)

Description

Specifies that the Media Player can be displayed as a full screen.

Access in runtime Read and write

Syntax

Object.**ShowFeatureFullScreen**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the "Full screen" button is displayed in the toolbar of the Media Player.

FALSE, if no "Full screen" button is displayed in the toolbar of the Media Player.

See also

MediaPlayer (Page 343)

ShowFeatureFullVolume (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowFeaturePause (RT Professional)

Description

Specifies that the "Pause" button is displayed in runtime.

Access in runtime: Read and write

Syntax

Object.**ShowFeaturePause**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the "Pause" button is displayed in runtime.

FALSE, if no "Pause" button is displayed in runtime.

See also

MediaPlayer (Page 343)

ShowFeaturePlay (RT Professional)**Description**

Specifies that the "Play" button is displayed in runtime.

Access in runtime: Read and write

Syntax

Object.**ShowFeaturePlay**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the "Play" button is displayed in runtime.

FALSE, if no "Play" button is displayed in runtime.

See also

MediaPlayer (Page 343)

ShowFeatureStop (RT Professional)**Description**

Specifies that the "Stop" button is displayed in runtime.

Access in runtime: Read and write

Syntax

Object.**ShowFeatureStop**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the "Stop" button is displayed in runtime.

FALSE, if no "Stop" button is displayed in runtime.

See also

MediaPlayer (Page 343)

ShowFillLevel (RT Professional)

Description

Specifies whether the selected object is filled.

Access in runtime: Read and write

Syntax

Object.**ShowFillLevel**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon

- Rectangle
- RoundButton*
- TextField
- WindowsSlider*

*: Read access only

BOOLEAN

Optional.TRUE, if the selected object is filled.

ShowFocusRectangle (RT Professional)**Description**

Specifies whether the button is given a selection border when it is activated in Runtime.

Access in runtime: Read and write

Syntax

Object.**ShowFocusRectangle**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Clock

BOOLEAN

Optional TRUE, if the button is given a selection border when it is activated in Runtime.

See also

Clock (Page 287)

ShowHelpButton (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowHorizontalGridlines (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowInnerDial (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowLargeTicksOnly (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether only large scale marks are shown.

Access in runtime: Read and write

Syntax

Object.**ShowLargeTicksOnly**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE if only large scale marks are shown.

See also

Bar (Page 258)

ShowLeadingZeros (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowLimitLines (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowLimitRanges (RT Professional)**Description**

No access in runtime.

ShowLimitMarkers (RT Professional)**Description**

Specifies whether the limit values are shown as a scale value.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ShowLimitMarkers**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

You have no access in runtime with the following format:

- Slider

BOOLEAN

Optional. TRUE, if the limit value is shown as a scale value.

ShowLoopInAlarmButton (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowMilliseconds (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowNavigationButtons (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowPathInformation (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowPeakValuePointer (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether a slave pointer will be used for the selected object.

The slave pointer shows the maximum pointer deflection provided in Runtime while the process screen is loaded. If you reload the process screen, the slave pointer will be reset.

Access in runtime: Read and write

Syntax

Object.**ShowPeakValuePointer**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the slave pointer is used.

See also

Gauge (Page 320)

ShowPendingAlarms (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowPosition (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the value of the current slider position should also be displayed numerically. The value is then displayed beneath the slider.

Access in runtime: Read and write

Syntax

Object.**ShowPosition**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Slider

BOOLEAN

Optional TRUE if the value is also shown numerically.

See also

Slider (Page 410)

ShowProcessValue (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowReadButton (RT Advanced)**Description**

No access in runtime.

ShowRuler (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether a scale division (help line) is displayed for an axis label of the object.

Access in runtime: Read and write

Syntax

Object.**ShowRuler**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl
- TrendView

BOOLEAN

Optional. TRUE, if the scale gradation is shown.

ShowRulerInAxis (RT Professional)

Description

Specifies whether rulers are also displayed in the time axes.

Access in runtime Read and write

Syntax

Object.**ShowRulerInAxis**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the rulers are also displayed in the time axes.

FALSE if the rulers are not displayed in the time axes.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

ShowScale (RT Professional)

Description

Specifies whether the values are also shown in a scale.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ShowScale**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar

You have no access in runtime with the following format:

- Slider

BOOLEAN

Optional. TRUE, if the values will also be shown in a scale.

ShowScrollBar (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowScrollbars (RT Professional)

Description

Specifies whether scroll bars are displayed.

Access in runtime: Read and write

Syntax

Object.**ShowScrollbars**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl
- ScreenWindow

BOOLEAN

Optional. TRUE if scroll bars are displayed.

ShowSignForPositiveLabel (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowSortButton (RT Professional)**Description**

Specifies whether the sorting button is shown above the vertical scroll bar. Click this sorting button to sort the selected column based on the configured sorting criteria. The sorting button is not displayed if the table does not contain a vertical scroll bar.

Access in runtime Read and write

Syntax

Object.**ShowSortButton**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the sort button is displayed. You can sort the selected column.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

ShowSortIcon (RT Professional)**Description**

Specifies whether the sorting icon is displayed.
Access in runtime Read and write

Syntax

Object.**ShowSortIcon**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the sort icon is displayed.

FALSE if the sort icon is not displayed.

See also

AlarmControl (Page 236)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

ShowSortIndex (RT Professional)**Description**

Specifies whether a sort index is displayed.

Access in runtime Read and write

Syntax

Object.**ShowSortIndex**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if a sort index is displayed.

FALSE if a sort index is not displayed.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ShowSplittedView (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ShowStatisticRuler (RT Professional)

Description

Specifies whether the lines that define the statistics area are shown.

Access in runtime: Read and write

Syntax

Object.**ShowStatisticRuler**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE, if the lines for the statistics area are shown.

FALSE, if no lines for the statistics area are shown.

See also

OnlineTrendControl (Page 360)

ShowStatusBar (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the status bar is shown.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ShowStatusBar**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- HTMLBrowser

You have no access in runtime with the following format:

- MediaPlayer

BOOLEAN

Optional. TRUE, if the status bar is shown.

ShowTableGridlines (RT Advanced)**Description**

Specifies whether grid lines are shown in the table.

Access in runtime:

- RT Advanced: Read and write
- RT Professional: No access

Syntax

Object.**ShowTableGridlines**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- StatusForce

You have no access in runtime with the following format:

- TrendView
- UserView

BOOLEAN

Optional. TRUE if grid lines are shown in the table.

ShowThumb (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the slider of the "Slider" object will be displayed.

Access in runtime:

- RT Advanced: Read and write
- RT Professional: Read

Syntax

Object.**ShowThumb**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Slider

BOOLEAN

Optional. TRUE, if the slider is shown.

See also

Slider (Page 410)

ShowTickLabels (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the label is shown in the scale.

Access in runtime

- RT Advanced: Read and write
- RT Professional: Read

Syntax

Object.**ShowTickLabels**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Slider

You have no access in runtime with the following format:

- Bar

BOOLEAN

Optional. TRUE if the label is shown.

Comments

The increments for the measured value are automatically established dependent upon the specified measurement range and the size of the object.

ShowTicks (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the marks in the scale are displayed.

Access in runtime Read and write

Syntax

Object.**ShowTicks**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock
- Slider*

* RT Professional, read access only

BOOLEAN

Optional. TRUE if the marks are shown.

See also

Clock (Page 287)

Slider (Page 410)

ShowTime (RT Advanced)

Description

No access in runtime.

ShowTimeAxis (RT Advanced)

Description

No access in runtime.

ShowTimeAxisLabeling (RT Advanced)

Description

No access in runtime.

ShowTitle (RT Professional)

Description

Specifies the style of the window border and window title for the object.

Access in runtime Read and write

Syntax

Object.**ShowTitle**[=WindowHeaderStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

WindowHeaderStyle

Optional. A value or constant that specifies the style of the window border for the object.

Value	Description
0	Neither window border not title are displayed.
1	A narrow window border with title is displayed.
2	A standard window border with title is displayed.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ShowToolBar (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowToolBarBackgroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ShowTracker (RT Professional)**Description**

Specifies that the tracker is displayed.

Access in runtime: Read and write

Syntax

Object.**ShowTracker**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

BOOLEAN

Optional.

TRUE, if the tracker is displayed.

FALSE, if no tracker is displayed.

See also

MediaPlayer (Page 343)

ShowTrendIcon (RT Professional)

Description

Specifies whether an icon is displayed below the value axes. The icon indicates the trend currently displayed in the foreground.

Access in runtime Read and write

Syntax

Object.**ShowTrendIcon**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the icon is displayed.

FALSE if the icon is not displayed.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

ShowTrendIndicator (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the tendency (increasing or falling) of the measurement value that is to be monitored is shown with a small arrow.

Access in runtime: Read and write

Syntax

Object.**ShowTrendIndicator**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the tendency (increasing or falling) of the measurement value that is to be monitored is shown with a small arrow.

See also

Bar (Page 258)

ShowValueAxis1 (RT Advanced)**Description**

No access in runtime.

ShowValueAxis1Label (RT Advanced)**Description**

No access in runtime.

ShowValueAxis2 (RT Advanced)**Description**

No access in runtime.

ShowValueAxis2Label (RT Advanced)

Description

No access in runtime.

ShowValueTable (RT Advanced)

Description

No access in runtime.

ShowWriteButton (RT Advanced)

Description

No access in runtime.

ShowY1HlpLine (RT Advanced)

Description

No access in runtime.

ShowY2HlpLine (RT Advanced)

Description

No access in runtime.

SimpleView (RT Advanced)

Description

No access in runtime.

SimpleViewToolbar (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Size (RT Professional)

Description

No access in runtime.

Sizeable (RT Professional)

Description

Specifies that the size of an object can be changed in Runtime.

Access in Runtime: Read and write

Syntax

Object.**Sizeable**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Sizable

Optional. A value or a constant that specifies whether the size of the object can be modified.

See also

TrendRulerControl (Page 444)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

UserArchiveControl (Page 469)

SmartTags (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns the SmartTags list.

Access in Runtime: Read

Syntax

Object.SmartTags

Object

Required A ""HMIRuntime" object.

See also

HMIRuntime (Page 208)

SortByTimeDirection (RT Advanced)

Description

Specifies whether the last alarm received is shown at the top (ascending sorting order).

Access in runtime Read and write

Syntax

Object.**SortByTimeDirection**[=SortByTimeDirection]

Object

Required. An object of the type "ScreenItems" with the following format:

- AlarmView

SortByTimeDirection

Optional. A value or a constant that specifies the sorting order in the object.

Value	Designation	Description
0	Descending	The last alarm received is shown at the bottom.
1	Ascending	The last alarm received is shown at the top.

See also

AlarmView (Page 251)

SortByTimeEnabled (RT Advanced)

Description

Specifies whether alarm sorting can be altered according to the time.

Access in runtime Read and write

Syntax

Object.**SortByTimeEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItems" with the following format:

- AlarmView

BOOLEAN

Optional. TRUE if sorting can be altered by the device operator.

See also

AlarmView (Page 251)

SortSequence (RT Professional)

Description

Specifies how the sorting order is changed if the operator clicks on the column header in runtime.

Access in runtime: Read and write

Syntax

Object.**SortSequence**[=GridSortSequence]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSortSequence

Optional. A value or a constant that specifies how the sorting order will be changed by a mouse click.

Value	Description	Explanation
0	Up/down/none	You can toggle between ascending, descending and no sorting by means of mouse click.
1	Up/down	You can toggle between ascending and descending sorting order by means of mouse click.

SourceControl (RT Professional)**Description**

Specifies the trend or table view that is connected to the value table.

Access in runtime: Read and write

Syntax

Object.**SourceControl**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

HmiObjectHandle

Optional. A value or a constant that specifies the trend or table view that is connected to the value table.

See also

TrendRulerControl (Page 444)

SourceControlType (RT Professional)**Description**

Specifies the type of data source.

Access in runtime: Read and write

Syntax

Object.**SourceControlType**[=TrendRulerControlSourceControlType]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

TrendRulerControlSourceType

Optional. Value or constant that specifies the type of the data source of the value table.

Value	Designation	Explanation
0	None	Specifies no type for the data source of the value table.
1	f(t) trend view	Specifies that the data source is an f(t) trend view
2	Table view	Specifies that the data source is a table view
3	f(x) trend view	Specifies that the data source is an f(x) trend view

See also

TrendRulerControl (Page 444)

SplittedViewRatio (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

StartAngle (RT Professional)**Description**

Specifies the angle by which the start point deviates from the zero position (0°).

Access in runtime: Read and write

Syntax

Object.**StartAngle**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- CircleSegment
- CircularArc
- EllipseSegment
- EllipticalArc
- TubeArcObject

Int32

Optional. A value or a constant that specifies the angle by which the start point deviates from the zero position (0°).

See also

CircleSegment (Page 282)

CircularArc (Page 285)

EllipseSegment (Page 303)

EllipticalArc (Page 306)

TubeArcObject (Page 460)

StartLeft (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

StartStyle (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies how the line start is displayed.

Access in runtime: Read and write

Syntax

Object.**StartStyle**[=LineEndStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector
- Line
- Polyline

LineEndStyle

Optional. A value or constant which determines the line start. Value range from 0 to 6.

Value	VB constant	Explanation
0	hmiLineEndStyleNone	The line has no start symbol.
1	hmiLineEndStyleArrow	The line starts with an arrowhead.
2	hmiLineEndStyleFilledArrow	The line starts with a filled arrowhead.

Value	VB constant	Explanation
3	hmiLineEndStyleFilledArrowReversed	The line starts with a reversed arrowhead.
4	hmiLineEndStyleLine	The line starts with a vertical line.
5	hmiLineEndStyleCircle	The line starts with a circle.
6	hmiLineEndStyleFilledCircle	The line starts with a filled circle.

StartTop (RT Professional)

Description

No access in runtime.

StartValue (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Defines the absolute value of the zero point of the scale indicator.

Access in runtime: Read and write

Syntax

Object.**StartValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the absolute value of the zero point for the scale indicator.

See also

Bar (Page 258)

State (RT Professional)

Description

Returns the status of a message.

The following table shows the possible states of a message:

State	Alarm Log Status
1	Came In
2	Went Out
5	Came in and comment
6	Gone and comment

StatisticAreaColumns (RT Professional)

Description

No access in runtime.

StatisticResultColumns (RT Professional)

Description

No access in runtime.

StatusbarBackColor (RT Professional)

Description

Specifies the background color for status bars.

Also set "StatusbarUseBackColor" to "TRUE".

Access in runtime Read and write

Syntax

Object.**StatusbarBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the background color of the status bar.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementAdd (RT Professional)**Description**

Creates a new, user-defined status bar element. The newly created element is automatically referenced with "StatusbarElementIndex".

Access in runtime Read and write

Syntax

Object.**StatusbarElementAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Required. A value or a constant that specifies the name of the new user-defined element of the status bar with "StatusbarElementName".

See also

StatusbarElementIndex (Page 968)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementAutoSize (RT Professional)**Description**

Specifies whether the width of a status bar element referenced with "StatusbarElementIndex" is set automatically.

Access in runtime Read and write

Syntax

Object.**StatusbarElementAutoSize**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the width of the selected element is set automatically.

FALSE if the width of the selected element is not set automatically.

See also

UserArchiveControl (Page 469)
TrendRulerControl (Page 444)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
AlarmControl (Page 236)

StatusbarElementCount (RT Professional)

Description

Specifies the number of configured status bar elements.
Access in runtime Read and write

Syntax

Object.**StatusbarElementCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the number of configured status bar elements.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

StatusbarElementIconId (RT Professional)

Description

References a status bar element with its icon ID.

The referencing of an element with its icon ID does not depend on the actual order of the elements.

Access in runtime Read and write

Syntax

Object.**StatusbarElementIconId**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the status bar element to be edited with the icon ID.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementID (RT Professional)

Description

References a status bar element with its element ID. To access the properties of a status bar element, you need to set ""StatusbarElementID"".

The referencing of an element with its element ID does not depend on the actual order of the elements. The element ID are listed in the inspector window of the object under "Properties > Properties > Status bar > Status bar - Elements".

Access in runtime Read and write

Syntax

Object.**StatusbarElementID**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the status bar element to be edited with the element ID.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementIndex (RT Professional)

Description

References a status bar element. To access the properties of a status bar element, you need to set ""StatusbarElementIndex"".

Values between 0 and "StatusbarElementCount" minus 1 are valid for "StatusbarElementIndex". The "StatusbarElementCount" property specifies the number of configured status bar elements.

Access in runtime Read and write

Syntax

Object.**StatusbarElementIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the status bar element to be edited with the index.

See also

StatusbarElementCount (Page 965)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementName (RT Professional)

Description

Specifies the name of a status bar element referenced with "StatusbarElementIndex".

Access in runtime Read and write

Syntax

Object.**StatusbarElementName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl

- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the name of the referenced status bar element.

See also

StatusbarElementIndex (Page 967)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementRemove (RT Professional)**Description**

Removes the user-defined status bar element referenced with its name.

Access in runtime Read and write

Syntax

Object.**StatusbarElementRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the name of the referenced user-defined element of the status bar to be removed.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

StatusbarElementRename (RT Professional)

Description

Specifies the new name of a user-defined status bar element referenced with "StatusbarElementIndex".

Access in runtime Read and write

Syntax

Object.**StatusbarElementRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or constant that specifies the new name of the selected user-defined status bar element.

See also

StatusbarElementIndex (Page 967)
AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementRepos (RT Professional)

Description

Specifies the position of an element referenced with "StatusbarElementIndex" in the status bar of the object.

If you have changed the position of the element with "StatusbarElementRepos", "StatusbarElementIndex" is assigned the value of "StatusbarElementRepos".

Access in runtime Read and write

Syntax

Object.**StatusbarElementRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the position of the referenced element in the status bar. Value range from 0 to "StatusbarElementCount" minus 1. Values outside this range have no effect.

0: The element referenced is positioned on the left.

See also

StatusbarElementIndex (Page 967)

StatusbarElementCount (Page 965)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElements (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

StatusbarElementText (RT Professional)

Description

Specifies the text of a user-defined status bar element referenced with "StatusbarElementIndex".

Access in runtime Read and write

Syntax

Object.**StatusbarElementText**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the text of the selected element of the status bar.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementTooltipText (RT Professional)

Description

Specifies the text for the tooltip of a user-defined status bar element referenced with "StatusbarElementIndex".

Access in runtime Read and write

Syntax

Object.**StatusbarElementTooltipText**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the tooltip text for the selected user-defined status bar element.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementUserDefined (RT Professional)

Description

Specifies whether the configuration engineer has added a status bar element referenced with "StatusbarElementIndex" as a new user-defined element.

Access in runtime Read and write

Syntax

Object.**StatusbarElementUserDefined**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the status bar element is user-defined.

FALSE if the status bar element is specified by the system.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementVisible (RT Professional)

Description

Specifies whether the status bar element referenced with "StatusbarElementIndex" is displayed in the object.

Access in runtime Read and write

Syntax

Object.**StatusbarElementVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional. TRUE if the referenced status bar element is displayed.

See also

StatusbarElementIndex (Page 967)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarElementWidth (RT Professional)**Description**

Specifies the width in pixels of a status bar element referenced with "StatusbarElementIndex".

Also set "StatusbarElementAutoSize (Page 964)" to "FALSE".

Access in runtime Read and write

Syntax

Object.**StatusbarElementWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or constant that specifies the width of the referenced status bar element in pixels.

See also

StatusbarElementIndex (Page 967)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarFont (RT Professional)

Description

Specifies the font of the texts in the status bar.

Access in runtime: Read and write

Syntax

Object.**StatusbarFont**[=Font]

Object

Required. A "ScreenItem" object with the following characteristics:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Font

Optional. A value or constant that specifies the font of the texts in the status bar.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

StatusbarFontColor (RT Professional)

Description

Specifies the color of the texts in the status bar.
Access in runtime Read and write

Syntax

Object.**StatusbarFontColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or constant that specifies the color of the texts in the status bar.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

StatusbarShowTooltips (RT Professional)

Description

Specifies whether tooltips are displayed for the elements of the status bar in runtime.
Access in runtime Read and write

Syntax

Object.**StatusbarShowTooltips**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the tooltips are shown.

FALSE if the tooltips are not shown.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarText (RT Professional)

Description

Specifies the default text in the status bar.

Access in runtime: Read and write

Syntax

Object.**StatusbarText**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the default text in the status bar.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarUseBackColor (RT Professional)

Description

Specifies whether the background color of the status bar is shown.

Access in runtime Read and write

Syntax

Object.**StatusbarUseBackColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the background color of the status bar is displayed.

FALSE if the background color of the status bar is not displayed.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatusbarVisible (RT Professional)

Description

Specifies whether the status bar of the control is displayed.

Access in runtime Read and write

Syntax

Object.**StatusbarVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the status bar is displayed.

FALSE if the status bar is not displayed.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

StatuslineFont (RT Advanced)**Description**

No access in runtime.

StepBackColor (RT Professional)**Description**

Specifies the background color of the step.

Access in runtime Read and write

Syntax

Object.**StepBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- S7GraphOverview

You have no access in runtime with the following format:

Color

Optional. A value or a constant that specifies the background color of the step.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

S7GraphOverview (Page 403)

StepFont (RT Professional)

Description

Specifies the font color of the step.

Access in runtime Read and write

Syntax

Object.**StepFont**[=Font]

Object

Required. An object of the type "ScreenItem" with the following format:

- S7GraphOverview

Font

Optional. A value or a constant that specifies the font.

See also

S7GraphOverview (Page 403)

StepSeconds (RT Professional)

Description

Specifies the step interval in seconds after actuating the "Forward" or "Backward" button.

Access in runtime: Read and write

Syntax

Object.**StepSeconds**[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- MediaPlayer

Int32

Optional. A value or a constant that specifies the step interval in seconds after actuating the "Forward" or "Backward" button.

See also

MediaPlayer (Page 343)

StepTextColor (RT Professional)

Description

Specifies the text color of the step.

Access in runtime Read and write

Syntax

Object.**StepTextColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- S7GraphOverview

You have no access in runtime with the following format:

Color

Optional. A value or a constant that specifies the background color of the step.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

S7GraphOverview (Page 403)

Style (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the line style.

Access in runtime: Read and write

Syntax

Object.**Style**[=LineStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- CircularArc
- Connector
- EllipticalArc
- Line
- Polyline

LineStyle

Optional. A value or a constant that specifies the line style. Value range from 0 to 4.

Value	VB constant	Explanation
0	<code>hmiLineStyleSolid</code>	Solid line
1	<code>hmiLineStyleDash</code>	Dashed line
2	<code>hmiLineStyleDot</code>	Dotted line
3	<code>hmiLineStyleDashDot</code>	Dash-dot line
4	<code>hmiLineStyleDashDotDot</code>	Dash-dot-dot line

StyleItem (RT Professional)**Description**

No access in runtime.

StyleSettings (RT Professional)**Description**

Specifies the style in which the object is displayed.

Access in runtime Read and write

Syntax

Object.**StyleSettings**[=WinCCStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- Button
- RoundButton*
- Window Slider*

*: Read access only

WinCCStyle

Optional. A value or a constant that specifies the style in which the object is displayed.

Value	Designation	Description
0	User Defined	Specifies that the object is displayed according to your settings.
1	Global	Specifies that the object is displayed in the globally set design.
2	Windows Style	Specifies that the object is displayed in the Windows style.

SupportsInplaceEdit (RT Professional)**Description**

No access in runtime.

SupportsS7DiagnosticsInSimpleView (RT Advanced)**Description**

No access in runtime.

SupportsUserDefinedToolbarButtons (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

SwapFirstWithLastConnection (RT Professional)

Description

Specifies that the first and last connection should be swapped.

Access in Runtime: Read and write

Syntax

Object.**SwapFirstWithLastConnection**[=Boolean]

Object

Required. An object of the type "ScreenItem" with the format:

- Connector

Boolean

Optional. A value or a constant which specifies that the first and last connection are swapped.

See also

Connector (Page 293)

SwitchOrientation (RT Advanced)

Description

No access in runtime.

SymbolTableFont (RT Professional)

Description

Specifies the font of the symbol table in the PLC code display.

Access in runtime: Read and write

Syntax

Object.**SymbolTableFont**[=Font]

Object

Required. A "ScreenItem" object with the following characteristics:

- PLCCodeViewer

Font

Optional. A value or a constant that specifies the font in the symbol table in the PLC code display.

See also

PLCCodeViewer (Page 377)

SysDiagBuffButtonVisible (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

1.5.5.13 Properties T (Panels, Comfort Panels, RT Advanced, RT Professional)**TabIndex (RT Professional)****Description**

Specifies the position that the object takes in the order if you toggle between the objects with the tabulator key in Alpha-Cursor mode.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read

Syntax

Object.**TabIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- DiscSpaceView

You have no access in runtime with the following format:

-

Int32

Optional. A value or a constant which specifies the position in the tabulator order.

TabIndexAlpha (RT Professional)

Description

Specifies the position that the object takes in the order if you toggle between the objects with the tabulator key in Alpha-Cursor mode.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read

Syntax

Object.**TabIndexAlpha**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- DiscSpaceView

You have no access in runtime with the following format:

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge

- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- S7GraphOverview
- ScreenWindow
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject

- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowsSlider
- ZoneLabelView
- ZoneQualityView

Int32

Optional. A value or a constant which specifies the position in the tabulator order.

TableBackColor (RT Advanced)**Description**

Specifies the background color of the table cells.

Access in runtime Read and write

Syntax

Object.**TableBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmView
- RecipeView*
- StatusForce
- TrendView
- UserView

*: Read access only

Color

Optional. A value or constant which determines the background color of the table cells.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TableColor (RT Professional)

Description

Specifies the background color of the table rows in the object.

Access in runtime Read and write

Syntax

Object.**TableColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or constant that specifies the background color of the table rows.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TableColor2 (RT Professional)

Description

Specifies the second background color of the table rows in the object.

Also set "UseTableColor2 (Page 1161)" to "TRUE".

Access in runtime Read and write

Syntax

Object.**TableColor2**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or constant that specifies the second background color of the table rows.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TableColumnsWidthAndOrder (RT Advanced)

Description

No access in runtime.

TableEvenRowBackColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TableFont (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableForeColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the text color used in the table cells of the object.

Access in runtime: Read and write

Syntax

Object.**TableForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- AlarmView
- OnlineTableControl
- RecipeView*
- StatusForce
- TrendRulerControl
- UserArchiveControl
- UserView

*: Read access only

Color

Optional. A value or a constant that specifies the text color used in the table cells.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TableForeColor2 (RT Professional)

Description

Specifies the second text color used in the table cells of the object.

Also set "UseTableColor2 (Page 1161)" to "TRUE".

Access in runtime Read and write

Syntax

Object.**TableForeColor2**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the second font color used in the table cells.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TableGridLineColor (RT Advanced)

Description

Specifies the color of the grid lines in the table.

Access in runtime:

- RT Advanced: Read and write
- RT Professional: No access

Syntax

Object.**TableGridLineColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- RecipeView*
- TrendView*
- UserView

*: Read access only

You have no access in runtime with the following format:

- StatusForce
- SysDiagControl

Color

Optional. A value or constant which determines the color of the gridlines of the table.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TableHeaderBackColor (RT Advanced)

Description

Specifies the background color in the header of the table.

Access in runtime Read and write

Syntax

Object.**TableHeaderBackColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmView
- RecipeView*
- StatusForce
- TrendView*
- UserView

*: Read access only

Color

Optional. A value or a constant that specifies the background color in the header.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TableHeaderBackFillStyle (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderBorderBackColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderBorderColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderBorderWidth (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderCornerRadius (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderEdgeStyle (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderFirstGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderFirstGradientOffset (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderFont (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderForeColor (RT Advanced)**Description**

Specifies the text color in the header of the table.

Access in runtime Read and write

Syntax

Object.**TableHeaderForeColor**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- AlarmView
- RecipeView*
- StatusForce*
- TrendView*
- UserView

*: Read access only

Color

Optional. A value or a constant that specifies the text color in the header.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TableHeaderMiddleGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TableHeaderPaddingBottom (RT Advanced)

Description

No access in runtime.

TableHeaderPaddingLeft (RT Advanced)

Description

No access in runtime.

TableHeaderPaddingRight (RT Advanced)**Description**

No access in runtime.

TableHeaderPaddingTop (RT Advanced)**Description**

No access in runtime.

TableHeaderSecondGradientColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TableHeaderSecondGradientOffset (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

Tag4DataRecord (RT Advanced)**Description**

No access in runtime.

Tag4RecipeNumber (RT Advanced)**Description**

No access in runtime.

TagForExternalTime (RT Advanced)**Description**

No access in runtime.

TagPrefix (RT Professional)

Description

Specifies the tag prefix that is prefixed for all tags contained in the screen window. In this way, a screen that is embedded in a screen window retains access to its own tags while another accesses other tags.

The change to the tag prefix does not become effective until the screen is reloaded. The change is effected automatically during a screen change. Without screen change the change is only effect if the screen name is re-assigned.

The prefix is freely definable but must match the name of the structure tag.

Note

Assigning a user data type

You can also select a user data type for a screen that is integrated in a screen window. Afterwards, the data type elements are assigned directly from the selected user data type.

Note

Tag prefix versus indirectly addressed tags

Avoid use of tag prefixes and indirectly addressed tags at the same time.

Access in Runtime: Read and write

Syntax

Object.**TagPrefix**[= STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- ScreenWindow

STRING

Optional. A value or a constant that specifies the tag prefix.

Example

The "InOutPut" screen should be displayed in the screen window. The "InOutPut" screen contains three I/O fields that are linked to a structure tag. The structure tag consists of the elements .IO1, .IO2, .IO3; one element for each I/O field. The element name is preceded by a period. The period is necessary for addressing the elements of the structure tags as structure elements with the correct syntax.

Three such structure tags with the structure names Struct1, Struct2 and Struct3 are defined in the project, for example.

In this case, the tag prefix is the structure name. Specify "Struct2" as tag prefix, for example. The I/O fields in the "InOutPut" screen are then linked to the elements of structure tag "Struct2".

Tag prefix: "Struct2":

- Output value (first IO field): .IO1
- Output value (second IO field): .IO2
- Output value (third IO field): .IO3

The current tag connection in the screen window:

- Output value (first I/O field): Struct2.EA1
- Output value (second I/O field): Struct2.EA2
- Output value (third I/O field): Struct2.EA3

See also

ScreenWindow (Page 405)

Tags (RT Professional)

Description

Returns an object of type "Tags".

Access during runtime: Read

Syntax

Object.**Tags**

Object

Necessary. An object of the "HMIRuntime" type.

Example

The following example accesses the tag "Tag1":

```
'VBS86
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
```

See also

HMIRuntime (Page 208)

TcpPortNr (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Template (RT Professional)

Description

Specifies the template for displaying the window content in the "Print job/Script diagnostic" object.

Access in Runtime: Read and write

Syntax

Object.**Template**[=TemplateType]

Object

Required. An object of the type "ScreenItem" with the following format:

- ApplicationWindow

TemplateType

Optional. A value or a constant that specifies the template.

The following templates are possible depending on the value of the property "WindowsContents (Page 1226)":

Window Contents = Global Script

- "GSC diagnostics"
The "Script diagnostic" object is supplied by applications of the Global Script. The results of the diagnosis system are displayed.
- "GSC Runtime"
The "Script diagnostic" object is supplied by applications of the Global Script. The analysis results regarding characteristics in Runtime are displayed.

Window Contents = Print Jobs

- "All Jobs":
The "Print job" object is supplied by the logging system. The available reports are displayed as a list.
- "All Jobs - Context Menu":
The "Print job" object is supplied by the logging system. The available reports are displayed as a list. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the log.

- "Job Detail View":
The "Print job" object is supplied by the logging system. The available reports are displayed in a selection menu. Detailed information is displayed for the selected report.
- "Selected Jobs - Context Menu":
The "Print job" object is supplied by the logging system. The available reports are displayed as a list. This list only contains reports which you have activated the option "Mark for print job list" in the "Print Job Properties" dialog. The shortcut menu enables the selection of print options, display of a print preview as well as a printout of the log.

See also

ApplicationWindow (Page 256)

Text (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the label for the text field.

Access in runtime: Read and write

Syntax

Object.**Text**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- CheckBox
- ComboBox
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- TextField

STRING

Optional. A value or a constant that specifies the labeling.

TextAreaBottomMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TextAreaLeftMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TextAreaRightMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TextAreaTopMargin (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TextHandles (RT Professional)

Description

No access in runtime.

TextList (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns the text list that supplies the object with values.

Access in runtime

- RT Advanced: No access
- RT Professional: Read

Syntax

Object.**TextList**[=HmiObjectHandle]

Object

Required. An object of the type "ScreenItem" with the following format:

- SymbolicIOField

You have no access in runtime with the following formats:

- Button

HmiObjectHandle

Optional. A value or a constant that returns the text list that supplies the object with values.

See also

SymbolicIOField (Page 424)

Button (Page 266)

TextOff (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the text that will be displayed in the "Off" status.

Access in runtime Read and write

Syntax

Object.**TextOff**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- Button
- Switch*

* RT Advanced read access only, RT Professional no access

You have no access in runtime with the following formats:

- SymbolicIOField

STRING

Optional. A value or a constant that specifies the labeling for the "off" status.

Comments

The property is only available if the referenced object is of the "Text" type.

TextOn (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the text that will be displayed in the "On" status.

Access in runtime: Read and write

Syntax

Object.**TextOn**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- Switch*

* RT Advanced read access only, RT Professional no access

You have no access in runtime with the following formats:

- SymbolicIOField

STRING

Optional. A value or a constant that specifies the labeling for the "on" status.

Comments

The property is only available if the referenced object is of the "Text" type.

TextOrientation (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the text orientation.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**TextOrientation**[=TextOrientation]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- IOField
- OptionGroup
- RoundButton*
- SymbolicIOField*

- TextField
- WindowSlider*

*: Read access only

You have no access in runtime with the following formats:

- DateTimeField
- Switch

TextOrientation

Optional. A value or a constant that specifies the text orientation.

Value	VB constant	Explanation
0	hmiTextHorizontal	The text is shown horizontally.
-1	hmiTextRotated90Degree	The text is shown vertically and left-justified.
1	hmiTextRotated270Degree	The text is shown vertically and right-justified.

Texts (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TextualObjectPositions (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TextualObjectsAutoSize (RT Advanced)

Description

No access in runtime.

TextualObjectsBorderBackColor (RT Advanced)

Description

No access in runtime.

TextualObjectsBorderColor (RT Advanced)

Description

No access in runtime.

TextualObjectsBorderWidth (RT Advanced)

Description

No access in runtime.

TextualObjectsCornerRadius (RT Advanced)

Description

No access in runtime.

TextualObjectsEdgeStyle (RT Advanced)

Description

No access in runtime.

TextualObjectsPaddingBottom (RT Advanced)

Description

No access in runtime.

TextualObjectsPaddingLeft (RT Advanced)

Description

No access in runtime.

TextualObjectsPaddingRight (RT Advanced)

Description

No access in runtime.

TextualObjectsPaddingTop (RT Advanced)

Description

No access in runtime.

ThumbBackColor (RT Professional)

Description

Specifies the background color of the slider.

Access in runtime Read

Syntax

Object.**ThumbBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- Slider
- WindowSlider

Color

Optional. A value or a constant that specifies the background color of the slider.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Slider (Page 410)

WindowSlider (Page 485)

ThumbPicture (RT Professional)

Description

Specifies a graphic for the slider element in the slider.

Access in runtime: Read

Syntax

Object.**ThumbPicture**[=HmiObjectHandle]

Object

Required. A "ScreenItem" object with the following characteristics:

- Slider

HmiObjectHandle

Optional. A value or constant that specifies the graphic for the slider element in the slider.

See also

Slider (Page 410)

TickDistance (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TicksColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the hour markers on the face of the "Clock" object.

Access in runtime Read and write

Syntax

Object.**TicksColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Clock

Color

Optional. A value or a constant that specifies the color of the hour markers.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Clock (Page 287)

TickStyle (RT Professional)

Description

Specifies how the marks are displayed in the scale.

Also set "ShowTicks" to "TRUE".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**TickStyle**[=SliderTickStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- Slider

You have no access in runtime with the following formats:

- Clock

SliderTickStyle

Optional. A value or a constant that specifies how the marks are displayed in the scale.

Value	VB constant	Explanation
0	<code>hmiSliderTickStyleNone</code>	The scale has no marks.
1	<code>hmiSliderTickStyleEffect1</code>	The large scale marks are shown in white with black shadow marks. The small scale marks are shown in black.
2	<code>hmiSliderTickStyleEffect2</code>	The large scale marks are shown in black with white shadow marks. The small scale marks are shown in black.
3	<code>hmiSliderTickStyleNormal</code>	All scale marks are shown in black.

Comments

Automatic scaling can show two marks in the scale so close together that they look like one broad mark. To correct this effect, lengthen or shorten the slider.

TimeAxes (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

TimeAxisAdd (RT Professional)

Description

Creates a new time axis. The newly created time axis is automatically referenced with "TimeAxisIndex".

Access in runtime: Read and write

Syntax

Object.**TimeAxisAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTrendControl

STRING

Required. A value or a constant that specifies the name of the new time axis with "TimeAxisName".

TimeAxisAlignment (RT Professional)

Description

Specifies the alignment of the time axis.

Access in runtime: Read and write

Syntax

Object.**TimeAxisAlignment**[=AxisAlignment]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

AxisAlignment

Optional. A value or a constant that specifies the alignment.

Value	Designation	Explanation
0	Bottom	The time axis selected is displayed below the trend.
1	Top	The time axis selected is displayed above the trend.

See also

OnlineTrendControl (Page 360)

TimeAxisBegin (RT Advanced)**Description**

No access in runtime.

TimeAxisBeginTime (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the start time for the display of the time axis referenced with "TimeAxisIndex".

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**TimeAxisBeginTime**[=DateTime]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTrendControl

You have no access in runtime with the following format:

- TrendView

DateTime

Optional. A value or a constant that specifies the start time for the display of the specified trend.

TimeAxisColor (RT Professional)

Description

Specifies the color of the time axis referenced with "TimeAxisIndex".

Access in runtime Read and write

Syntax

Object.**TimeAxisColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the time axis.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTrendControl (Page 360)

TimeAxisCount (RT Professional)

Description

Specifies the number of configured time axes.

Access in runtime Read and write

Syntax

Object.**TimeAxisCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the number of configured time axes.

See also

OnlineTrendControl (Page 360)

TimeAxisCountPoints (RT Advanced)**Description**

No access in runtime.

TimeAxisDateFormat (RT Professional)**Description**

Specifies the date format for displaying the time axis referenced with "TimeAxisIndex".

Access in runtime Read and write

Syntax

Object.**TimeAxisDateFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the date format.

See also

OnlineTrendControl (Page 360)

TimeAxisEnd (RT Advanced)**Description**

No access in runtime.

TimeAxisEndTime (RT Professional)

Description

Specifies the end time for the display of the selected trend. Whether or not the information is evaluated depends on the "Autorange", "UseTimeRange(i)" and "ShareTimeAxis" properties.

Access in runtime: Read and write

Syntax

Object.**TimeAxisEndTime**[=DateTime]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTrendControl

DateTime

Optional A value or a constant that specifies the end time for the display of the specified trend.

See also

OnlineTrendControl (Page 360)

TimeAxisIndex (RT Professional)

Description

References a time axis. To access the properties of a time axis, you need to set ""TimeAxisIndex"".

Values between 0 and "TimeAxisIndex" minus 1 are valid for "TimeAxisCount". The "TimeAxisCount" property specifies the number of configured time axes.

Access in runtime Read and write

Syntax

Object.**TimeAxisIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies, with the index, the time axis to be edited.

See also

TimeAxisCount (Page 1014)

OnlineTrendControl (Page 360)

TimeAxisInTrendColor (RT Professional)**Description**

Specifies whether the color of the time axis referenced with "TimeAxisIndex" corresponds to the trend color.

Access in runtime Read and write

Syntax

Object.**TimeAxisInTrendColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the referenced axis is shown in the trend color. The value of "TimeAxisColor" is not effective.

FALSE if the referenced axis is shown in the color specified in "TimeAxisColor".

See also

TimeAxisIndex (Page 1016)

OnlineTrendControl (Page 360)

TimeAxisLabel (RT Professional)**Description**

Specifies the label text for a time axis referenced with "TimeAxisIndex".

Access in runtime Read and write

Syntax

Object.**TimeAxisLabel**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the label text for the referenced time axis.

See also

TimeAxisIndex (Page 1016)

OnlineTrendControl (Page 360)

TimeAxisMeasurePoints (RT Professional)

Description

Specifies the number of measuring points displayed for the time axis referenced with "TimeAxisIndex".

Access in runtime Read and write

Syntax

Object.**TimeAxisMeasurePoints**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the number of measurement points.

See also

OnlineTrendControl (Page 360)

TimeAxisMode (RT Advanced)

Description

No access in runtime.

TimeAxisName (RT Professional)

Description

Specifies the name of a time axis referenced with "TimeAxisIndex".

Access in runtime Read and write

Syntax

Object.**TimeAxisName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced time axis.

See also

TimeAxisIndex (Page 1016)

OnlineTrendControl (Page 360)

TimeAxisOnline (RT Professional)

Description

Specifies that the time axis is updated on an ongoing basis.

Access in runtime: Read and write

Syntax

Object.**TimeAxisOnline**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE, if the time axis is updated on an ongoing basis.

FALSE, if the time axis is not updated on an ongoing basis.

See also

OnlineTrendControl (Page 360)

TimeAxisRange (RT Advanced)**Description**

No access in runtime.

TimeAxisRangeType (RT Professional)**Description**

Defines the time range setting for the time axis referenced with "TimeAxisIndex".

Access in runtime Read and write

Syntax

Object.**TimeAxisRangeType**[=TimeRangeMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

TimeRangeMode

Optional. A value or a constant that specifies the settings for the time range.

The following settings are available:

Value	Designation	Description
0	Time span	Start time and time range are specified for the time column.
1	End time	Start time and end time are specified for the time column.
2	Measurement points	Defines the start time and the number of measurement points for the time column.

See also

OnlineTrendControl (Page 360)

TimeAxisRemove (RT Professional)**Description**

Removes a time axis referenced with its name.

Access in runtime Read and write

Syntax

Object.**TimeAxisRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that returns the name of the referenced time axis to be removed.

See also

OnlineTrendControl (Page 360)

TimeAxisRename (RT Professional)

Description

Specifies the new name of a time axis referenced with "TimeAxisIndex".

Access in runtime Read and write

Syntax

Object.**TimeAxisRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the new name of the selected time axis.

See also

TimeAxisIndex (Page 1016)

OnlineTrendControl (Page 360)

TimeAxisRepos (RT Professional)

Description

Specifies the position of a time axis referenced with "TimeAxisIndex" in the trend diagram of the object.

If you have changed the position of the time axis with "TimeAxisRepos", "TimeAxisIndex" is assigned the value of "TimeAxisRepos".

Access in runtime Read and write

Syntax

Object.**TimeAxisRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the position of the referenced time axis. Value range from 0 to "TimeAxisCount" minus 1. Values outside this range have no effect.

0: The referenced time axis is positioned outside.

See also

TimeAxisIndex (Page 1016)

TimeAxisCount (Page 1014)

OnlineTrendControl (Page 360)

TimeAxisShowDate (RT Professional)

Description

Specifies whether the time axis referenced with "TimeAxisIndex" is displayed with date and time.

Access in runtime Read and write

Syntax

Object.**TimeAxisShowDate**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE, if date and time are displayed. The date format is specified using the "TimeAxisDateFormat" property.

FALSE, if the date is not displayed. Only the time is displayed.

See also

OnlineTrendControl (Page 360)

TimeAxisSide (RT Advanced)**Description**

No access in runtime.

TimeAxisTimeFormat (RT Professional)**Description**

Specifies the format of the information along the time axis for the selected trend.

Access in runtime Read and write

Syntax

Object.**TimeAxisTimeFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant which defines the format for the time axis.

See also

OnlineTrendControl (Page 360)

TimeAxisTimeRange (RT Advanced)**Description**

No access in runtime.

TimeAxisTimeRangeBase (RT Professional)

Description

Specifies the time unit that defines the time range together with the time factor "TimeAxisTimeRangeFactor".

Access in runtime Read and write

Syntax

Object.**TimeAxisTimeRangeBase**[=TagLoggingTimeUnit]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

TagLoggingTimeUnit

Optional. A value or a constant that specifies the time unit.

The following settings are available:

Value	Designation
500	500 ms
1000	1 second
60000	1 minute
3600000	1 hour
86400000	1 day

See also

OnlineTrendControl (Page 360)

TimeAxisTimeRangeFactor (RT Professional)

Description

Specifies the time factor that together with time unit "TimeAxisTimeRangeBase" determines the time period.

Access in runtime: Read and write

Syntax

Object.**TimeAxisTimeRangeFactor**[=Int16]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

Int16

Optional. A value or a constant that specifies the time factor.

See also

OnlineTrendControl (Page 360)

TimeAxisTrendWindow (RT Professional)**Description**

Specifies the trend diagram in which the axis referenced with "TimeAxisIndex" is displayed.

Access in runtime Read and write

Syntax

Object.**TimeAxisTrendWindow**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the trend diagram.

See also

TrendWindowCount (Page 1130)

TimeAxisIndex (Page 1016)

OnlineTrendControl (Page 360)

TimeAxisVisible (RT Professional)**Description**

Specifies whether the time axis referenced with "TimeAxisIndex" is displayed in the object.

Access in runtime Read and write

Syntax

Object.**TimeAxisVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the referenced time axis is shown.

FALSE if the referenced time axis is hidden.

See also

TimeAxisIndex (Page 1016)

OnlineTrendControl (Page 360)

TimeBase (RT Professional)

Description

Specifies the time zone in which the time values are displayed.

Access in runtime Read and write

Syntax

Object.**TimeBase**[=TimeBase]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- UserArchiveControl

TimeBase

Optional. A value or a constant that specifies the time zone.

Value	VB constant	Explanation
0	hmiTimeBaseLocalTime-zone	Local time
1	hmiTimeBaseServerTimezone	Time zone of the server
2	hmiTimeBaseUTC	UTC (Universal Time Coordinated)
3	hmiTimeBaseProjectSetting	Project settings

Comments

You can specify the time mode for a particular HMI device in the project tree, in the object properties of the HMI device.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

UserArchiveControl (Page 469)

TimeColumnActualize (RT Professional)**Description**

Specifies whether the values of the time column referenced with "TimeColumnIndex" are updated.

Access in runtime Read and write

Syntax

Object.**TimeColumnActualize**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the time column is updated.

FALSE if the time column is not updated. This setting is useful for comparing tables.

See also

OnlineTableControl (Page 348)

TimeColumnAdd (RT Professional)

Description

Creates a new time column. The newly created time column is automatically referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Required. A value or a constant that specifies the name of the new time column with "TimeColumnName".

See also

TimeColumnIndex (Page 1035)

OnlineTableControl (Page 348)

TimeColumnAlignment (RT Professional)

Description

Specifies how the text in the time column referenced with "TimeColumnIndex" is aligned.

Access in runtime Read and write

Syntax

Object.**TimeColumnAlignment** [=HorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

HorizontalAlignment

Optional. A value or a constant that specifies how the text in the referenced time column is aligned.

Value	VB constant	Explanation
0	hmiAlignmentLeft	The text is left-justified.
1	hmiAlignmentCentered	The text is centered.
2	hmiAlignmentRight	The text is right-justified.

See also

OnlineTableControl (Page 348)

TimeColumnBackColor (RT Professional)**Description**

Specifies the background color of the time column referenced with "TimeColumnIndex".

Also set "UseColumnBackColor (Page 1150)" to "TRUE" and "TimeColumnUseValueColumnColors (Page 1045)" to "FALSE".

Access in runtime Read and write

Syntax

Object.**TimeColumnBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Color

Optional. A value or a constant that specifies the background color of the selected time column.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTableControl (Page 348)

TimeColumnBeginTime (RT Professional)

Description

Specifies the start of the time range for the time column referenced with "TimeColumnIndex".
Access in runtime Read and write

Syntax

Object.**TimeColumnBeginTime**[=DateTime]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

DateTime

Optional. A value or a constant that specifies the starting time for the selected time column.

See also

OnlineTableControl (Page 348)

TimeColumnCaption (RT Professional)

Description

Specifies the label of the time column referenced with "TimeColumnIndex".
Access in runtime Read and write

Syntax

Object.**TimeColumnCaption**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the caption of the time column.

See also

OnlineTableControl (Page 348)

TimeColumnCount (RT Professional)**Description**

Specifies the number of configured time columns.

Access in runtime: Read and write

Syntax

Object.**TimeColumnCount**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int32

Optional A value or a constant that specifies the number of configured time columns.

See also

OnlineTableControl (Page 348)

TimeColumnDateFormat (RT Professional)**Description**

Specifies the date format for displaying the time column referenced with "TimeColumnIndex".

Access in runtime: Read and write

Syntax

Object.**TimeColumnDateFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the date format for displaying the selected time column.

The following date formats are available:

Value	Explanation
dd.MM.yy	Day.Month.Year, e.g., 24.12.13.
dd.yyyd.MM	Day.Month.Year, e.g., 24.12.2013.
dd/MM/yy	Day/Month/Year, e.g., 24/12/13.
dd/MM/yyyy	Day/Month/Year, e.g., 24/12/2013.

TimeColumnEndTime (RT Professional)

Description

Defines the end of the time range of for a time column referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnBeginTime**[=DateTime]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

DateTime

Optional. A value or constant that specifies the end time.

See also

OnlineTableControl (Page 348)

TimeColumnForeColor (RT Professional)

Description

Specifies the font color of the time column referenced with "TimeColumnIndex".

Also set "UseColumnForeColor (Page 1150)" to "TRUE" and "TimeColumnUseValueColumnColors (Page 1045)" to "FALSE".

Access in runtime Read and write

Syntax

Object.**TimeColumnForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Color

Optional. A value or a constant that specifies the font color of the selected time column.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTableControl (Page 348)

TimeColumnHideText (RT Professional)**Description**

Specifies whether the content of the time column referenced with "TimeColumnIndex" is displayed as text.

Access in runtime Read and write

Syntax

Object.**TimeColumnHideText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the content is not displayed as text.

FALSE if the content is displayed as text.

See also

OnlineTableControl (Page 348)

TimeColumnHideTitleText (RT Professional)

Description

Specifies whether the header of the time column referenced with "TimeColumnIndex" is displayed as text.

Access in runtime Read and write

Syntax

Object.**TimeColumnHideTitleText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the header is not displayed as text.

FALSE if the header is displayed as text.

See also

OnlineTableControl (Page 348)

TimeColumnIndex (RT Professional)

Description

References a configured time column. To access the properties of a time column, you need to set ""TimeColumnIndex"".

Values between 0 and "TimeColumnCount" minus 1 are valid for "TimeColumnIndex". The property "TimeColumnCount" specifies the number of configured time columns.

Access in runtime: Read and write

Syntax

Object.**TimeColumnIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies, with the index, the time column to be edited.

TimeColumnLength (RT Professional)**Description**

Specifies the width of the time column referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnLength**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies the width of the selected time column.

See also

OnlineTableControl (Page 348)

TimeColumnMeasurePoints (RT Professional)**Description**

Specifies the number of measuring points to be displayed in the time column referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnMeasurePoints**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Int32

Optional. A value or constant that specifies the number of measurement points.

See also

OnlineTableControl (Page 348)

TimeColumnName (RT Professional)

Description

Specifies the name of a time column referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the name of the referenced time column.

See also

TimeColumnIndex (Page 1034)

OnlineTableControl (Page 348)

TimeColumnRangeType (RT Professional)

Description

Specifies the time range for a time column referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnRangeType**[=TimeRangeMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

TimeRangeMode

Optional. A value or a constant that specifies the time range for the selected time column.

Value	Description	Explanation
0	Time range	Start time and time range are specified for the time column.
1	Start to end time	Start time and end time are specified for the time column.
2	Number of measurement points	Defines the start time and the number of measurement points for the time column.

See also

OnlineTableControl (Page 348)

TimeColumnRemove (RT Professional)**Description**

Removes a time column referenced with its name.

Access in runtime Read and write

Syntax

Object.**TimeColumnRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that returns the name of the referenced time column to be removed.

See also

OnlineTableControl (Page 348)

TimeColumnRename (RT Professional)**Description**

Specifies the new name of a time column referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the new name of the selected time column.

See also

TimeColumnIndex (Page 1034)

OnlineTableControl (Page 348)

TimeColumnRepos (RT Professional)

Description

For multiple time columns with corresponding value columns, specifies the position of a time column referenced with "TimeColumnIndex".

If you have changed the position of the time column with "TimeColumnRepos", "TimeColumnIndex" is assigned the value of "TimeColumnRepos".

Access in runtime Read and write

Syntax

Object.**TimeColumnRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies the position of the referenced time column with the corresponding value columns. Value range from 0 to "TimeColumnCount" minus 1. Values outside this range have no effect.

0: The referenced time column is positioned on the left.

See also

TimeColumnIndex (Page 1034)

TimeColumnCount (Page 1031)

OnlineTableControl (Page 348)

TimeColumns (RT Professional)**Description**

No access in runtime.

TimeColumnShowDate (RT Professional)**Description**

Specifies whether the time column referenced with "TimeColumnIndex" is displayed with date and time.

Also set the date format with "TimeColumnDateFormat (Page 1031)".

Access in runtime Read and write

Syntax

Object.**TimeColumnShowDate**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE, if date and time are displayed.

FALSE if only the time is displayed.

See also

OnlineTableControl (Page 348)

TimeColumnShowIcon (RT Professional)

Description

Specifies whether the content of the time column referenced with "TimeColumnIndex" is displayed as an icon.

Access in runtime: Read and write

Syntax

Object. [=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the content is displayed as an icon.

FALSE if the content is not displayed as an icon.

TimeColumnShowTitleIcon (RT Professional)

Description

Specifies whether the header of the time column referenced with "TimeColumnIndex" is displayed as an icon.

Access in runtime Read and write

Syntax

Object.**TimeColumnShowTitleIcon**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the header is shown as an icon.

FALSE if the header is not shown as an icon.

See also

OnlineTableControl (Page 348)

TimeColumnSort (RT Professional)**Description**

Specifies how a time column referenced with "TimeColumnIndex" is sorted.

Access in runtime Read and write

Syntax

Object.**TimeColumnSort**[=SortMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

SortMode

Optional. A value or a constant that specifies how the time column referenced in "TimeColumnIndex" is sorted.

Value	Description	Explanation
0	No	No sorting
1	Ascending	Ascending order, starting at the lowest value.
2	Descending	Descending order, starting at the highest value.

See also

OnlineTableControl (Page 348)

TimeColumnSortIndex (RT Professional)**Description**

Specifies the sorting order of the time column referenced by "TimeColumnIndex". If you set the value to "0", the sorting criterion is removed in "TimeColumnSort".

Access in runtime: Read and write

Syntax

Object.**TimeColumnSortIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTableControl

Int32

Optional. Value or constant that specifies the sorting order of the time column referenced by "TimeColumnIndex". If you set the value to "0", the sorting criterion is removed in "TimeColumnSort".

TimeColumnTimeFormat (RT Professional)**Description**

Specifies the time format for displaying the time column referenced with "TimeColumnIndex".

Access in runtime Read and write

Syntax

Object.**TimeColumnTimeFormat**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the time format for displaying the selected time column.

Value	Explanation
Automatic	The time format is set automatically.
HH:mm:ss.ms	Hours:Minutes:Seconds, e.g. 15:35:44.240.
hh:mm:ss tt	Hours:Minutes:Seconds AM/PM, e.g. 03:35:44 PM.
hh:mm:ss.ms tt	Hours:Minutes:Seconds.Milliseconds AM/PM, e.g. 03:35:44.240 PM.

See also

OnlineTableControl (Page 348)

TimeColumnTimeRangeBase (RT Professional)**Description**

Specifies the time unit for calculating the time range displayed in the time column referenced with "TimeColumnIndex".

The time range is calculated with a time unit and a time factor. The time factor is specified in "TimeColumnTimeRangeFactor (Page 1044)".

Access in runtime Read and write

Syntax

Object.**TimeColumnTimeRangeBase**[=TagLoggingTimeUnit]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

TagLoggingTimeUnit

Optional. A value or a constant that specifies the time unit for determining the time range.

Value	Description
500	500 ms
1000	1 second
60000	1 minute
3600000	1 hour
86400000	1 day

See also

OnlineTableControl (Page 348)

TimeColumnTimeRangeFactor (RT Professional)

Description

Defines the factor for calculating the time range. Only integer factors are valid.

Access in runtime: Read and write

Syntax

Object.**TimeColumnTimeRangeFactor**[=Int16]

Object

Required. A "ScreenItem" object with the following format:

- OnlineTableControl

Int16

Optional A value or a constant that specifies the factor to determine the time range.

See also

OnlineTableControl (Page 348)

TimeColumnUseValueColumnColors (RT Professional)

Description

Specifies whether the time column referenced with "TimeColumnIndex" is displayed in the colors of the value column.

Access in runtime Read and write

Syntax

Object.**TimeColumnUseValueColumnColors**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the selected time column is displayed in the value column colors. The settings in "TimeColumnBackColor (Page 1029)" and "TimeColumnForeColor (Page 1032)" are disabled.

FALSE if the selected time column is displayed in the colors specified in "TimeColumnBackColor (Page 1029)" and "TimeColumnForeColor (Page 1032)"

See also

OnlineTableControl (Page 348)

TimeColumnVisible (RT Professional)

Description

Specifies whether the time column referenced with "TimeColumnIndex" is displayed in the table view.

Access in runtime Read and write

Syntax

Object.**TimeColumnVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the referenced time column is displayed in the table.

FALSE if the referenced time column is not displayed in the table.

See also

TimeColumnIndex (Page 1034)

OnlineTableControl (Page 348)

TimeDisplayMode (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TimeStamp (RT Professional)**Description**

Returns the time stamp of the last read access of a tag in local time as DATE.

Access in Runtime: Read

Syntax

Object.**TimeStamp**

Object

Required A "Tag" object.

Comments

To show the TimeStamp property in plain text, use the VBS function "FormatDateTime(Date[, NamedFormat])". The output is dependent on the language setting. To adjust the language, use the VBS function "SetLocale()'".

If you want to return the time stamp sorted by date, day and time, use the NamedFormat parameter or the VBS functions like Year, WeekDay, Day, Hour, Minute, Second. The name of a week day can be obtained using the VBS function WeekdayName.

Examples

The following example issues the time stamp of the tag "Tag11" with the aid of the function "FormatDateTime":

```
'VBS87
Dim objTag
Dim lngCount
lngCount = 0
Set objTag = HMIRuntime.Tags("Tag11")
objTag.Read
SetLocale("en-gb")
MsgBox FormatDateTime(objTag.TimeStamp)      'Output: e.g. 06/08/2002 9:07:50
MsgBox Year(objTag.TimeStamp)              'Output: e.g. 2002
MsgBox Month(objTag.TimeStamp)             'Output: e.g. 8
MsgBox Weekday(objTag.TimeStamp)          'Output: e.g. 3
MsgBox WeekdayName(Weekday(objTag.TimeStamp)) 'Output: e.g. Tuesday
MsgBox Day(objTag.TimeStamp)               'Output: e.g. 6
MsgBox Hour(objTag.TimeStamp)              'Output: e.g. 9
MsgBox Minute(objTag.TimeStamp)            'Output: e.g. 7
MsgBox Second(objTag.TimeStamp)            'Output: e.g. 50
For lngCount = 0 To 4
MsgBox FormatDateTime(objTag.TimeStamp, lngCount)
Next
'lngCount = 0: Output: e.g. 06/08/2002 9:07:50
'lngCount = 1: Output: e.g. 06 August 2002
'lngCount = 2: Output: e.g. 06/08/2002
'lngCount = 3: Output: e.g. 9:07:50
'lngCount = 4: Output: e.g. 9:07
```

The following example issues the time stamp of the tag "Tag1":

```
'VBS88
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.TimeStamp
```

See also

Tag (Page 228)

TimeStepBase (RT Professional)

Description

Specifies the time unit for the exactness of the time stamp displayed in the table. The exactness is determined by multiplying the factor and time unit.

Access in runtime: Read and write

Syntax

Object.**TimeStepBase**[=TimeStepBase]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTableControl

TimeStepBase

Optional. A value or a constant that specifies the time unit for the exactness of the time stamp.

Value	Designation	Explanation
0	Exact	Specifies that only values with exactly identical time stamp are displayed within a row of the table.
1000	Base1s	Specifies that the time range is the factor of the base with 1 ms.
100	Base100ms	Specifies that the time range is the factor of the base with 100 ms.
250	Base250ms	Specifies that the time range is the factor of the base with 250 ms.
500	Base500ms	Specifies that the time range is the factor of the base with 500 ms.

See also

OnlineTableControl (Page 348)

TimeStepFactor (RT Professional)

Description

Specifies the exactness of the time stamp that is formed with time unit "TimeStepBase".

You determine the exactness by multiplying the factor and time unit. For example, if you want to display all values that occurred within 3 seconds in the same row, you enter "3" for the factor and "Base1s" for the time unit.

The entered factor has no effect if "exact" is selected for the time unit.

Access in runtime: Read and write

Syntax

Object.**TimeStepFactor**[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies the time factor for the exactness of the time stamp.

See also

[OnlineTableControl](#) (Page 348)

TitleColor (RT Professional)

Description

Specifies the background color of the table header.

Access in runtime Read and write

Syntax

Object.**TitleColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the background color of the table header.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

[AlarmControl](#) (Page 236)

[OnlineTableControl](#) (Page 348)

[TrendRulerControl](#) (Page 444)

[UserArchiveControl](#) (Page 469)

TitleCut (RT Professional)

Description

Specifies whether the content of the fields in a title bar is to be shortened if the column is too narrow.

Access in runtime: Read and write

Syntax

Object.**TitleCut**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional TRUE if the column headers in the title bar are shortened if the column is too narrow.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TitleDarkShadowColor (RT Professional)

Description

Specifies the color for the dark side of the 3D shading for column and row headers in the object table.

Also set "TitleStyle (Page 1055)" to "1".

Access in runtime Read and write

Syntax

Object.**TitleDarkShadowColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color for the dark side of the 3D shading.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TitleForeColor (RT Professional)**Description**

Specifies the text color for the table column and row headers of the object.

Access in Runtime: Read and write

Syntax

Object.**TitleForeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or constant that specifies the font color in the column and row headers of the table.

Comments

You use the "RGB" function to define the color in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). The "Red" color is displayed as follows, for example:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TitleGridLineColor (RT Professional)**Description**

Specifies the color of the separation lines in the title bar of the table.

Access in runtime Read and write

Syntax

Object.**TitleGridLineColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color of the separation lines in the title bar of the table.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TitleLightShadowColor (RT Professional)

Description

Specifies the color for the light side of the 3D shading for table column and row headers of the object.

Also set "TitleStyle (Page 1055)" to "1".

Access in runtime Read and write

Syntax

Object.**TitleLightShadowColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the color for the light side of the 3D shading.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TitleSort (RT Professional)

Description

Specifies how sorting by column header is triggered.

Access in runtime Read and write

Syntax

Object.**TitleSort**[=GridSortTrigger]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSortTrigger

Optional. A value or constant that specifies how to trigger sorting by column header.

Value	Explanation
0	Sorting by column header is disabled.
1	Sorting is triggered by clicking in the column header.
2	Sorting is triggered by double-clicking in the column title.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

TitleStyle (RT Professional)

Description

Specifies whether to use a shading color for the text in the column headers.

Access in runtime Read and write

Syntax

Object.**TitleStyle**[=GridHeaderStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridHeaderStyle

Optional. A value or a constant that specifies whether the text in the columns headers is displayed with shading.

Value	Explanation
0	No shading color is used. Flat header style.
1	Enables the use of shading colors. 3D representation of the header.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

Toggle (RT Professional)

Description

Specifies whether the selected object engages after it has been operated in Runtime.

Access in runtime: Read and write

Syntax

Object.**Toggle**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton*

*: Read access only

BOOLEAN

Optional. TRUE, if the selected object engages after it has been operated in runtime.

Tolerance (RT Professional)**Description**

Specifies the limit for the storage space display as of which a deviation will be reported.

Access in runtime: Read

Syntax

Object.**Tolerance**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- DiscSpaceView

Int32

Optional. A value or a constant that specifies the limit for the disk space view as of which a deviation is reported.

See also

DiscSpaceView (Page 298)

ToleranceColor (RT Professional)**Description**

Specifies the colors in which the bars of the memory space display are shown as soon as the tolerance range is exceeded.

Access in runtime Read

Syntax

Object.**ToleranceColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- DiscSpaceView

Color

Optional. A value or a constant that specifies the colors in which the bars of the storage space display will be shown as soon as the tolerance range is exceeded.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

DiskSpaceView (Page 298)

ToleranceLowerLimit (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Sets the low limit for tolerance 1.

Access in runtime: Read and write

Syntax

Object.**ToleranceLowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the format:

- Bar

DOUBLE

Optional A value or a constant that specifies the low limit for tolerance 1.

Comments

The following values will be defined via the properties "ToleranceLowerLimit", "ToleranceLowerLimitColor" and "ToleranceLowerLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

ToleranceLowerLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the lower limit "ToleranceLowerLimit".

The "ToleranceLowerLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime Read and write

Syntax

Object.**ToleranceLowerLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the lower limit "ToleranceLowerLimit".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

ToleranceLowerLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the "ToleranceLowerLimit" limit is monitored. The limit, the display when the limit has been reached, and the type of evaluation are set with the properties "ToleranceLowerLimit", "ToleranceLowerLimitColor" and "ToleranceLowerLimitRelative".

Access in runtime: Read and write

Syntax

Object.**ToleranceLowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "ToleranceLowerLimit" limit is monitored.

See also

Bar (Page 258)

ToleranceLowerLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the low limit "ToleranceLowerLimit" is evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.**ToleranceLowerLimitRelative**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BOOLEAN

Optional.

TRUE if the low limit "ToleranceLowerLimit" is to be evaluated as a percentage.

FALSE if the low limit "ToleranceLowerLimit" is to be evaluated as an absolute value.

See also

Bar (Page 258)

ToleranceUpperLimit (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Sets the high limit for tolerance 1.

Access in runtime: Read and write

Syntax

Object.**ToleranceUpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the format:

- Bar

DOUBLE

Optional A value or a constant that specifies the high limit for tolerance 1.

Comments

The following values will be specified via the properties "ToleranceUpperLimit", "ToleranceUpperLimitColor" and "ToleranceUpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

ToleranceUpperLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color for the high limit "ToleranceUpperLimit".

Access in runtime Read and write

Syntax

Object.**ToleranceUpperLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the upper limit "ToleranceUpperLimit".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

ToleranceUpperLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies whether the "ToleranceUpperLimit" limit is to be monitored.

Access in runtime: Read and write

Syntax

`Object.ToleranceUpperLimitEnabled[=BOOLEAN]`

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "ToleranceUpperLimit" limit is to be monitored.

Comments

The following values will be specified with the properties "ToleranceUpperLimit", "ToleranceUpperLimitColor" and "ToleranceUpperLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

ToleranceUpperLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the higher limit "ToleranceUpperLimit" is to be evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.**ToleranceUpperLimitRelative**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BOOLEAN

Optional.

TRUE if the high limit "ToleranceUpperLimit" is to be evaluated as a percentage.

FALSE if the high limit "ToleranceUpperLimit" is to be evaluated as an absolute value.

See also

Bar (Page 258)

ToolBar_ButtonsHeight (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ToolBar_ButtonsWidth (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ToolBarAlignment (RT Professional)**Description**

Specifies the position of the toolbar.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ToolBarAlignment**[=ToolBarPosition]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

You have no access in runtime with the following format:

- SysDiagControl

ToolBarPosition

Optional. A value or a constant that specifies the position of the toolbar within the object.

Value	Designation
0	Top
1	Bottom
2	Left
3	Right

ToolbarBackColor (RT Professional)

Description

Specifies the background color of the toolbar.

Also set "ToolbarUseBackColor (Page 1082)" to "TRUE".

Access in runtime Read and write

Syntax

Object.**ToolbarBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

Optional. A value or a constant that specifies the background color of the toolbar.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarBackgroundColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ToolBarButtonActive (RT Professional)

Description

Specifies whether the function linked to the button is activated in runtime. Clicking the button in Runtime triggers the corresponding function.

Access in runtime Read and write

Syntax

Object.**ToolBarButtonActive**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the function assigned to the key is active.

FALSE if the function assigned to the key is not active. You can connect your own functions to the key by local scripts.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonAdd (RT Professional)

Description

Creates a new user-defined button in the object's toolbar. The newly created button is automatically referenced with "ToolBarButtonIndex".

Access in runtime Read and write

Syntax

Object.**ToolBarButtonAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Required. A value or a constant that specifies the name of the new button in the toolbar with "ToolBarButtonName".

See also

[ToolBarButtonIndex](#) (Page 1072)

[AlarmControl](#) (Page 236)

[FunctionTrendControl](#) (Page 309)

[OnlineTableControl](#) (Page 348)

[OnlineTrendControl](#) (Page 360)

[TrendRulerControl](#) (Page 444)

[UserArchiveControl](#) (Page 469)

ToolBarButtonAuthorization (RT Professional)

Description

Specifies the authorization for the selected key function. The authorizations are configured in the user administration.

Access in runtime Read and write

Syntax

Object.**ToolBarButtonAuthorization**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or constant that specifies the authorization for the selected key function.

See also

UserArchiveControl (Page 469)

TrendRulerControl (Page 444)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

ToolBarButtonBeginGroup (RT Professional)

Description

Inserts a leading separator (vertical line) for the selected key function on the toolbar. These separators can be used to group the icons of the button functions.

Access in runtime Read and write

Syntax

Object.**ToolBarButtonBeginGroup**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if a separator is inserted in front of the selected key function.

FALSE if no separator is inserted in front of the selected key function.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonClick (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Clicks a toolbar button. You call the function of the corresponding button using the ID of the button.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**ToolBarButtonClick**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- SysDiagControl*
- TrendRulerControl
- UserArchiveControl

*: Read access only

Int32

Optional. A value or a constant that specifies the ID of the button that is clicked.

ToolBarButtonCount (RT Professional)

Description

Specifies the number of configured buttons in the toolbar.

Access in runtime Read and write

Syntax

Object.**ToolBarButtonCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the number of configured buttons in the toolbar.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonEnabled (RT Professional)

Description

Enables operation of custom toolbar buttons.

Access in runtime: Read and write

Syntax

Object.**ToolBarButtonEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional TRUE if operation is enabled for the selected user-defined key in the toolbar.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonHeight (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ToolBarButtonHotKey (RT Professional)

Description

Specifies the hot key for the selected object button. Enter the hot key in ASCII coding, e.g. "112" for the <F1> key.

Also set "ToolBarUseHotKeys (Page 1083)" to "TRUE".

Access in runtime Read and write

Syntax

Object.**ToolBarButtonHotKey**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the shortcut for the selected button.

See also

TrendIndex (Page 1104)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonID (RT Professional)

Description

References a button with its ID. To access the properties of a status bar element, you need to set ""ToolBarButtonID"".

The referencing of a button with its ID does not depend on the actual order of the buttons. The ID are listed in the inspector window of the object under "Properties > Properties > Toolbar > Toolbar - Buttons".

Access in runtime Read and write

Syntax

Object.**ToolbarButtonID**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies, with the ID, the toolbar buttons to be edited.

See also

[ToolbarButtonClick](#) (Page 1067)

[AlarmControl](#) (Page 236)

[FunctionTrendControl](#) (Page 309)

[OnlineTableControl](#) (Page 348)

[OnlineTrendControl](#) (Page 360)

[TrendRulerControl](#) (Page 444)

[UserArchiveControl](#) (Page 469)

ToolbarButtonIndex (RT Professional)

Description

References a button. To access the properties of a button, you need to set `""ToolbarButtonIndex""`.

Values between 0 and `""ToolbarButtonCount""` minus 1 are valid for `""ToolbarButtonIndex""`. The `""ToolbarButtonCount""` property specifies the number of configured buttons.

Access in runtime Read and write

Syntax

Object.**ToolBarButtonIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies, with the index, the button to be edited.

See also

ToolBarButtonCount (Page 1068)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonLocked (RT Professional)

Description

Specifies whether the locked, pressed state of a user-defined button referenced with "ToolBarButtonIndex" is displayed.

Access in runtime Read and write

Syntax

Object.**ToolBarButtonLocked**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl

- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE if the pressed state is displayed for the referenced user-defined button in the toolbar.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolbarButtonName (RT Professional)**Description**

Specifies the name of a user-defined button referenced with "ToolbarButtonIndex".

Access in runtime Read and write

Syntax

Object.**ToolbarButtonName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the name of the referenced user-defined button.

See also

ToolbarButtonIndex (Page 1071)
AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

ToolbarButtonRemove (RT Professional)

Description

Removes a user-defined button referenced with its name.

Access in runtime Read and write

Syntax

Object.**ToolbarButtonRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the name of the referenced user-defined button to be removed.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonRename (RT Professional)

Description

Specifies the new name of a user-defined button referenced with "ToolBarButtonIndex".

Access in runtime Read and write

Syntax

Object.**ToolBarButtonRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional. A value or a constant that specifies the new name of the selected user-defined button.

See also

ToolBarButtonIndex (Page 1071)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonRepos (RT Professional)

Description

Specifies the position of a button referenced with "ToolBarButtonIndex" in the toolbar of the object.

If you have changed the position of the button with "ToolBarButtonRepos", "ToolBarButtonIndex" is assigned the value of "ToolBarButtonRepos".

Access in runtime Read and write

Syntax

Object.**ToolBarButtonRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

Optional. A value or a constant that specifies the position of the referenced button in the toolbar. Value range from 0 to "ToolBarButtonCount" minus 1. Values outside this range have no effect.

0: The referenced button is positioned on the left.

See also

ToolBarButtonCount (Page 1068)

ToolBarButtonIndex (Page 1071)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtons (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ToolBarButtonSettings (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

ToolBarButtonsForMigration (RT Advanced)**Description**

No access in runtime.

ToolBarButtonTooltipText (RT Professional)**Description**

Specifies the tooltip text for the user-defined button in the toolbar.

Access in runtime: Read and write

Syntax

Object.**ToolBarButtonTooltipText**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

Optional A value or a constant that specifies the tooltip text for the selected user-defined button.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)

ToolbarButtonUserDefined (RT Professional)

Description

Specifies whether the project engineer has added the toolbar button as new user-defined button.

Access in runtime: Read and write

Syntax

Object.**ToolbarButtonUserDefined**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Boolean

Optional.

TRUE if the toolbar key is user-defined.

FALSE if the toolbar key is specified by the system.

See also

AlarmControl (Page 236)
FunctionTrendControl (Page 309)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonVisible (RT Professional)

Description

Specifies whether the button referenced with "ToolBarButtonIndex" is displayed in the toolbar.

Access in runtime Read and write

Syntax

Object.**ToolBarButtonVisible**[=<BOOLEAN>]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the referenced button is shown.

FALSE if the referenced button is hidden.

See also

ToolBarButtonIndex (Page 1071)

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarButtonWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ToolBarEnabled (RT Advanced)

Description

No access in runtime.

ToolBarHeight (RT Advanced)

Description

No access in runtime.

ToolBarIconStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ToolBarLeft (RT Advanced)

Description

No access in runtime.

ToolBarShowTooltips (RT Professional)

Description

Specifies whether the tooltips for the key functions are displayed in runtime. The property can be dynamized with the ToolBarShowTooltips name. The property for specifying the tooltip text is "ToolBarButtonTooltipText".

Access in runtime Read and write

Syntax

Object.**ToolBarShowTooltips**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the tooltips are shown.

FALSE if the tooltips are not shown.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolbarStyle (RT Advanced)**Description**

No access in runtime.

ToolbarTop (RT Advanced)**Description**

No access in runtime.

ToolbarUseBackColor (RT Professional)**Description**

Specifies whether the background color for the toolbar is shown.

Access in runtime Read and write

Syntax

Object.**ToolbarUseBackColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the background color of the toolbar is displayed.

FALSE if the background color of the toolbar is not displayed.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolbarUseHotKeys (RT Professional)

Description

Specifies whether the hotkeys are activated for buttons in the toolbar. The hot keys are defined for each button with "ToolbarButtonHotKey (Page 1070)".

Access in runtime Read and write

Syntax

Object.**ToolbarUseHotKeys**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional. TRUE if the hot keys are activated.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolbarVisible (RT Professional)**Description**

Specifies whether the toolbar of the control is displayed.

Access in runtime Read and write

Syntax

Object.ToolbarVisible[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the toolbar is displayed.

FALSE if the toolbar is not displayed.

See also

AlarmControl (Page 236)

FunctionTrendControl (Page 309)

OnlineTableControl (Page 348)

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

ToolBarWidth (RT Advanced)

Description

No access in runtime.

ToolTipText (RT Advanced, RT Professional)

Description

Specifies the tooltip text.

Access in runtime: Read

Syntax

Object.**ToolTipText**[=STRING]

ToolTipText

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox

- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton *
- StatusForce *
- Switch *
- SymbolicIOField *
- TextField
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowSlider *

* Read access only

STRING

Optional. A value or a constant that specifies the tooltip text.

Top (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the value of the Y coordinate.

Access in runtime: Read

Syntax

Object.**Top**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- AlarmView
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- DateTimeField
- DiscSpaceView
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup

- PLCCodeViewer
 - Polygon
 - Polyline
 - ProDiagOverview
 - ProtectedAreaNameView
 - RangeLabelView
 - RangeQualityView
 - RecipeView *
 - Rectangle
 - RoundButton *
 - S7GraphOverview
 - ScreenWindow
 - Slider **
 - SmartClientView
 - StatusForce *
 - Switch *
 - SymbolLibrary
 - SymbolicIOField *
 - SysDiagControl *
 - TextField
 - TrendRulerControl
 - TrendView *
 - TubeArcObject
 - TubeDoubleTeeObject
 - TubeTeeObject
 - Tubepolyline
 - UserArchiveControl
 - UserView **
 - WlanQualityView
 - WindowSlider *
 - ZoneLabelView
 - ZoneQualityView
- * Read access only
- ** RT Professional read access only

You have no access in runtime with the following format:

- ApplicationWindow

Int32

Optional. A value or a constant that contains the value of the Y coordinate in pixels (measured from the top left edge of the screen).

Comments

The Y coordinate refers to the top left corner of the rectangle that surrounds the object. The screen limits are also monitored in runtime. If the assigned coordinate value exceeds the display size, the user-defined function is interrupted with an error message.

TopMargin (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TopOffset (RT Professional)**Description**

Specifies the vertical shift of zero point of the display of a screen that is larger than the screen window. The shift refers to the top edge of the screen window.

The screen is displayed cut. The picture scroll bars are located at the left and upper edge of the picture.

If you wish to display the screen in the screen window with horizontal and vertical offset of the screen scroll bars, use the "HorizontalScrollBarPosition (Page 741)" and "VerticalScrollbarPosition (Page 1206) properties for the offset.

Access in Runtime: Read and write

Syntax

Object.**TopOffset**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Screenwindow

Int32

Optional. A value or a constant that specifies the vertical shift of the screen display's zero point from the top margin of the screen window.

See also

ScreenWindow (Page 405)

Total (RT Professional)**Description**

Specifies the memory capacity.

Access in runtime: Read

Syntax

Object.**Total**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the format:

- DiskSpaceView

DOUBLE

Optional. A value or a constant that specifies the memory capacity.

TransitionHeaderFont (RT Professional)**Description**

Specifies the font in the information area of this PLC code display.

Access in runtime: Read and write

Syntax

Object.**TransitionHeaderFont**[=Font]

Object

Required. A "ScreenItem" object with the following characteristics:

- PLCCodeViewer

Font

Optional. A value or a constant that specifies the font.

See also

PLCCodeViewer (Page 377)

Transparency (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the object transparency in percent.

0 = no transparency; 100 = complete transparency (invisible).

The text and fields of the graphic objects are only transparent with "100."

In runtime, a completely transparent object (invisible) is also functional.

Access in runtime: Read and write

Syntax

Object.**Transparency**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline

- Rectangle
- RoundButton *
- Slider *
- SymbolicIOField *
- TextField
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowSlider *

* Read access only

** RT Professional, read access only

Int32

Optional. A value or constant that specifies the transparency of the object in percent.

TransparentColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies which color of the allocated graphic (*.bmp, *dib) is to be set to "transparent".

Also specify with "UseTransparentColor" that the color is to be shown as transparent.

Access in runtime: Read and write

Syntax

Object.**TransparentColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- GraphicIOField
- GraphicView

Color

Optional. A value or a constant that specifies the color that will be shown transparent.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TransparentColorDeactivatedPicture (RT Professional)

Description

Sets the color of the assigned bitmap object to "transparent" for the "disabled" status.

Access in runtime: Read

Syntax

Object.**TransparentColorDeactivatedPicture**[=Color]

Object

Required. A "ScreenItem" object with the following format:

- RoundButton

Color

Optional A value or a constant that specifies which color of the allocated bitmap object will be set to transparent" for the status "disabled".

Comments

You can use the "RGB" function to define the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this: RGB(255, 0, 0). You can also use the VBS color constants such as `vbRed` and `vbGreen`.

The "PicDeactUseTransColor" property must have the value TRUE so that the color can be set to "transparent".

See also

RoundButton (Page 399)

TransparentColorPictureOff (RT Professional)

Description

Specifies which color of the allocated bitmap object will be set to transparent" for the "off" status.

Access in runtime Read and write

Syntax

Object.**TransparentColorPictureOff**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Button
- RoundButton*

*: Read access only

Color

Optional. A value or a constant that specifies which color of the allocated bitmap object will be set to transparent" for the "off" status.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TransparentColorPictureOn (RT Professional)

Description

Specifies which color of the allocated bitmap object will be set to transparent" for the "on" status.

The "PicDownUseTransColor" property must have the value TRUE so that the color can be set to "transparent".

Access in runtime Read and write

Syntax

Object.**TransparentColorPictureOn**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Button
- RoundButton*

*: Read access only

Color

Optional. A value or a constant that specifies which color of the allocated bitmap object will be set to "transparent" for the "on" status.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TrendActualize (RT Professional)

Description

Specifies whether the selected trend is updated.

Access in runtime Read and write

Syntax

Object.**TrendActualize**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the selected trend is always updated.

FALSE if the selected trend is not updated. This setting is useful when a logged trend is compared with a current trend.

See also

FunctionTrendControl (Page 309)

TrendAdd (RT Professional)

Description

Creates a new trend. The newly created trend is automatically referenced with "TrendIndex".

Access in runtime: Read and write

Syntax

Object.**TrendAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Required. A value or a constant that specifies the name of the new trend with "TrendName".

TrendAutoRangeBeginTagName (RT Professional)

Description

Specifies a tag that defines the start value of the data range of the trend. You specify additionally using the "TrendAutoRangeSource (Page 1098)" property that the data range of the trend is determined dynamically.

Access in runtime: Read and write

Syntax

Object.**TrendAutoRangeBeginTagName**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the tag for the start value.

See also

OnlineTrendControl (Page 360)

TrendAutoRangeBeginValue (RT Professional)

Description

Specifies the start value of the data range of the trend. You specify additionally using the "TrendAutoRangeSource (Page 1098)" property that the data range of the trend is set statically.

Access in runtime: Read and write

Syntax

Object.**TrendAutoRangeBeginValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the start value of the data range of the trend.

See also

OnlineTrendControl (Page 360)

TrendAutoRangeEndTagName (RT Professional)**Description**

Specifies a tag that defines the end value of the data range of the trend. You specify additionally using the "TrendAutoRangeSource (Page 1098)" property that the data range of the trend is determined dynamically.

Access in runtime: Read and write

Syntax

Object.**TrendAutoRangeEndTagName**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the tag for the end value.

See also

OnlineTrendControl (Page 360)

TrendAutoRangeEndValue (RT Professional)**Description**

Specifies the end value of the data range of the trend. You specify additionally using the "TrendAutoRangeSource (Page 1098)" property that the data range of the trend is set statically.

Access in runtime: Read and write

Syntax

Object.**TrendAutoRangeEndValue**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the end value of the data range of the trend.

See also

OnlineTrendControl (Page 360)

TrendAutoRangeSource (RT Professional)

Description

Specifies how the automatic data range of the trend data is determined.

Access in runtime: Read and write

Syntax

Object.**TrendAutoRangeSource**[=AutorangeSourceType]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTrendControl

AutorangeSourceType

Optional. A value or a constant that specifies the type of data range.

The following settings are available:

Value	Designation	Description
x	Adapt automatically	Specifies that the displayed value range is automatically adapted to the current values.
1	Dynamic range	Specifies that the low limit and high limit of the displayed value range is formed from the values of connected online tags. The tags of the low limit and high limit are mapped in the "TrendAutoRangeBeginTagName (Page 1095)" and "TrendAutoRangeEndTagName (Page 1096)" properties.
2	Static range	Specifies that the displayed value range is defined by the configuration of the low limit and high limit of the value range. The values of the low limit and high limit are mapped in the "TrendAutoRangeBeginValue (Page 1095)" and "TrendAutoRangeEndValue (Page 1096)" properties.

TrendBeginTime (RT Professional)

Description

Defines the start time of the time range for data transfer to the selected trend.

Access in runtime: Read and write

Syntax

Object.**TrendBeginTime**[=DateTime]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

DateTime

Optional. Defines the start time for data supply for the selected trend.

See also

FunctionTrendControl (Page 309)

TrendColor (RT Professional)

Description

Specifies the color of the trend line of a referenced trend of the trend view.

Access in Runtime: Read and write

Syntax

Object.**TrendColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the trend line of a referenced trend in the object.

Comments

You use the "RGB" function to define the color in RGB format (red, green, blue). Enter the corresponding decimal value for each of the three RGB values (range from 0 to 255). The "Red" color is displayed as follows, for example:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

TrendIndex (Page 1104)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendCount (RT Professional)

Description

Specifies the number of configured trends.

Access in runtime Read and write

Syntax

Object.**TrendCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the number of configured trends.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendEndTime (RT Professional)

Description

Defines the end of the time range for data connections of a selected trend.

Access in runtime: Read and write

Syntax

Object.**TrendEndTime**[=DateTime]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

DateTime

Optional. Specifies the end time for data supply for the selected trend.

See also

FunctionTrendControl (Page 309)

TrendExtendedColorSet (RT Professional)

Description

Specifies whether the point and fill colors of the trend are shown.

Access in runtime Read and write

Syntax

Object.**TrendExtendedColorSet**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the settings in "TrendFillColor (Page 1103)" and "TrendPointColor (Page 1111)" are effective.

FALSE if the settings for point and fill color are not effective.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendFill (RT Professional)**Description**

Specifies whether the area below the trend is shown filled.

If "TrendExtendedColorSet (Page 1100)" is set to "FALSE", the trend color is also used as the fill color.

Access in runtime Read and write

Syntax

Object.TrendFill[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the area beneath the trend is filled.

FALSE if the trend is not shown filled.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendFillColor (RT Professional)

Description

Defines the fill color of the trend.

The setting is effective if the properties "TrendFill (Page 1101)" and "TrendExtendedColorSet (Page 1100)" are also set to "TRUE".

Access in runtime Read and write

Syntax

Object.**TrendFillColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the fill color of the selected trend.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendIndex (RT Professional)

Description

References a configured trend. To access the properties of a trend, you need to set `""TrendIndex""`.

Values between 0 and "TrendCount" minus 1 are valid for "TrendIndex". The "TrendCount" property specifies the number of configured trends.

Access in runtime Read and write

Syntax

Object.**TrendIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies, with the index, the trend to be edited.

See also

TrendCount (Page 1099)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendIndicatorColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the trend view. The trend indicator represents the tendency (rising or falling) of the measurement value that is to be monitored with a small arrow. To activate the trend indicator, the property "ShowTrendIndicator" must have the value "TRUE".

Access in runtime Read and write

Syntax

Object.**TrendIndicatorColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color of the trend indicator.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

TrendLabel (RT Professional)

Description

Defines the label text for a trend referenced with "TrendIndex".

Access in runtime: Read and write

Syntax

Object.**TrendLabel**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the label text for the referenced trend.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendLineStyle (RT Professional)

Example

Specifies which line type is used for display of the trend.

Access in runtime Read and write

Syntax

Object.**TrendLineStyle**[=LineStyle]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl

LineStyle

Optional. A value or a constant that specifies the type of line for displaying the trend.

Value	Explanation
0	The trend is shown as a solid line.
1	The trend is visualized as dashed line.
2	The trend is visualized as dotted line.
3	The trend is visualized as dot-dash line.
4	The trend is visualized as dash-dot-dot line.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendLineType (RT Professional)

Description

Specifies how the trend is displayed.

Access in runtime Read and write

Syntax

Object.**TrendLineType**[=TrendLineTypeScada]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl

TrendLineTypeScada

Optional. A value or a constant that specifies how the trend is displayed.

Value	Explanation
0	Only the trend dots are displayed.
1	Visualizes a trend with linear interconnection of points.
2	Visualizes a stepped trend and its interconnected points.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendLineWidth (RT Professional)**Description**

Specifies the line thickness of the selected trend in pixels.

Access in runtime: Read and write

Syntax

Object.**TrendLineWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the line weight of the selected trend in pixels.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendLowerLimit (RT Professional)

Description

Specifies the low limit of a tag that is displayed in the object as a trend. If the tag drops below the value of "TrendLowerLimit", the affected trend dots are marked in the color set in "TrendLowerLimitColor (Page 1108)".

Using the "TrendLowerLimitColoring (Page 1109)" property, specify additionally that the display of the low limit violation is activated.

Access in runtime Read and write

Syntax

Object.**TrendLowerLimit**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the lower limit for a tag displayed in a trend view.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendLowerLimitColor (RT Professional)

Description

Specifies the color that marks the trend values that are below the value of "TrendLowerLimit (Page 1107)".

The setting is effective if the "TrendLowerLimitColoring (Page 1109)" property has the value "TRUE".

Access in runtime: Read and write

Syntax

Object.**TrendLowerLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of trend values below the "TrendLowerLimit (Page 1107)" value.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

TrendLowerLimitColoring (RT Professional)**Description**

Specifies whether "TrendLowerLimitColor" is used to identify tag values which are less than the value of "TrendLowerLimit".

Access in runtime: Read and write

Syntax

Object.**TrendLowerLimitColoring**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the "TrendLowerLimitColor" property is effective.

FALSE if the "TrendLowerLimitColor" property is not effective.

TrendMeasurePoints (RT Professional)

Description

Specifies the number of measurement points for visualization of the selected trend.

Defines the number of value pairs provided to the trend from a user archive.

Access in runtime: Read and write

Syntax

Object.**TrendMeasurePoints**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the number of measurement points or value pairs for the selected trend.

See also

FunctionTrendControl (Page 309)

TrendName (RT Professional)

Description

Specifies the name of a trend referenced with "TrendIndex".

Access in runtime Read and write

Syntax

Object.**TrendName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced trend.

See also

TrendIndex (Page 1103)
FunctionTrendControl (Page 309)
OnlineTrendControl (Page 360)

TrendPointColor (RT Professional)

Description

Specifies the color of the dots on the reference trend.

The setting is effective if the "TrendExtendedColorSet (Page 1100)" property has the value "TRUE".

Access in runtime Read and write

Syntax

Object.**TrendPointColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the dots in the referenced trend.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

TrendIndex (Page 1103)
FunctionTrendControl (Page 309)
OnlineTrendControl (Page 360)

TrendPointStyle (RT Professional)

Description

Specifies how the trend points are displayed.

Access in runtime Read and write

Syntax

Object.**TrendPointStyle**[=PointStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

PointStyle

Optional. A value or a constant that specifies how the dots in the trend are displayed.

Value	Explanation
0	The points are not displayed. The set dot width is disabled.
1	The trend points are visualized with a size of one pixel. The set dot width is disabled.
2	The dots are displayed as square. The set dot width is enabled.
3	The dots are displayed as circles. The set dot width is enabled.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendPointWidth (RT Professional)

Description

Specifies the dot weight in pixels. You can only define the dot weight for "square" and "circular" dots.

Access in runtime: Read and write

Syntax

Object.**TrendPointWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the dot weight of the selected trend in pixels.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendProvider (RT Professional)**Description**

Specifies the data source for a selected trend.

Access in runtime Read and write

Syntax

Object.**TrendProvider**[=DataSourceType]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

DataSourceType

Optional. A value or a constant that specifies the data source for the selected trend.

Value	Description	Explanation
0	None	No data source configured for implementation in Runtime by means of a user-defined function.
1	Logging tags	Data source with archive tags of a process value archive.
2	HMI tags	Data supply with tag values of HMI tags
3	Recipe data	Data supply with columns of a recipe

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendProviderCLSID (RT Professional)**Description**

Specifies the provider CLSID of the data for a trend.

Access in runtime Read and write

Syntax

Object.**TrendProviderCLSID**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the provider CLSID.

Value	Description
	No configured data supply in order to establish a connection in Runtime via a script.
{416A09D2-8B5A-11D2-8B81-006097A45D48}	Data source with archive tags of a process value archive.
{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}	Data supply with online tags from tag management.
{2DC9B1C8-4FC1-41B1-B354-3E469A13FBFD}	Data supply with columns of a user log.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendRangeType (RT Professional)**Description**

Specifies the time range during which the selected trend is supplied with data.

You can only define the number of measuring points if you select user archives as the data source.

Access in runtime Read and write

Syntax

Object.**TrendRangeType**[=TimeRangeMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

TimeRangeMode

Optional. A value or a constant that specifies the time range during which the selected trend is supplied with data.

Value	Description	Explanation
0	Time range	Defines the start time and the time range for the data connection.
1	Start to end time	Defines the start and end time for the data connection.
2	Number of measurement points	Defines the start time and the number of measurement points for the data connection.

See also

FunctionTrendControl (Page 309)

TrendRemove (RT Professional)**Description**

Removes a trend referenced with its name.

Access in runtime Read and write

Syntax

Object.**TrendRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced trend to be removed.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendRename (RT Professional)**Description**

Specifies the new name of a trend referenced with "TrendIndex".

Access in runtime Read and write

Syntax

Object.**TrendRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the new name of the selected trend.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendRepos (RT Professional)**Description**

Specifies the position of a trend referenced with "TrendIndex" in the trend window of the object.

If you have changed the position of the trend with "TrendRepos", "TrendIndex" is assigned the value of "TrendRepos".

Access in runtime Read and write

Syntax

Object.**TrendRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the position of the referenced trend in the trend window. Value range from 0 to "TrendCount" minus 1. Values outside this range have no effect.

0: The referenced trend is positioned at the front.

See also

TrendIndex (Page 1103)

TrendCount (Page 1099)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

Trends (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TrendSelectTagName (RT Professional)**Description**

Specifies that the dialog for selecting the tag name for the data source of the value axis is displayed initially in runtime.

Access in runtime: Read and write

Syntax

Object.**TrendSelectTagName**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE, if the dialog for selecting the tag name for the data source of the value axis is displayed in the screen.

FALSE, , if no dialog for selecting the tag name for the data source of the value axis is displayed in the screen.

TrendSelectTagNameX (RT Professional)

Description

Specifies that the dialog for selecting the tag name of the data source of the x-axis is displayed initially in runtime.

Access in runtime Read and write

Syntax

Object.**TrendSelectTagNameX**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional. TRUE if the dialog for selecting the tag name of the data source of the X-axis is displayed in the screen.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

TrendSelectTagNameY (RT Professional)

Description

Specifies that the dialog for selecting the tag name for the data source of the Y axis is displayed initially in runtime.

Access in runtime Read and write

Syntax

Object.**TrendSelectTagNameY**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional. TRUE if the dialog for selecting the tag name of the data source of the Y-axis is displayed in the screen.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

TrendsForPrinting (RT Advanced)**Description**

No access in runtime.

TrendTagName (RT Professional)**Description**

Specifies the name of the tag that supplies the value axis with data.

Access in runtime: Read and write

Syntax

Object.**TrendTagName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the tag for the value axis.

TrendTagNameX (RT Professional)**Description**

Specifies the name of the connected HMI tag or column for the X-axis. You select an HMI tag or a column with the selection button.

Access in runtime: Read and write

Syntax

Object.**TrendTagNameX**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the HMI tag or column for the X axis.

See also

FunctionTrendControl (Page 309)

TrendTagNameY (RT Professional)

Description

Specifies the name of the connected HMI tag or column for the Y-axis. You select an HMI tag or a column with the selection button.

Access in runtime: Read and write

Syntax

Object.**TrendTagNameY**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl

STRING

Optional A value or a constant that specifies the name of the HMI tag or column for the Y axis.

See also

FunctionTrendControl (Page 309)

TrendTimeAxis (RT Professional)

Description

Specifies the time axis to be used for the selected trend.

Access in runtime Read and write

Syntax

Object.**TrendTimeAxis**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the time axis for the selected trend.

See also

TimeAxisCount (Page 1014)

OnlineTrendControl (Page 360)

TrendTimeRangeBase (RT Professional)

Description

Specifies the time unit for calculating the time range.

Access in runtime Read and write

Syntax

Object.**TrendTimeRangeBase**[=TagLoggingTimeUnit]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

TagLoggingTimeUnit

Optional. A value or a constant that specifies the time unit for determining the time range.

Value	Description
500	500 ms
1000	1 second

Value	Description
60000	1 minute
3600000	1 hour
86400000	1 day

See also

FunctionTrendControl (Page 309)

TrendTimeRangeFactor (RT Professional)

Description

Specifies the factor for determining the time interval for which the referenced trend is displayed.

You determine the time span by multiplying the factor and time unit.

The setting is effective if "TrendRangeType (Page 1113)" property has the value "0".

Access in runtime: Read and write

Syntax

Object.**TrendTimeRangeFactor**[=Int32]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the factor for determining the time span of the referenced trend.

TrendTrendWindow (RT Professional)

Description

Defines the trend window for visualizing the trend selected.

Define the available trend windows in the "Trend window" tab.

Access in runtime: Read and write

Syntax

Object.**TrendTrendWindow**[=STRING]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional A value or constant that specifies the name of the trend window for the selected trend.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendUncertainColor (RT Professional)

Description

Specifies the color for values of uncertain status.

Access in runtime Read and write

Syntax

Object.**TrendUncertainColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the values of uncertain status in the selected trend.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendUncertainColoring (RT Professional)**Description**

Specifies that values of uncertain status are to be highlighted.

Access in runtime Read and write

Syntax

Object.**TrendUncertainColoring**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if values of uncertain status are highlighted.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendUpperLimit (RT Professional)**Description**

Specifies the high limit of a tag that is displayed in the specified object as a trend. If the tag exceeds the value of "TrendUpperLimit", the relevant trend dots are marked in the color specified with "TrendUpperLimitColor (Page 1125)".

Also specify with "TrendUpperLimitColoring (Page 1126)" that limit violation display is activated.

Access in runtime Read and write

Syntax

Object.**TrendUpperLimit**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the higher limit for values in the selected trend.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendUpperLimitColor (RT Professional)**Description**

Specifies the color that marks trend values which are above the value of "TrendUpperLimit (Page 1123)".

The setting is effective if "TrendUpperLimitColoring (Page 1126)" has the value "TRUE".

Access in runtime Read and write

Syntax

Object.**TrendUpperLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of trend values above the "TrendUpperLimit (Page 1123)" value.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). The "Red" color is displayed as follows, for example:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendUpperLimitColoring (RT Professional)**Description**

Specifies whether the selection frame is shown with the color defined by the system.

Access in runtime Read and write

Syntax

Object.**TrendUpperLimitColoring**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the "TrendUpperLimitColor" property is enabled.

FALSE if the "TrendUpperLimitColor" property is disabled.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendValueAlign (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

TrendValueAxis (RT Professional)

Description

Specifies the value axis that is used for the trend selected.

Access in runtime Read and write

Syntax

Object.**TrendValueAxis**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the value axis for the selected trend.

See also

ValueAxisCount (Page 1179)

OnlineTrendControl (Page 360)

TrendValueUnit (RT Professional)

Description

Specifies a unit of the values for the "Display values" trend type. The unit is dependent on the value to be displayed, e.g., "%" or "°C".

Access in runtime: Read and write

Syntax

Object.**TrendValueUnit**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the unit for the values of the value axis.

See also

OnlineTrendControl (Page 360)

TrendVisible (RT Professional)**Description**

Specifies whether the trend referenced with "TrendIndex" is displayed in the object.

Access in runtime Read and write

Syntax

Object.**TrendVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE, if the referenced trend is displayed.

FALSE, if the referenced trend is not displayed.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowAdd (RT Professional)**Description**

Creates a new trend view. The newly created trend view is automatically referenced with "TrendWindowIndex".

Access in runtime Read and write

Syntax

Object.**TrendWindowAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Required. A value or a constant that specifies the name of the new trend view with "TrendWindowName".

See also

TrendWindowIndex (Page 1134)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowCoarseGrid (RT Professional)**Description**

Specifies if the grid lines for the main scale are displayed.

Access in runtime Read and write

Syntax

Object.**TrendWindowCoarseGrid**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the grid lines for the main scale are displayed.

FALSE if the grid lines for the main scale are not displayed.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowCoarseGridColor (RT Professional)

Description

Specifies the color of the main grid for the referenced diagram in the object.

Access in runtime: Read and write

Syntax

Object.**TrendWindowCoarseGridColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the main grid for the referenced diagram.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

TrendWindowIndex (Page 1134)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowCount (RT Professional)

Description

Specifies the number of configured trend diagrams in the trend view.

Access in runtime Read and write

Syntax

Object.**TrendWindowCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the number of configured trend diagrams in the trend view.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowFineGrid (RT Professional)**Description**

Specifies whether grid lines are displayed for the secondary scaling.

Access in runtime Read and write

Syntax

Object.**TrendWindowFineGrid**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the grid lines for the secondary scale are displayed.

FALSE if the grid lines for the secondary scale are not displayed.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowFineGridColor (RT Professional)

Description

Specifies the color of the secondary grid for the referenced diagram in the object.

Access in runtime Read and write

Syntax

Object.**TrendWindowFineGridColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the help grid for the referenced diagram.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

TrendWindowIndex (Page 1134)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowForegroundTrendGrid (RT Professional)

Description

Specifies whether only the grid lines for the foreground trend are displayed in the selected trend window.

Access in runtime Read and write

Syntax

Object.**TrendWindowForegroundTrendGrid**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the grid lines for the foreground trend are displayed in the trend window.

FALSE if the grid lines for all trends are displayed in the trend window.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowGridInTrendColor (RT Professional)

Description

Specifies whether the grid lines for the main scaling are displayed in the trend color.

Access in runtime: Read and write

Syntax

Object.**TrendWindowGridInTrendColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the grid lines are displayed in the trend color.

FALSE if the grid lines are displayed in the color set in "TrendWindowCoarseGridColor (Page 1129)".

See also

FunctionTrendControl (Page 309)

TrendWindowHorizontalGrid (RT Professional)**Description**

Specifies whether the horizontal grid lines are shown.

Access in runtime: Read and write

Syntax

Object.**TrendWindowHorizontalGrid**[=BOOLEAN]

Object

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE: The horizontal grid lines are displayed.

FALSE: The horizontal grid lines are not displayed.

See also

FunctionTrendControl (Page 309)

TrendWindowIndex (RT Professional)**Description**

References a trend view. To access the properties of a trend view, you need to set ""TrendWindowIndex"".

Values between 0 and "TrendWindowCount" minus 1 are valid for "TrendWindowIndex". The "TrendWindowCount" property specifies the number of configured trend views.

Access in runtime Read and write

Syntax

Object.**TrendWindowIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies, with the index, the trend view to be edited.

See also

TrendWindowCount (Page 1129)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowName (RT Professional)**Description**

Specifies the name of a trend view referenced with "TrendWindowIndex".

Access in runtime Read and write

Syntax

Object.**TrendWindowName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced trend view.

See also

TrendWindowIndex (Page 1133)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowRemove (RT Professional)

Description

Removes a trend view referenced with its name.

Access in runtime Read and write

Syntax

Object.**TrendWindowRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced trend view to be removed.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowRename (RT Professional)

Description

Specifies the new name of a trend view referenced with "TrendWindowIndex".

Access in runtime Read and write

Syntax

Object.**TrendWindowRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the new name of the selected trend view.

See also

TrendWindowIndex (Page 1133)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowRepos (RT Professional)

Description

Specifies the position of a trend view referenced with "TrendWindowIndex".

If you have changed the position of the trend view with "TrendWindowRepos", "TrendWindowIndex" is assigned the value of "TrendWindowRepos".

Access in runtime Read and write

Syntax

Object.**TrendWindwRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the position of the referenced trend view. Value range from 0 to "TrendWindowCount" minus 1. Values outside this range have no effect.

0: The referenced trend view is positioned at the front.

See also

TrendWindowIndex (Page 1133)

TrendWindowCount (Page 1129)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowRulerColor (RT Professional)

Description

Specifies the color of the ruler.

The setting is effective if the "TrendWindowRulerStyle (Page 1139)" property has the value "1".

Access in runtime: Read and write

Syntax

Object.**TrendWindowRulerColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the ruler.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowRulerLayer (RT Professional)

Description

Specifies the display layer of the ruler in the trend view.

Access in runtime Read and write

Syntax

Object.**TrendWindowRulerLayer**[=RulerLayer]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

RulerLayer

Optional. A value or a constant that specifies the display layer of the ruler in the trend view.

Value	Explanation
0	The ruler is visualized on a layer under the grid.
1	The ruler is positioned on top of the trend and under the grid.
2	The ruler is positioned on top of the trend.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowRulerStyle (RT Professional)**Description**

Specifies the appearance of the ruler.

Access in runtime Read and write

Syntax

Object.**TrendWindowRulerStyle**[=RulerStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

RulerStyle

Optional. A value or a constant that specifies how the ruler is displayed.

Value	Explanation
0	The ruler is displayed as a single black line.
1	The ruler is displayed in the configured color and line width.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowRulerWidth (RT Professional)

Description

Specifies the thickness of the ruler in pixels.

The setting is effective if the "TrendWindowRulerStyle (Page 1138)" property has the value "1".

Access in runtime: Read and write

Syntax

Object.**TrendWindowRulerWidth**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the width of the ruler in pixels.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindows (RT Professional)

Description

No access in runtime.

TrendWindowSpacePortion (RT Professional)

Description

Specifies the range proportion of the referenced diagram in the diagram area of the object.

Access in runtime: Read and write

Syntax

Object.**TrendWindowSpacePortion**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

Int32

Optional. A value or constant that specifies the range proportion of the referenced diagram in the diagram area of the object.

Comments

The amount of each range proportion can be calculated from the total number of range components. If, for example, you have configured a total of three diagrams, a range proportion of "1" each will result in three diagrams of equal size. To increase the size of range components in relation to each other, increase the range proportion of one or more diagrams.

See also

TrendWindowIndex (Page 1133)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowStatisticRulerColor (RT Professional)**Description**

Specifies the color of the statistics ruler. You specify additionally using the "TrendWindowStatisticRulerStyle (Page 1142)" property that the statistics ruler can be configured graphically.

Access in runtime: Read and write

Syntax

Object.**TrendWindowStatisticRulerColor**[=Color]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the statistics ruler.

Comments

You use the "RGB" function to define the color in RGB format (red, green, blue). To do this, you specify the corresponding decimal value for each of the three RGB values (value range from 0 to 255). The "Red" color is displayed as follows, for example:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTrendControl (Page 360)

TrendWindowStatisticRulerStyle (RT Professional)

Description

Defines the graphical configurability of the lines for specifying the statistics area in the trend window.

Access in runtime Read and write

Syntax

Object.**TrendWindowStatisticRulerStyle**[=RulerStyle]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

RulerStyle

Optional. A value or a constant that specifies whether the statistics ruler can be configured graphically.

The following settings are available:

Value	Designation	Description
0	Simple	The ruler is displayed as a simple black line.
1	Graphic	The ruler is displayed in the configured color and line width.

See also

OnlineTrendControl (Page 360)

TrendWindowStatisticRulerWidth (RT Professional)

Description

Specifies the line width of the statistics ruler. You specify additionally using the "TrendWindowStatisticRulerStyle (Page 1141)" property that the statistics ruler can be configured graphically.

Access in runtime: Read and write

Syntax

Object.**TrendWindowStatisticRulerWidth**[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the line width of the statistics ruler.

See also

OnlineTrendControl (Page 360)

TrendWindowVerticalGrid (RT Professional)

Description

Specifies whether the vertical grid lines are displayed.

Access in runtime Read and write

Syntax

Object.**TrendWindowVerticalGrid**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the vertical grid lines are displayed.

FALSE if the vertical grid lines are not displayed.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendWindowVisible (RT Professional)**Description**

Specifies whether the trend view referenced with "TrendWindowIndex" is displayed.

Access in runtime: Read and write

Syntax

Object.**TrendWindowVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the referenced trend view is displayed.

FALSE if the referenced trend view is not displayed.

See also

TrendWindowIndex (Page 1133)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendXAxis (RT Professional)**Description**

Defines the X axis to be used for the referenced trend.

Access in Runtime: Read and write

Syntax

Object.**TrendXAxis**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or constant that specifies the name of the X-axis used for the referenced trend.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

TrendYAxis (RT Professional)

Description

Specifies which Y axis is to be used for the referenced trend.

Access in Runtime: Read and write

Syntax

Object.**TrendYAxis**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the name of the Y-axis used for the referenced trend.

See also

TrendIndex (Page 1103)

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

1.5.5.14 Properties U-W (Panels, Comfort Panels, RT Advanced, RT Professional)

Unit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the unit of measurement.

Access in runtime: Read and write

Syntax

Object.**Unit**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar
- IOField

STRING

Optional. A value or a constant that specifies the unit of measurement.

See also

Bar (Page 258)

IOField (Page 332)

UnitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the text color for the unit of measurement.

Access in runtime Read and write

Syntax

Object.**UnitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the text color for the measurement unit.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

UnitFont (RT Professional)

Description

Specifies the font of the unit of measurement.

Access in runtime: Read and write

Syntax

Object.**UnitFont**[=Font]

Object

Required. A "ScreenItem" object with the following characteristics:

- Gauge

Font

Optional. A value or a constant that specifies the font.

See also

Gauge (Page 320)

UnitText (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the text for the unit of measurement.

Access in runtime Read and write

Syntax

Object.**UnitText**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

STRING

Optional. A value or a constant that specifies the text for the measurement unit.

Comments

Enter a text to show, for example, the physical unit of the value displayed.

See also

Gauge (Page 320)

UnitTop (RT Professional)**Description**

Specifies the distance from the unit of measurement to the upper edge of the object. The lettering can only be positioned along the vertical diameter of the scale. The value of the property refers to the height of the object and is measured from the upper edge of the object to the lower edge of the lettering.

Access in runtime Read and write

Syntax

Object.**UnitTop**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

DOUBLE

Optional. A value or a constant that specifies the distance between the measurement unit and the upper edge of the object.

Value	Description
0	The lower edge of the lettering is positioned on the upper limit of the selected object. The text is no longer visible as it is positioned outside the object.
1	The lower edge of the lettering is positioned on the lower limit.

See also

Gauge (Page 320)

UnitViewColumnOrder (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UpdateButtonVisible (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UpperLimit (RT Professional)

Description

Specifies the upper limit for input values.

Access in runtime: Read and write

Syntax

Object.**UpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- IOField

DOUBLE

Optional A value or a constant that specifies the upper limit for input values.

See also

IOField (Page 332)

UseAutoScaling (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UseButtonFirstGradient (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UseButtonSecondGradient (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UseColumnBackColor (RT Professional)**Description**

Specifies the settings for the background color of the columns.

Access in runtime Read and write

Syntax

Object.**UseColumnBackColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the background color configured in "TableColor (Page 991)" is used.

FALSE if the background color configured for the columns in the table view is used.

See also

OnlineTableControl (Page 348)

UseColumnForeColor (RT Professional)**Description**

Specifies the settings for the font color of the columns.

Access in runtime Read and write

Syntax

Object.**UseColumnForeColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the font color specified with "TableForeColor (Page 993)" is used.

FALSE, if the font color specified for the columns in the table view is used.

See also

OnlineTableControl (Page 348)

UseCurserKeyScroll (RT Advanced)**Description**

No access in runtime.

See also

OnlineTableControl (Page 348)

Used (RT Professional)**Description**

Returns the size of the used disk space.

Access in runtime: Read

Syntax

Object.**Used**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

DOUBLE

Optional A value or a constant that returns the size of the used disk space.

See also

DiskSpaceView (Page 298)

UseDesignColorSchema (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the colors defined in the global color scheme of the current design are to be used for this object.

Access in runtime:

- RT Advanced: No access
- RT Professional: Read and write

Syntax

Object.**UseDesignColorSchema**[=BOOLEAN]

Object

Required. An object of the "ScreenItem" type with the following format:

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup

- Polygon
- Polyline
- Rectangle
- RoundButton*
- Slider*
- SymbolicIOField*
- TextField
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowSlider*

*: Read access only

You have no access in runtime with the following format:

- AlarmView
- DateTimeField
- RecipeView
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

BOOLEAN

Optional.

TRUE if the object is displayed with the colors from the global color scheme defined for this object type.

FALSE if the object is displayed with the colors as per the settings in the object.

UseDesignShadowSettings (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the object is displayed with global shading. The global shading is specified in the current design.

Access in runtime: Read

Syntax

Object.**UseDesignShadowSettings**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- Bar
- Button
- CheckBox
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polyline
- RoundButton
- Slider
- SymbolicIOField
- TextField
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowSlider

You have no access in runtime with the following format:

- CircleSegment

BOOLEAN

Optional.

TRUE if the object is displayed with the global shading specified for this object type.

FALSE if the object is displayed without shadowing.

See also

Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
GraphicView (Page 327)
Line (Page 337)
Polygon (Page 379)
Polyline (Page 382)
Rectangle (Page 396)
TextField (Page 440)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)
Bar (Page 258)
Button (Page 266)
CheckBox (Page 275)
Clock (Page 287)
ComboBox (Page 290)
Gauge (Page 320)
GraphicIOField (Page 324)
IOField (Page 332)
Listbox (Page 340)
MultiLineEdit (Page 345)
OptionGroup (Page 372)
RoundButton (Page 399)
Slider (Page 410)

SymbolicIOField (Page 424)

WindowSlider (Page 485)

UsedPercent (RT Professional)

Description

Specifies the measured value for the used disk space as a percentage. The values can be queried in Runtime. The values cannot be predefined.

Access in runtime: Read

Syntax

Object.**UsedPercent**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- DiscSpaceView

Int32

Optional A value or a constant that returns the measured values for the used disk space as a percentage.

See also

DiskSpaceView (Page 298)

UseExponentialFormat (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the figures are shown exponentially (e.g. "1.00e+000").

Access in runtime: Read and write

Syntax

Object.**UseExponentialFormat**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the figures are shown exponentially (e.g. "1.00e+000").

See also

Bar (Page 258)

UseFirstGradient (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UseFlashTransparentColor (RT Professional)

Description

Specifies whether the color of the bitmap object of a flashing graphic will be set to "transparent".

Access in runtime: Read and write

Syntax

Object.**UseFlashTransparentColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- "GraphicIOField"

BOOLEAN

Optional. TRUE, if the color of the bitmap object of a flashing graphic will be set to "transparent".

See also

GraphicIOField (Page 324)

UseMessageColor (RT Professional)

Description

Specifies whether the agreed colors of the message classes are displayed.

Access in runtime Read and write

Syntax

Object.**UseMessageColor** [=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl

BOOLEAN

Optional.

TRUE if the colors are displayed.

FALSE if the color settings specified for the table content on the "Layout" tab are effective.

See also

AlarmControl (Page 236)

UserData (Professional) (RT Professional)**Description**

Specifies the value that is passed to the VB script when a user-defined menu entry or toolbar button is executed.

Use the "Data" field in the "Menus and toolbars" editor to pass a parameter to the procedure.

Access in Runtime: Read and write

Syntax

Object.**UserData**[=String]

Object

Required A "Item"-type object.

String

Optional. A value or constant that is passed to the VB script when a user-defined menu entry or toolbar button is executed.

Example

The following example shows the script "ChangeScreen" that is executed when you click a user-defined menu item. You transfer the screen name to the script with the "Data" field.

```
Sub ChangeScreen (ByVal Item)
    Dim objScreen, strScreenName
    ' "UserData" contains the screen name specified
    ' in editor menus and toolbars.
    strScreenName = Item.Userdata
    HMIRuntime.BaseScreenName = strScreenName
End Sub
```

UserName (RT Professional)

Description

Returns the name of the user who triggered the alarm object.

UseScadaRendererStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UseSecondGradient (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UseSelectedTitleColor (RT Professional)

Description

Specifies whether a selection color is used for the headers of selected table cells.

Access in runtime Read and write

Syntax

Object.**UseSelectedTitleColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if a highlight color is used. The "Background" and "Font" settings are active in Runtime.

FALSE if a highlight color is not used. The "Background" and "Font" settings are disabled in Runtime.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

UseSourceBackColors (RT Professional)**Description**

Specifies that the background color of the data source is used.

Access in runtime: Read and write

Syntax

Object.**UseSourceBackColors**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the background color of the connected trend or table view is used.

FALSE, if the background color configured in the value table is used.

See also

TrendRulerControl (Page 444)

UseSourceForeColors (RT Professional)

Description

Specifies that the font color of the data source is used.

Access in runtime: Read and write

Syntax

Object.**UseSourceForeColors**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following characteristics:

- TrendRulerControl

BOOLEAN

Optional.

TRUE, if the font color of the connected trend or table view is used.

FALSE, if the font color configured in the value table is used.

See also

TrendRulerControl (Page 444)

UseSystemScrollbarWidth (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UseTableColor2 (RT Professional)

Description

Specifies whether a second row color is used for the representation of the table.

Access in runtime Read and write

Syntax

Object.**UseTableColor2**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if the background colors from "TableColor (Page 991)" and "TableColor2 (Page 991)" are used alternately.

FALSE if the background colors from "TableColor (Page 991)" are used for all rows.

See also

TableColor2 (Page 991)

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

UseTableHeaderFirstGradient (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UseTableHeaderSecondGradient (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UseTagLimitColors (RT Professional)

Description

Specifies whether the violation of the configured high and low limits for the object are highlighted in color.

Access in runtime Read and write

Syntax

Object.**UseTagLimitColors**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- IOField

BOOLEAN

Optional. TRUE if the limits are highlighted in color.

See also

IOField (Page 332)

UseTransparentColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the color defined with the "TransparentColor" property is shown as transparent.

Access in runtime: Read and write

Syntax

Object.**UseTransparentColor**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- GraphicIOField
- GraphicView

BOOLEAN

Optional. TRUE if the specified color is to be shown as transparent.

See also

GraphicIOField (Page 324)

GraphicView (Page 327)

UseTransparentColorDeactivatedPicture (RT Professional)**Description**

Specifies if the transparent color defined with the property "TransparentColorDeactivatedPicture" for the status "disabled" will be used.

Access in runtime: Read

Syntax

Object.**UseTransparentColorDeactivatedPicture**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- RoundButton

BOOLEAN

Optional. TRUE, if the transparent color defined with the property "TransparentColorDeactivatedPicture" for the status "disabled" is used.

See also

RoundButton (Page 399)

UseTransparentColorPictureOff (RT Professional)**Description**

Specifies if the transparent color defined with the property "TransparentColorPictureOff" for the "off" status will be used.

Access in runtime: Read and write

Syntax

Object.**UseTransparentColorPictureOff**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton*

*: Read access only

BOOLEAN

Optional. TRUE, if the transparent color defined with the property "TransparentColorPictureOff" for the "off" status is used.

UseTransparentColorPictureOn (RT Professional)**Description**

Specifies if the transparent color defined with the property "TransparentColorPictureOn" for the "on" status will be used.

Access in runtime: Read and write

Syntax

Object.UseTransparentColorPictureOn[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Button
- RoundButton*

*: Read access only

BOOLEAN

Optional. TRUE, if the transparent color defined with the property "TransparentColorPictureOn" for the "on" status is used.

UseTrendNameAsLabel (RT Professional)**Description**

Specifies whether the "Name" or "Label" property is used as a designation for the trend in runtime.

Access in runtime Read and write

Syntax

Object.UseTrendNameAsLabel[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the name configured under "Properties > Properties > Trends > Name" is used.

FALSE if the name configured under "Properties > Properties > Trends > Label" is used.

See also

FunctionTrendControl (Page 309)

OnlineTrendControl (Page 360)

UseTwoHandOperation (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UseUdp (RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_AKZ (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_Descriptor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_InstallationDate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_LADDR (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_Name (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_OKZ (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_OperationState (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_OrderID (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_ProfileID (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_Rack (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_Slot (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_SoftwareRevision (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_SpecificProfileData (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_State (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_Station (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_SubAddress (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ColumnWidth_SubSlot (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_SubSystem (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ColumnWidth_Type (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_AKZ (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_Descriptor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_InstallationDate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_LADDR (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_Name (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ShowItem_OKZ (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ShowItem_OperationState (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ShowItem_OrderID (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ShowItem_ProfileID (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ShowItem_Rack (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ShowItem_Slot (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

No access in runtime.

UV_ShowItem_SoftwareRevision (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_SpecificProfileData (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_State (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_Station (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_SubAddress (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_SubSlot (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_SubSystem (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

UV_ShowItem_Type (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

Value (RT Professional)

Description

Specifies a value for the object being used or returns it.

Access during runtime: Read and write

Syntax

Object.**Value**[=VARIANT]

Object

Necessary. An object of type "Tag", "DataItem" or "ScreenItem" with the format "Gauge".

VARIANT

Optional The value that is specified dependent on the object being used:

- Tag.Value: returns the tag value for the last read access or specifies the future tag value. Use the "Read" method to read the tag value from the "Value" property. You assign a new tag value to the "Value" property with the "Write" method.
- Dataset.Value: Specifies a value or returns a copy of the value or the object reference. When returning object references, ensure that the object reference is multithread-capable.
- ScreenItem("Gauge_1").Value: Specifies the value to which the pointer points. The value range within the values set via the properties "ValueMin", and "ValueMax".

Examples

The following example writes a new value in the "Tag1" tag:

```
'VBS94
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Value = 50
objTag.Write
```

The example shows how to add a value to a list of tags and output it as a trace. After that, the value is changed, output again and then removed. It make sense to perform this in several different actions:

```
'VBS198
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet("motor1").Value = 55
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")
```

See also

[DataItem \(Page 203\)](#)

[Tag \(Page 228\)](#)

[Gauge \(Page 320\)](#)

ValueAxes (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ValueAxis1AutoRange (RT Advanced)

Description

No access in runtime.

ValueAxis1Begin (RT Advanced)

Description

No access in runtime.

ValueAxis1End (RT Advanced)

Description

No access in runtime.

ValueAxis1LabelLength (RT Advanced)**Description**

No access in runtime.

ValueAxis1Style (RT Advanced)**Description**

No access in runtime.

ValueAxis2AutoRange (RT Advanced)**Description**

No access in runtime.

ValueAxis2Begin (RT Advanced)**Description**

No access in runtime.

ValueAxis2End (RT Advanced)**Description**

No access in runtime.

ValueAxis2LabelLength (RT Advanced)**Description**

No access in runtime.

ValueAxis2Style (RT Advanced)**Description**

No access in runtime.

ValueAxisAdd (RT Professional)

Description

Creates a new value axis. The newly created value axis is automatically referenced with "ValueAxisIndex".

Access in runtime Read and write

Syntax

Object.**ValueAxisAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Required. A value or a constant that specifies the name of the new value axis with "ValueAxisName".

See also

ValueAxisIndex (Page 1180)

OnlineTrendControl (Page 360)

ValueAxisAlignment (RT Professional)

Description

Specifies the alignment of the value axis referenced with "ValueAxisIndex".

Access in runtime Read and write

Syntax

Object.**ValueAxisAlignment**[=AxisAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

AxisAlignment

Optional. A value or a constant that specifies the alignment of the value axis.

Value	Designation	Description
0	Left	The value axis selected is displayed to the left of the trend
1	Right	The value axis selected is displayed to the right of the trend.

See also

OnlineTrendControl (Page 360)

ValueAxisAutoPrecisions (RT Professional)**Description**

Specifies whether the number of decimal places for a value axis referenced with "ValueAxisIndex" is set automatically.

Access in runtime Read and write

Syntax

Object.**ValueAxisAutoPrecisions**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the number of decimal places is automatically determined.

FALSE if the value of "ValueAxisPrecisions" is used.

See also

ValueAxisPrecisions (Page 1183)

ValueAxisIndex (Page 1180)

OnlineTrendControl (Page 360)

ValueAxisAutoRange (RT Professional)

Description

Specifies whether the value range of a value axis referenced with "ValueAxisIndex" is calculated automatically.

Access in runtime Read and write

Syntax

Object.**ValueAxisAutoRange**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the value range is calculated automatically.

FALSE if the value range is specified with "ValueAxisBeginValue" and "ValueAxisEndValue".

See also

ValueAxisBeginValue (Page 1177)

ValueAxisEndValue (Page 1179)

ValueAxisIndex (Page 1180)

OnlineTrendControl (Page 360)

ValueAxisBeginValue (RT Professional)

Description

Specifies the lower end of the value range of a value axis referenced with "ValueAxisIndex".

Also set "ValueAxisAutoRange" to "FALSE".

Access in runtime Read and write

Syntax

Object.**ValueAxisBeginValue**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the low limit of the value range of the referenced value axis.

See also

ValueAxisIndex (Page 1180)

ValueAxisAutoRange (Page 1176)

OnlineTrendControl (Page 360)

ValueAxisColor (RT Professional)**Description**

Specifies the color of a value axis referenced with "ValueAxisIndex". This property is only evaluated if the "ValueAxisInTrendColor (Page 1181)" property is set to "FALSE".

Access in runtime Read and write

Syntax

Object.**ValueAxisColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Color

Optional. A value or a constant that specifies the color of the value axis.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTrendControl (Page 360)

ValueAxisCount (RT Professional)

Description

Specifies the number of configured value axes.

Access in runtime Read and write

Syntax

Object.ValueAxisCount[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the number of configured value axes.

See also

OnlineTrendControl (Page 360)

ValueAxisEndValue (RT Professional)

Description

Specifies the upper end of the value range of a value axis referenced with "ValueAxisIndex".

Also set "ValueAxisAutoRange" to "FALSE".

Access in runtime Read and write

Syntax

Object.ValueAxisEndValue[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

DOUBLE

Optional. A value or a constant that specifies the high limit of the value range of the referenced value axis.

See also

ValueAxisIndex (Page 1180)

ValueAxisAutoRange (Page 1176)

OnlineTrendControl (Page 360)

ValueAxisExponentialFormat (RT Professional)**Description**

Specifies whether the values of a value axis referenced with "ValueAxisIndex" are displayed in exponential notation.

Access in runtime Read and write

Syntax

Object.**ValueAxisExponentialFormat**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the values are displayed in exponential notation.

FALSE if the values are not displayed in exponential notation.

See also

ValueAxisIndex (Page 1180)

OnlineTrendControl (Page 360)

ValueAxisIndex (RT Professional)**Description**

References a value axis. To access the properties of a value axis, you need to set ""ValueAxisIndex"".

Values between 0 and "ValueAxisCount" minus 1 are valid for "ValueAxisIndex". The "ValueAxisCount" property specifies the number of configured value axes.

Access in runtime Read and write

Syntax

Object.ValueAxisIndex[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies, with the index, the value axis to be edited.

See also

ValueAxisCount (Page 1178)

OnlineTrendControl (Page 360)

ValueAxisInTrendColor (RT Professional)**Description**

Specifies whether the color of the axis referenced with "ValueAxisIndex" corresponds to the trend color.

Access in runtime Read and write

Syntax

Object.ValueAxisInTrendColor[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the referenced axis is shown in the trend color. The value of "ValueAxisColor" is not effective.

FALSE if the referenced axis is shown in the color specified in "ValueAxisColor".

See also

ValueAxisIndex (Page 1179)

OnlineTrendControl (Page 360)

ValueAxisLabel (RT Professional)

Description

Defines the label text for a value axis referenced with "ValueAxisIndex".

Access in runtime Read and write

Syntax

Object.ValueAxisLabel[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the label text for the referenced value axis.

See also

ValueAxisIndex (Page 1179)

OnlineTrendControl (Page 360)

ValueAxisName (RT Professional)

Description

Specifies the name of a value axis referenced with "ValueAxisIndex".

Access in runtime Read and write

Syntax

Object.ValueAxisName[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced value axis.

See also

ValueAxisIndex (Page 1179)

OnlineTrendControl (Page 360)

ValueAxisPrecisions (RT Professional)

Description

Specifies the number of decimal places for the label of the value axis referenced with "ValueAxisIndex".

Access in runtime Read and write

Syntax

Object.ValueAxisPrecisions[=Int16]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int16

Optional. A value or a constant that specifies the number of decimal places.

See also

OnlineTrendControl (Page 360)

ValueAxisRemove (RT Professional)

Description

Removes a value axis referenced with its name.

Access in runtime Read and write

Syntax

Object.ValueAxisRemove[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced value axis to be removed.

See also

OnlineTrendControl (Page 360)

ValueAxisRename (RT Professional)**Description**

Specifies the new name of a value axis referenced with "ValueAxisIndex".

Access in runtime Read and write

Syntax

Object.**ValueAxisRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the new name of the value axis.

See also

ValueAxisIndex (Page 1179)

OnlineTrendControl (Page 360)

ValueAxisRepos (RT Professional)**Description**

Specifies the position of a value axis referenced with "ValueAxisIndex".

If you have changed the position of the value axis with "ValueAxisRepos", "ValueAxisIndex" is assigned the value of "ValueAxisRepos".

Access in runtime Read and write

Syntax

Object.**ValueAxisRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

Int32

Optional. A value or a constant that specifies the position of the referenced value axis. Value range from 0 to "ValueAxisCount" minus 1. Values outside this range have no effect.

0: The referenced value axis is positioned outside.

See also

ValueAxisIndex (Page 1179)

ValueAxisCount (Page 1178)

OnlineTrendControl (Page 360)

ValueAxisScalingType (RT Professional)**Description**

Specifies the type of scaling for the value axis referenced with "ValueAxisIndex".

Access in runtime Read and write

Syntax

Object.ValueAxisScalingType[=AxisScalingType]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

AxisScalingType

Optional. A value or a constant that specifies the type of scaling for the referenced value axis.

Value	VB constant	Designation
0	hmiBarScalingLinear	Linear
1	hmiBarScalingLogarithmic	Logarithmic
2	hmiBarScalingNegativeLogarithmic	Negative logarithmic

See also

ValueAxisIndex (Page 1179)

OnlineTrendControl (Page 360)

ValueAxisTrendWindow (RT Professional)

Description

Specifies the trend diagram in which the axis referenced with "ValueAxisIndex" is displayed.

Access in runtime Read and write

Syntax

Object.**ValueAxisTrendWindow**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

STRING

Optional. A value or a constant that specifies the name of the trend diagram.

See also

TrendWindowCount (Page 1129)

ValueAxisIndex (Page 1179)

OnlineTrendControl (Page 360)

ValueAxisVisible (RT Professional)

Description

Specifies whether the value axis referenced with "ValueAxisIndex" is displayed in the object.

Access in runtime Read and write

Syntax

Object.**ValueAxisVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTrendControl

BOOLEAN

Optional.

TRUE if the referenced value axis is displayed.

FALSE if the referenced value axis is not displayed.

See also

ValueAxisIndex (Page 1179)

OnlineTrendControl (Page 360)

ValueCaption (RT Advanced)

Description

No access in runtime.

ValueColumnAdd (RT Professional)

Description

Creates a new value column. The newly created value column is automatically referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.**ValueColumnAdd**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Required. A value or a constant that specifies the name of the new value column with "ValueColumnName".

See also

ValueColumnIndex (Page 1193)

OnlineTableControl (Page 348)

ValueColumnAlignment (RT Professional)

Description

Specifies how the text in the value column referenced with "ValueColumnIndex" is aligned.

Access in runtime Read and write

Syntax

Object.ValueColumnAlignment[=HorizontalAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

HorizontalAlignment

Optional. A value or a constant that specifies how the text in the referenced value column is aligned.

Value	VB constant	Explanation
0	hmiAlignmentLeft	The text is left-justified.
1	hmiAlignmentCentered	The text is centered.
2	hmiAlignmentRight	The text is right-justified.

See also

OnlineTableControl (Page 348)

ValueColumnAutoPrecisions (RT Professional)

Description

Specifies whether the number of decimal places for a value axis referenced with "ValueColumnIndex" is set automatically.

Access in runtime Read and write

Syntax

Object.ValueColumnAutoPrecisions[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the number of decimal places is automatically determined.

FALSE if the value of "ValueColumnPrecisions" is used.

See also

ValueColumnPrecisions (Page 1194)

ValueColumnIndex (Page 1193)

OnlineTableControl (Page 348)

ValueColumnBackColor (RT Professional)

Description

Specifies the background color of the value column referenced with "ValueColumnIndex".

Also set "UseColumnBackColor (Page 1149)" to "TRUE".

Access in runtime Read and write

Syntax

Object.**ValueColumnBackColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Color

Optional. A value or a constant that specifies the background color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTableControl (Page 348)

ValueColumnCaption (RT Professional)

Description

Specifies the name of the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.**ValueColumnCaption**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the name of the value column.

See also

OnlineTableControl (Page 348)

ValueColumnCount (RT Professional)

Description

Specifies the number of configured value columns.

Access in runtime Read and write

Syntax

Object.**ValueColumnCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies the number of configured value columns.

See also

OnlineTableControl (Page 348)

ValueColumnExponentialFormat (RT Professional)

Description

Specifies whether the values of a value column referenced with "ValueColumnIndex" are displayed in exponential notation.

Access in runtime Read and write

Syntax

Object.ValueColumnExponentialFormat[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the values are displayed in exponential notation.

FALSE if the values are not displayed in exponential notation.

See also

ValueColumnIndex (Page 1193)

OnlineTableControl (Page 348)

ValueColumnForeColor (RT Professional)

Description

Specifies the font color of the value column referenced with "ValueColumnIndex".

Also set "UseColumnForeColor (Page 1149)" to "TRUE".

Access in runtime Read and write

Syntax

Object.ValueColumnForeColor[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Color

Optional. A value or a constant which specifies the font color.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

OnlineTableControl (Page 348)

ValueColumnHideText (RT Professional)**Description**

Specifies whether the text in the value column referenced with "ValueColumnIndex" is hidden.
Access in runtime Read and write

Syntax

Object.**ValueColumnHideText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE, if the text in the value column is hidden.

FALSE, if the text in the value column is displayed.

See also

OnlineTableControl (Page 348)

ValueColumnHideTitleText (RT Professional)**Description**

Specifies whether the text in the header of the value column referenced with "ValueColumnIndex" is hidden.

Access in runtime Read and write

Syntax

Object.**ValueColumnHideTitleText**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE, if the text in the header of the value column is hidden.

FALSE, if the text in the header of the value column is displayed.

See also

OnlineTableControl (Page 348)

ValueColumnIndex (RT Professional)

Description

References a value column. To access the properties of a value column, you need to set ""ValueColumnIndex"".

Values between 0 and "ValueColumnCount" minus 1 are valid for "ValueColumnIndex". The "ValueColumnCount" property specifies the number of configured value columns.

Access in runtime Read and write

Syntax

Object.**ValueColumnIndex**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies, with the index, the value column to be edited.

See also

ValueColumnCount (Page 1189)

OnlineTableControl (Page 348)

ValueColumnLength (RT Professional)

Description

Specifies the width in characters of the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.**ValueColumnLength**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies the number of characters displayed.

See also

OnlineTableControl (Page 348)

ValueColumnName (RT Professional)

Description

Specifies the name of a value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.**ValueColumnName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the name of the referenced value column.

See also

ValueColumnIndex (Page 1192)

OnlineTableControl (Page 348)

ValueColumnPrecisions (RT Professional)

Description

Specifies the number of decimal places for values in the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.ValueColumnPrecisions[=Int16]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Int16

Optional. A value or constant that specifies the number of decimal places.

See also

OnlineTableControl (Page 348)

ValueColumnProvider (RT Professional)

Description

Specifies the data source for the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.ValueColumnProvider[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the data supply of a value column.

Value	Description
Logging tags	Data source with archive tags of a process value archive.
Tags	Data supply with online tags from tag management.

See also

OnlineTableControl (Page 348)

ValueCollectionProviderCLSID (RT Professional)**Description**

Specifies the provider CLSID of the data for a value column referenced with "ValueCollectionIndex".

Access in runtime Read and write

Syntax

Object.**ValueCollectionProviderCLSID**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the provider CLSID.

Value	Description
{416A09D2-8B5A-11D2-8B81-006097A45D48}	Data source with archive tags of a process value archive.
{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}	Data supply with online tags from tag management.

See also

OnlineTableControl (Page 348)

ValueCollectionRemove (RT Professional)**Description**

Removes a value column referenced with its name.

Access in runtime Read and write

Syntax

Object.**ValueCollectionRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the name of the referenced value column to be removed.

See also

OnlineTableControl (Page 348)

ValueColumnRename (RT Professional)

Description

Specifies the new name of a value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.**ValueColumnRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the new name of the value column.

See also

ValueColumnIndex (Page 1192)

OnlineTableControl (Page 348)

ValueColumnRepos (RT Professional)

Description

Specifies the position of a value column referenced with "ValueColumnIndex".

If you have changed the position of the value column with "ValueColumnRepos", "ValueColumnIndex" is assigned the value of "ValueColumnRepos".

Access in runtime Read and write

Syntax

Object.**ValueColumnRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies the position of the referenced value column. Value range from 0 to "ValueColumnCount" minus 1. Values outside this range have no effect.

0: The referenced value column is positioned on the left.

See also

ValueColumnIndex (Page 1192)

ValueColumnCount (Page 1189)

OnlineTableControl (Page 348)

ValueColumns (RT Professional)**Description**

No access in runtime.

ValueColumnSelectTagName (RT Professional)**Description**

Specifies that the dialog for selecting the tag name of the data source for the value column referenced with "ValueColumnIndex" is displayed in Runtime initial.

Access in runtime Read and write

Syntax

Object.ValueColumnSelectTagName[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE, if the dialog for selecting the tag name for the data source of the value column is displayed in the screen.

FALSE, if no dialog for selecting the tag name for the data source of the value column is displayed in the screen.

See also

OnlineTableControl (Page 348)

ValueColumnShowIcon (RT Professional)

Description

Specifies whether an icon is displayed in the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.ValueColumnShowIcon[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE, if an icon is shown.

FALSE, if no icon is shown.

See also

OnlineTableControl (Page 348)

ValueColumnShowTitleIcon (RT Professional)

Description

Specifies whether an icon is displayed in the header of the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.ValueColumnShowTitleIcon[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE, if an icon is shown in the title of the current column.

FALSE, if no icon is shown in the title of the current column.

See also

OnlineTableControl (Page 348)

ValueColumnSort (RT Professional)**Description**

Specifies the type of sorting for the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.**ValueColumnSort**[=SortMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

SortMode

Optional. A value or a constant that specifies the type of sorting in the current column.

Value	Designation	Description
0	No sorting	The values in the column are not sorted.
1	Ascending order	The values in the value column are sorted from lowest to highest value, starting from the top row.
2	Descending order	The values in the value column are sorted from highest to the lowest value, starting from the lowest row.

See also

OnlineTableControl (Page 348)

ValueColumnSortIndex (RT Professional)**Description**

Specifies the sorting order.

Access in runtime: Read and write

Syntax

Object.ValueColumnSortIndex[=Int32]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTableControl

Int32

Optional. A value or a constant that specifies the order when sorting.

See also

OnlineTableControl (Page 348)

ValueColumnTagName (RT Professional)

Description

Specifies the names of the tags whose values are displayed in the value column referenced with "ValueColumnIndex".

Access in runtime Read and write

Syntax

Object.ValueColumnTagName[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the name of the tag whose values are displayed in the current column.

See also

OnlineTableControl (Page 348)

ValueColumnTimeColumn (RT Professional)

Description

Specifies the associated time column.

Access in runtime: Read and write

Syntax

Object.**ValueColumnTimeColumn**[=STRING]

Object

Required. A "ScreenItem" object with the following characteristics:

- OnlineTableControl

STRING

Optional. A value or a constant that specifies the associated time column.

See also

OnlineTableControl (Page 348)

ValueColumnVisible (RT Professional)

Description

Specifies whether the value column referenced with "ValueColumnIndex" is displayed in the object.

Access in runtime Read and write

Syntax

Object.**ValueColumnVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- OnlineTableControl

BOOLEAN

Optional.

TRUE if the referenced value column is displayed.

FALSE if the referenced value column is not displayed.

See also

ValueColumnIndex (Page 1192)

OnlineTableControl (Page 348)

ValueColumnWidth (RT Advanced)

Description

No access in runtime.

ValueTableHeight (RT Advanced)

Description

No access in runtime.

ValueTableLeft (RT Advanced)

Description

No access in runtime.

ValueTableTop (RT Advanced)

Description

No access in runtime.

ValueTableWidth (RT Advanced)

Description

No access in runtime.

ValueY1HlpLine (RT Advanced)

Description

No access in runtime.

ValueY2HlpLine (RT Advanced)

Description

No access in runtime.

VerticalAlignment (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the vertical alignment of the text in the specified object.

Access in runtime: Read and write

Syntax

Object.**VerticalAlignment**[=VerticalAlignment]

Object

Required. An object of the type "ScreenItem" with the format:

- Button
- CheckBox
- DateTimeField
- IOField
- OptionGroup
- RoundButton *
- Switch *
- SymbolicIOField **
- TextField

* Read access only

** RT Professional, read access only

VerticalAlignment

Optional. A value that specifies the vertical alignment.

Value	VB constant	Explanation
0	hmiAlignmentTop	The text is shown at the top.
1	hmiAlignmentMiddle	The text is shown in the middle.
2	hmiAlignmentBottom	The text is shown at the bottom.

See also

Button (Page 266)

DateTimeField (Page 296)

IOField (Page 332)

OptionGroup (Page 372)

RoundButton (Page 399)

Switch (Page 420)

SymbolicIOField (Page 424)

TextField (Page 440)

CheckBox (Page 275)

VerticalGridLines (RT Professional)

Description

Specifies whether vertical separation lines are displayed.

Access in runtime Read and write

Syntax

Object.**VerticalGridLines**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

Optional.

TRUE if vertical separation lines are displayed.

FALSE if vertical separation lines are not displayed.

See also

AlarmControl (Page 236)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

VerticalPictureAlignment (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

VerticalScrollBarEnabled (RT Advanced)

Description

No access in runtime.

VerticalScrollbarPosition (RT Professional)

Description

Specifies the position of the slider in the object's vertical scroll bar. The screen is displayed in the screen window with the vertical shift of the scroll bar's slider.

To display the screen as a cutout where the scroll bars are located at the left and top edge of the screen, use the LeftOffset (Page 778)" and "LeftOffset (Page 778)" properties as the source of this cutout.

Access in Runtime: Read and write

Syntax

Object.**VerticalScrollBarPosition**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- Screenwindow

Int32

Optional. A value or a constant that specifies the vertical position of the slider in the object's scroll bar.

See also

TopOffset (Page 1088)

ScreenWindow (Page 405)

VerticalScrolling (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

VerticalScrollingEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ViewOnly (RT Advanced)

Description

Specifies whether the Sm@rtClient display will be used for remote monitoring or remote maintenance.

Remote maintenance means that settings can be changed on the monitored device.

Remote monitoring means that settings of the monitored device cannot be changed.

Access in runtime: Read and write

Syntax

Object.**ViewOnly**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- SmartClientView

BOOLEAN

Optional TRUE, if the Sm@rtClient display is to be used only for remote monitoring.

See also

SmartClientView (Page 413)

ViewType (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

ViewTypeForSaveStream (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Access in Runtime:

- RT Advanced: Read
- RT Professional: No access

Syntax

Object.ViewTimeForSaveStream[=[!]: AlarmViewBasicMode | Int16 | PasswordViewType]

Object

Required. An object of the type "ScreenItem" with the format:

- RecipeView

You have no access in runtime with the following format:

- AlarmView
- UserView

[!]: AlarmViewBasicMode | Int16 | PasswordViewType:

Optional. tbd tbd tbd tbd are.

Visible (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the selected object is visible.

Access in runtime: Read and write

Syntax

Object.Visible[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the format:

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- CircleSegment

- CircularArc
- ComboBox
- Connector
- DateTimeField
- DiscSpaceView
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- ProDiagOverview
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- RoundButton
- S7GraphOverview
- ScreenWindow
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TrendRulerControl

- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- WlanQualityView
- WindowSlider
- ZoneLabelView
- ZoneQualityView

You have no access in runtime with the following format:

- PdfView

BOOLEAN

Optional. TRUE, if the object is visible.

See also

AlarmControl (Page 236)

AlarmView (Page 251)

ApplicationWindow (Page 256)

Bar (Page 258)

BatteryView (Page 265)

Button (Page 266)

ChannelDiagnose (Page 274)

CheckBox (Page 275)

Circle (Page 279)

CircleSegment (Page 282)

CircularArc (Page 285)

Clock (Page 287)

ComboBox (Page 290)

DateTimeField (Page 296)

DiskSpaceView (Page 298)

Ellipse (Page 300)

EllipseSegment (Page 303)

EllipticalArc (Page 306)

FunctionTrendControl (Page 309)

Gauge (Page 320)
GraphicIOField (Page 324)
HTMLBrowser (Page 330)
IOField (Page 332)
Line (Page 337)
Listbox (Page 340)
MediaPlayer (Page 343)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)
PLCCodeViewer (Page 377)
Polygon (Page 379)
Polyline (Page 382)
ProtectedAreaNameView (Page 386)
RangeLabelView (Page 387)
RangeQualityView (Page 389)
RecipeView (Page 390)
Rectangle (Page 396)
RoundButton (Page 399)
ScreenWindow (Page 405)
Slider (Page 410)
SmartClientView (Page 413)
StatusForce (Page 415)
Switch (Page 420)
SymbolicIOField (Page 424)
SymbolLibrary (Page 429)
SysDiagControl (Page 431)
TextField (Page 440)
TrendRulerControl (Page 444)
TrendView (Page 454)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)

UserArchiveControl (Page 469)

UserView (Page 481)

WindowSlider (Page 485)

ZoneLabelView (Page 490)

ZoneQualityView (Page 491)

S7GraphOverview (Page 403)

WLanQualityView (Page 488)

VisibleItems (RT Advanced)

Description

No access in runtime.

Warning (RT Professional)

Description

Specifies the percentage of used memory space as of which a warning is to be generated.

Access in runtime: Read

Syntax

Object.**Warning**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- DiskSpaceView

Int32

Optional. A value or a constant that specifies the percentage of used storage space as of which a warning is generated.

See also

DiskSpaceView (Page 298)

WarningColor (RT Professional)

Description

Specifies the color in which the bar of the memory space view will be shown as soon as the warning level is reached.

Access in runtime Read

Syntax

Object.**WarningColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- DiskSpaceView

Color

Optional. A value or a constant that specifies the color in which the bar of the storage space display will be shown as soon as the warning area is exceeded.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

DiskSpaceView (Page 298)

WarningLowerLimit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the lower limit for "WarningLowerLimit".

The "WarningLowerLimitEnable" property must have the value "TRUE" for the limit to be monitored.

Access in runtime: Read and write

Syntax

Object.**WarningLowerLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the lower limit for "WarningLowerLimit".

Comments

The following values will be specified with the properties "WarningLowerLimitColor" and "WarningLowerLimitRelative":

- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

WarningLowerLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)**Description**

Specifies the color for the low limit "WarningLowerLimit".

The "WarningLowerLimitEnable" property must have the value "TRUE" if the bar color is to change when the limit is reached.

Access in runtime Read and write

Syntax

Object.WarningLowerLimitColor[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the lower limit "WarningLowerLimit".

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

The following values are defined with the "WarningUpperLimit", "WarningUpperLimitColor" and "WarningUpperLimitRelative" properties:

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

WarningLowerLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the "WarningLowerLimit" limit is to be monitored.

Access in runtime: Read and write

Syntax

Object.**WarningLowerLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional TRUE, if the "WarningLowerLimit" limit is monitored.

Comments

The following values will be defined via the properties "WarningLowerLimit", "WarningLowerLimitColor" and "WarningLowerLimitRelative":

- Limit
- Representation upon reaching the limit
- Type of evaluation

See also

Bar (Page 258)

WarningLowerLimitRelative (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the low limit "WarningLowerLimit" is evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.**WarningLowerLimitRelative**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BOOLEAN

Optional.

TRUE if the low limit "WarningLowerLimit" is to be evaluated as a percentage.

FALSE if the low limit "WarningLowerLimit" is to be evaluated as an absolute value.

See also

Bar (Page 258)

WarningRangeColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color of the warning range on the scale of the "Gauge" object.

The "WarningRangeVisible" property must have the value TRUE so that the warning range is displayed.

Access in runtime Read and write

Syntax

Object.**WarningRangeColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Gauge

Color

Optional. A value or a constant that specifies the color of the warning range.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Gauge (Page 320)

WarningRangeStart (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies at which scale value the warning range of the "Gauge" object will start.

The "WarningRangeVisible" property must have the value TRUE so that the warning range is displayed.

Access in runtime: Read and write

Syntax

Object.**WarningRangeStart**[=SINGLE]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

SINGLE

Optional A value or a constant that contains the scale value for the start of the warning range.

Comments

The range extends from the value "Warning" through to the value "Danger". If no range is activated for "Danger", the range for "Warning" extends to the end of the scale.

See also

Gauge (Page 320)

WarningRangeVisible (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the warning range in the scale of the "Gauge" object will be displayed.

Access in runtime: Read and write

Syntax

Object.**WarningRangeVisible**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Gauge

BOOLEAN

Optional TRUE, if the warning range will be displayed in the scale.

Comments

Specifies the color of the warning range with the "WarningRangeColor" property.

Specifies the start of the warning range with the "WarningRangeStart" property.

See also

Gauge (Page 320)

WarningUpperLimit (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the upper warning limit.

The "WarningUpperLimitEnabled" property must have the value TRUE so that the limit is monitored.

Access in runtime: Read and write

Syntax

Object.**WarningUpperLimit**[=DOUBLE]

Object

Required. A "ScreenItem" object with the following format:

- Bar

DOUBLE

Optional A value or a constant that specifies the higher limit.

Comments

"WarningUpperLimitColor" defines the display for when the limit is reached.

"WarningUpperLimitRelative" specifies the type of evaluation.

See also

Bar (Page 258)

WarningUpperLimitColor (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the color for the high warning limit.

The ""WarningUpperLimitEnabled" property must have the value TRUE if the bar color is to change once the limit has been reached.

Access in runtime

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**WarningUpperLimitColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

Color

Optional. A value or a constant that specifies the color for the higher limit.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

Bar (Page 258)

WarningUpperLimitEnabled (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the higher limit is to be monitored.

Access in runtime:

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**WarningUpperLimitEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- Bar

BOOLEAN

Optional. TRUE, if the higher limit is monitored.

WarningUpperLimitRelative (RT Professional)

Description

Specifies whether the high limit "WarningUpperLimit" is to be evaluated as a percentage or as an absolute value.

Access in runtime Read and write

Syntax

Object.**WarningUpperLimitRelative**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Bar

BOOLEAN

Optional. TRUE if the low limit "WarningUpperLimit" is to be evaluated as a percentage.

See also

Bar (Page 258)

Width (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies the width of the object in pixels.

Access in runtime: Read and write

Syntax

Object.**Width**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- AlarmControl
- AlarmView*
- ApplicationWindow
- ChannelDiagnose
- CircularArc
- ComboBox
- DiscSpaceView
- EllipticalArc
- FunctionTrendControl*
- GraphicIOField*
- GraphicView*
- HTMLBrowser*
- ListBox
- MediaPlayer*
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- PLCCodeViewer*
- ProDiagOverview*
- S7GraphOverview*
- ScreenWindow
- SmartClientView

- Switch*
- SymbolLibrary*
- SymbolicIOField*
- TrendRulerControl
- TrendView*
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- UserArchiveControl
- WindowSlider

*: You have no access in "Runtime Advanced" and "Panel Runtime".

Int32

Optional. A value or a constant that specifies the width of the object in pixels.

See also

AlarmControl (Page 236)
AlarmView (Page 251)
ApplicationWindow (Page 256)
Bar (Page 258)
BatteryView (Page 265)
Button (Page 266)
ChannelDiagnose (Page 274)
CheckBox (Page 275)
Circle (Page 279)
CircleSegment (Page 282)
CircularArc (Page 285)
Clock (Page 287)
ComboBox (Page 290)
DateTimeField (Page 296)
DiskSpaceView (Page 298)
Ellipse (Page 300)
EllipseSegment (Page 303)
EllipticalArc (Page 306)
FunctionTrendControl (Page 309)
Gauge (Page 320)

GraphicIOField (Page 324)
GraphicView (Page 327)
HTMLBrowser (Page 330)
IOField (Page 332)
Line (Page 337)
Listbox (Page 340)
MediaPlayer (Page 343)
MultiLineEdit (Page 345)
OnlineTableControl (Page 348)
OnlineTrendControl (Page 360)
OptionGroup (Page 372)
PLCCodeViewer (Page 377)
Polygon (Page 379)
Polyline (Page 382)
ProtectedAreaNameView (Page 386)
RangeLabelView (Page 387)
RangeQualityView (Page 389)
RecipeView (Page 390)
Rectangle (Page 396)
RoundButton (Page 399)
ScreenWindow (Page 405)
Slider (Page 410)
SmartClientView (Page 413)
StatusForce (Page 415)
Switch (Page 420)
SymbolicIOField (Page 424)
SymbolLibrary (Page 429)
SysDiagControl (Page 431)
TextField (Page 440)
TrendRulerControl (Page 444)
TrendView (Page 454)
TubeArcObject (Page 460)
TubeDoubleTeeObject (Page 462)
TubePolyline (Page 464)
TubeTeeObject (Page 467)

UserArchiveControl (Page 469)

UserView (Page 481)

WindowSlider (Page 485)

WLanQualityView (Page 488)

ZoneLabelView (Page 490)

ZoneQualityView (Page 491)

S7GraphOverview (Page 403)

WindowCloseEnabled (RT Professional)

Description

Indicates whether a window can be closed in runtime.

Access in runtime: Read and write

Syntax

Object.**WindowCloseEnabled**[=BOOLEAN]

Object

Required. A "ScreenItem" object with the following format:

- ApplicationWindow
- ScreenWindow

BOOLEAN

Optional. TRUE, if the window can be closed in runtime.

See also

ApplicationWindow (Page 256)

ScreenWindow (Page 405)

WindowMaximizeEnabled (RT Professional)

Description

Returns whether the object can be maximized in runtime.

Access in runtime Read and write

Syntax

Object.**WindowMaximizeEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

Optional.

TRUE if the object can be maximized in runtime.

FALSE if the object cannot be maximized in runtime.

See also

ApplicationWindow (Page 256)

ScreenWindow (Page 405)

WindowMovingEnabled (RT Professional)

Description

Returns whether the object can be moved in runtime.

Access in runtime: Read and write

Syntax

Object.**WindowMovingEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

Optional.

TRUE if the object can be moved in runtime.

FALSE if the object cannot be moved in runtime.

See also

ApplicationWindow (Page 256)

ScreenWindow (Page 405)

WindowOnTop (RT Professional)

Description

Returns whether the object always remains in the foreground in runtime.

Access in runtime: Read and write

Syntax

Object.**WindowOnTop**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- ApplicationWindow
- Screenwindow

BOOLEAN

Optional.

TRUE if the object is always in the foreground in runtime.

FALSE if the object is not always in the foreground in runtime.

See also

ApplicationWindow (Page 256)

ScreenWindow (Page 405)

WindowsContents (RT Professional)

Description

Returns the content of the print job or script diagnostics.

Access in runtime Read and write

Syntax

Object.**WindowsContents**[=WindowContent]

Object

Required. An object of the type "ScreenItem" with the following format:

- ApplicationWindow

WindowContent

Optional. A value or a constant that returns the content displayed in the object.

Designation	Description
Function	An application of the global script is displayed. The templates used for this are defined in the "Template (Page 1002)" property: • GSC diagnostics The results of the diagnostics system are displayed. • GSC Runtime The analysis results on characteristics in runtime are displayed.
Print jobs	A report is displayed. The templates used for this are specified in the "Template (Page 1002)" property:

See also

ApplicationWindow (Page 256)

WindowSizingEnabled (RT Professional)**Description**

Indicates whether the size can be changed.

Access in runtime Read and write

Syntax

Object.**WindowSizingEnabled**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- ApplicationWindow
- ScreenWindow

BOOLEAN

Optional. TRUE, if the size can be changed.

See also

ApplicationWindow (Page 256)

ScreenWindow (Page 405)

WindowsStyle (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Specifies whether the object is displayed in the general Windows style.

Access in runtime:

- RT Advanced: NoAccess
- RT Professional: Read and write

Syntax

Object.**WindowsStyle**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- Button
- WindowSlider

You have no access in runtime with the following format:

- Switch

BOOLEAN

Optional. TRUE, if the object will be displayed in the general Windows style.

See also

Button (Page 266)

WindowSlider (Page 485)

Switch (Page 420)

WindowStartupPosition (RT Professional)

Description

Returns the position and the mode of the screen window when an independent screen window is configured. Set "IndipendentWindow (Page 745)" to "TRUE".

Access in runtime: Read and write

Syntax

Object.**WindowStartupPosition**[=PositionMode]

Object

Required. An object of the type "ScreenItem" with the following format:

- ScreenWindow

PositionMode

Optional. A value or a constant that returns the position and the mode of the screen window.

Designation	Description
Maximized	The screen window is adjusted to the monitor size.
As configured	The screen window is displayed in the configured size and at the configured position.
Centered	The screen window is displayed in the configured size in a central position.

See also

ScreenWindow (Page 405)

1.5.5.15 Properties X-Z (Panels, Comfort Panels, RT Advanced, RT Professional)**XAxes (Panels, Comfort Panels, RT Advanced, RT Professional)****Description**

No access in runtime.

XAxisAdd (RT Professional)**Description**

Creates a new X-axis. The newly created X-axis is automatically referenced with "XAxisIndex".

Access in runtime: Read and write

Syntax

Object.XAxisAdd[=STRING]

Object

Required. An object of the type "ScreenItem" with the format:

- FunctionTrendControl

STRING

Required. A value or a constant that specifies the name of the new X-axis with "XAxisName".

See also

XAxisIndex (Page 1235)

FunctionTrendControl (Page 309)

XAxisAlignment (RT Professional)

Description

Specifies the alignment of the X-axis referenced with "XAxisIndex".

Access in runtime Read and write

Syntax

Object.XAxisAlignment[=AxisAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

AxisAlignment

Optional. A value or a constant that specifies the alignment of the X-axis in relation to the trend window:

Value	Designation
0	Left
1	Right

See also

FunctionTrendControl (Page 309)

XAxisAutoPrecisions (RT Professional)

Description

Specifies whether the number of decimal places for an X-axis referenced with "XAxisIndex" is set automatically.

Access in runtime Read and write

Syntax

Object.XAxisAutoPrecisions[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the number of decimal places is automatically determined.

FALSE if the value of "XAxisPrecisions" is used.

See also

XAxisIndex (Page 1235)

FunctionTrendControl (Page 309)

XAxisAutoRange (RT Professional)

Description

Specifies whether the value range of an X-axis referenced with "XAxisIndex" is calculated automatically.

Access in runtime Read and write

Syntax

Object.XAxisAutoRange[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the value range is calculated automatically.

FALSE if the value range is specified with "XAxisBeginValue" and "XAxisEndValue".

See also

XAxisIndex (Page 1235)

XAxisEndValue (Page 1233)

XAxisBeginValue (Page 1232)

FunctionTrendControl (Page 309)

XAxisBeginValue (RT Professional)

Description

Specifies the lower end of the value range of an X-axis referenced with "XAxisIndex".

Also set "XAxisAutoRange" to "FALSE".

Access in runtime Read and write

Syntax

Object.**XAxisBeginValue**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

DOUBLE

Optional. A value or a constant that specifies the low limit of the value range of the referenced X-axis.

See also

XAxisAutoRange (Page 1230)

XAxisIndex (Page 1235)

FunctionTrendControl (Page 309)

XAxisColor (RT Professional)

Description

Specifies the color of an X-axis referenced with "XAxisIndex".

Also set "XAxisInTrendColor" to "FALSE".

Access in runtime Read and write

Syntax

Object.**XAxisColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Color

Optional. A value or a constant that specifies the color of the referenced axis.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

FunctionTrendControl (Page 309)

XAxisCount (RT Professional)**Description**

Specifies the number of configured X-axes.

Access in runtime: Read and write

Syntax

Object.**XAxisCount**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the number of configured X-axes.

See also

FunctionTrendControl (Page 309)

XAxisEndValue (RT Professional)**Description**

Specifies the upper end of the value range of an X-axis referenced with "XAxisIndex".

Also set "XAxisAutoRange" to "FALSE".

Access in runtime Read and write

Syntax

Object.**XAxisEndValue**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

DOUBLE

Optional. A value or a constant that specifies the high limit of the value range of the referenced X-axis.

See also

XAxisAutoRange (Page 1230)

XAxisIndex (Page 1235)

FunctionTrendControl (Page 309)

XAxisExponentialFormat (RT Professional)

Description

Specifies whether the values of an X-axis referenced with "XAxisIndex" are displayed in exponential notation.

Access in runtime Read and write

Syntax

Object.**XAxisExponentialFormat**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the values are displayed in exponential notation.

FALSE if the values are not displayed in exponential notation.

See also

XAxisIndex (Page 1235)

FunctionTrendControl (Page 309)

XAxisIndex (RT Professional)

Description

References an X-axis. To access the properties of an X-axis, you need to set ""YAxisIndex"".

Values between 0 and "XAxisCount" minus 1 are valid for "XAxisIndex". The "XAxisCount" property specifies the number of configured X-axes.

Access in runtime Read and write

Syntax

Object.XAxisIndex[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies, with the index, the X-axis to be edited.

See also

XAxisCount (Page 1232)

FunctionTrendControl (Page 309)

XAxisInTrendColor (RT Professional)

Description

Specifies whether the color of the axis referenced with "XAxisIndex" corresponds to the trend color.

Access in runtime Read and write

Syntax

Object.XAxisInTrendColor[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the referenced axis is shown in the trend color. The value of "XAxisColor" is not effective.

FALSE if the referenced axis is shown in the color specified in "XAxisColor".

See also

XAxisIndex (Page 1234)

FunctionTrendControl (Page 309)

XAxisLabel (RT Professional)**Description**

Specifies the label text for an X-axis referenced with "XAxisIndex".

Access in runtime Read and write

Syntax

Object.XAxisLabel[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the label text for the referenced X-axis.

See also

XAxisIndex (Page 1234)

FunctionTrendControl (Page 309)

XAxisName (RT Professional)**Description**

Specifies the name of an X-axis referenced with "XAxisIndex".

Access in runtime Read and write

Syntax

Object.**XAxisName**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced X-axis.

See also

XAxisIndex (Page 1234)

FunctionTrendControl (Page 309)

XAxisPrecisions (RT Professional)

Description

Specifies the number of decimal places to be displayed for the values of an X-axis referenced with "XAxisIndex".

Also set "XAxisAutoPrecisions" to "FALSE".

Access in runtime Read and write

Syntax

Object.**XAxisPrecisions**[=Int16]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int16

Optional. A value or a constant that specifies the number of decimal places.

See also

FunctionTrendControl (Page 309)

XAxisRemove (RT Professional)

Description

Removes an X-axis referenced with its name.

Access in runtime Read and write

Syntax

Object.**XAxisRemove**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced X-axis to be removed.

See also

FunctionTrendControl (Page 309)

XAxisRename (RT Professional)

Description

Specifies the new name of an X-axis referenced with "XAxisIndex".

Access in runtime Read and write

Syntax

Object.**XAxisRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the new name of the X-axis.

See also

XAxisIndex (Page 1234)

FunctionTrendControl (Page 309)

XAxisRepos (RT Professional)

Description

For multiple X-axes, specifies the position of an X-axis referenced with "XAxisIndex".

If you have changed the position of the X-axis with "XAxisRepos", "XAxisIndex" is assigned the value of "XAxisRepos".

Access in runtime Read and write

Syntax

Object.XAxisRepos[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the position of the referenced X-axis. Value range from 0 to "XAxisCount" minus 1. Values outside this range have no effect.

0: The referenced X-axis is positioned outside.

See also

XAxisIndex (Page 1234)

XAxisCount (Page 1232)

FunctionTrendControl (Page 309)

XAxisScalingType (RT Professional)

Description

Specifies the type of scaling for the X-axis referenced with "XAxisIndex".

Access in runtime Read and write

Syntax

Object.XAxisScalingType[=AxisScalingType]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

AxisScalingType

Optional. A value or a constant that specifies the type of scaling of the referenced X axis.

Value	VB constant	Designation
0	hmiBarScalingLinear	Linear
1	hmiBarScalingLogarithmic	Logarithmic
2	hmiBarScalingNegativeLogarithmic	Negative logarithmic

See also

XAxisIndex (Page 1234)

FunctionTrendControl (Page 309)

XAxisTrendWindow (RT Professional)**Description**

Specifies the trend diagram in which the axis referenced with "XAxisIndex" is displayed.

Access in runtime Read and write

Syntax

Object.XAxisTrendWindow[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the name of the trend diagram.

See also

XAxisIndex (Page 1234)

FunctionTrendControl (Page 309)

XAxisVisible (RT Professional)**Description**

Specifies whether the X-axis referenced with "XAxisIndex" is displayed in the object.

Access in runtime Read and write

Syntax

Object.XAxisVisible[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE, if the referenced X axis is displayed.

FALSE, if the referenced X axis is not displayed.

See also

XAxisIndex (Page 1234)

FunctionTrendControl (Page 309)

YAxes (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

No access in runtime.

YAxisAdd (RT Professional)

Description

Creates a new Y-axis. The newly created Y-axis is automatically referenced with "YAxisIndex".

Access in runtime: Read and write

Syntax

Object.YAxisAdd[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Required. A value or a constant that specifies the name of the new Y-axis with "YAxisName".

See also

YAxisIndex (Page 1246)

FunctionTrendControl (Page 309)

YAxisAlignment (RT Professional)**Description**

Specifies the alignment of the Y-axis referenced with "YAxisIndex".

Access in runtime Read and write

Syntax

Object.YAxisAlignment[=AxisAlignment]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

AxisAlignment

Optional. A value or a constant that specifies the alignment of the Y-axis with regard to the trend window:

Value	Designation
0	Left
1	Right

See also

FunctionTrendControl (Page 309)

YAxisAutoPrecisions (RT Professional)**Description**

Specifies whether the number of decimal places is specified automatically for a Y-axis referenced with "YAxisIndex".

Access in runtime Read and write

Syntax

Object.YAxisAutoPrecisions[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the number of decimal places is automatically determined.

FALSE if the value of "YAxisPrecisions" is used.

See also

FunctionTrendControl (Page 309)

YAxisAutoRange (RT Professional)

Description

Specifies whether the value range of a Y-axis referenced with "YAxisIndex" is calculated automatically.

Access in runtime Read and write

Syntax

Object.YAxisAutoRange[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the value range is calculated automatically.

FALSE if the value range is specified with "YAxisBeginValue" and "YAxisEndValue".

See also

FunctionTrendControl (Page 309)

YAxisBeginValue (RT Professional)

Description

Specifies the lower end of the value range of a Y-axis referenced with "YAxisIndex".

Also set "YAxisAutoRange" to "FALSE".

Access in runtime: Read and write

Syntax

Object.**YAxisBeginValue**[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

DOUBLE

Optional. A value or a constant that specifies the low limit of the value range of the referenced Y-axis.

See also

FunctionTrendControl (Page 309)

YAxisColor (RT Professional)

Description

Specifies the color of a Y-axis referenced with "YAxisIndex".

Also set "YAxisInTrendColor" to "FALSE".

Access in runtime Read and write

Syntax

Object.**YAxisColor**[=Color]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Color

Optional. A value or a constant that specifies the color of the referenced axis.

Comments

You use the "RGB" function to specify the color in RGB format (red, green, blue). To do this, enter the appropriate decimal value for each of the three RGB values (range from 0 to 255). For example, the color "red" is shown like this:

RGB (255, 0, 0)

You can also use the VBS color constants such as `vbRed` and `vbGreen`.

See also

FunctionTrendControl (Page 309)

YAxisCount (RT Professional)

Description

Specifies the number of configured Y-axes.

Access in runtime Read and write

Syntax

Object.YAxisCount[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the number of configured Y-axes.

See also

FunctionTrendControl (Page 309)

YAxisEndValue (RT Professional)

Description

Specifies the upper end of the value range of a Y-axis referenced with "YAxisIndex".

Also set "YAxisAutoRange" to "FALSE".

Access in runtime Read and write

Syntax

Object.YAxisEndValue[=DOUBLE]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

DOUBLE

Optional. A value or a constant that specifies the high limit of the value range of the referenced Y-axis.

See also

FunctionTrendControl (Page 309)

YAxisExponentialFormat (RT Professional)**Description**

Specifies whether the values of a Y-axis referenced with "YAxisIndex" are displayed in exponential notation.

Access in runtime Read and write

Syntax

Object.**YAxisExponentialFormat**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the values are displayed in exponential notation.

FALSE if the values are not displayed in exponential notation.

See also

FunctionTrendControl (Page 309)

YAxisIndex (RT Professional)**Description**

References a Y-axis. To access the properties of a Y-axis, you need to set ""YAxisIndex"".

Values between 0 and "YAxisCount" minus 1 are valid for "YAxisIndex". The "YAxisCount" property specifies the number of configured Y-axes.

Access in runtime Read and write

Syntax

Object.YAxisIndex[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies, with the index, the Y-axis to be edited.

See also

FunctionTrendControl (Page 309)

YAxisInTrendColor (RT Professional)

Description

Specifies whether the color of the axis referenced with "YAxisIndex" corresponds to the trend color.

Access in runtime Read and write

Syntax

Object.YAxisInTrendColor[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE if the referenced axis is shown in the trend color. The value of "YAxisColor" is not effective.

FALSE if the referenced axis is shown in the color specified in "YAxisColor".

See also

YAxisIndex (Page 1245)

FunctionTrendControl (Page 309)

YAxisLabel (RT Professional)

Description

Specifies the label text for a Y-axis referenced with "YAxisIndex".

Access in runtime Read and write

Syntax

Object.YAxisLabel[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the label text for the referenced Y-axis.

See also

FunctionTrendControl (Page 309)

YAxisName (RT Professional)

Description

Specifies the name of a Y-axis referenced with "YAxisIndex".

Access in runtime Read and write

Syntax

Object.YAxisName[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced Y-axis.

See also

FunctionTrendControl (Page 309)

YAxisPrecisions (RT Professional)

Description

Specifies the number of decimal places to be displayed for the values of a Y-axis referenced with "YAxisIndex".

Also set "YAxisAutoPrecisions" to "FALSE".

Access in runtime Read and write

Syntax

Object.YAxisPrecisions[=Int16]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int16

Optional. A value or a constant that specifies the number of decimal places.

See also

FunctionTrendControl (Page 309)

YAxisRemove (RT Professional)

Description

Removes a Y-axis referenced with its name.

Access in runtime Read and write

Syntax

Object.YAxisRemove[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the name of the referenced Y-axis to be removed.

See also

FunctionTrendControl (Page 309)

YAxisRename (RT Professional)**Description**

Specifies the new name of a Y-axis referenced with "YAxisIndex".

Access in runtime Read and write

Syntax

Object.**YAxisRename**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the new name of the selected Y-axis.

See also

FunctionTrendControl (Page 309)

YAxisRepos (RT Professional)**Description**

For multiple Y-axes, specifies the position of a Y-axis referenced with "YAxisIndex".

If you have changed positioning of the Y-axis with "YAxisRepos", "YAxisIndex" receives the value of "YAxisRepos".

Access in runtime

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**YAxisRepos**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

Int32

Optional. A value or a constant that specifies the position of the referenced Y-axis. Value range from 0 to "YAxisCount" minus 1. Values outside this range have no effect.

0: The referenced Y-axis is positioned outside.

YAxisScalingType (RT Professional)**Description**

Specifies the type of scaling for the Y-axis referenced with "YAxisIndex".

Access in runtime

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**YAxisScalingType**[=AxisScalingType]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

AxisScalingType

Optional. A value or a constant that specifies the type of scaling of the referenced Y axis.

Value	VB constant	Designation
0	hmiBarScalingLinear	Linear
1	hmiBarScalingLogarithmic	Logarithmic
2	hmiBarScalingNegativeLogarithmic	Negative logarithmic

YAxisTrendWindow (RT Professional)**Description**

Specifies the trend diagram in which the axis referenced with "YAxisIndex" is displayed.

Access in runtime

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**YAxisTrendWindow**[=STRING]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

STRING

Optional. A value or a constant that specifies the name of the trend diagram.

YAxisVisible (RT Professional)**Description**

Specifies whether the Y-axis referenced with "YAxisIndex" is displayed in the object.

Access in runtime

- RT Advanced: Read
- RT Professional: Read and write

Syntax

Object.**YAxisVisible**[=BOOLEAN]

Object

Required. An object of the type "ScreenItem" with the following format:

- FunctionTrendControl

BOOLEAN

Optional.

TRUE, if the referenced Y axis is displayed.

FALSE, if the referenced Y axis is not displayed.

ZeroPoint (RT Professional)**Description**

Specifies the position of the zero point as a percentage of the bar height. The zero point can also be located outside of the displayed area.

The "ScalingType" property must be set to "Auto".

The "ShowScale" property must be set to "TRUE".

Access in runtime: Read and write

Syntax

Object.**ZeroPoint**[=Int32]

Object

Required. A "ScreenItem" object with the following format:

- Bar

Int32

Optional. A value or a constant that specifies the position of the zero point as a percentage of the bar height.

See also

Bar (Page 258)

ZoomFactor (RT Professional)

Description

Specifies the zoom factor of a screen or screen window.

Access in runtime Read and write

Syntax

Object.**ZoomFactor**[=Int32]

Object

Required. An object of the type "ScreenItem" with the following format:

- ScreenWindow

Int32

Optional. A value or a constant that specifies the zoom factor for displaying the embedded graphic.

See also

ScreenWindow (Page 405)

1.5.6 Methods (Panels, Comfort Panels, RT Advanced, RT Professional)

1.5.6.1 Methods A-G (Panels, Comfort Panels, RT Advanced, RT Professional)

Activate (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

It only makes sense to use the Activate method with the following operable screen objects. An error message is output at screen objects which cannot be operated, for example rectangles.

- IOField
- Switch
- SymbolLibrary
- TrendView
- FunctionTrendControl
- HTMLBrowser
- Slider
- GraphicIOField
- SymbolicIOField
- Button
- AlarmControl
- UserView
- UserArchiveControl
- SmartClientView
- StatusForce

In Runtime Advanced and Panels, the permanent area is activated when the method is applied to a screen object. Use the "BaseScreenName" property to activate a screen that has not been selected.

Syntax

```
Expression.Activate()
```

Expression

Required. An output which returns an object of the "Screen" or "ScreenItem" type.

Parameters

--

Example

```
Dim objO
Set objO = HMIRuntime.Screens("Screen_1").ScreenItems("IO-Field_1")
objO.Activate
```

See also

- [ScreenItem \(Page 218\)](#)
- [Screen \(Page 215\)](#)
- [ChannelDiagnose \(Page 274\)](#)
- [CheckBox \(Page 275\)](#)
- [Circle \(Page 279\)](#)
- [CircleSegment \(Page 282\)](#)
- [CircularArc \(Page 285\)](#)
- [Clock \(Page 287\)](#)
- [Connector \(Page 293\)](#)
- [DateTimeField \(Page 296\)](#)
- [DiskSpaceView \(Page 298\)](#)
- [Ellipse \(Page 300\)](#)
- [EllipseSegment \(Page 303\)](#)
- [EllipticalArc \(Page 306\)](#)
- [Gauge \(Page 320\)](#)
- [GraphicIOField \(Page 324\)](#)
- [GraphicView \(Page 327\)](#)
- [HTMLBrowser \(Page 330\)](#)
- [IOField \(Page 332\)](#)
- [Rectangle \(Page 396\)](#)
- [ScriptDiagnostics \(Page 408\)](#)
- [Switch \(Page 420\)](#)
- [SymbolicIOField \(Page 424\)](#)
- [SymbolLibrary \(Page 429\)](#)
- [TextField \(Page 440\)](#)
- [TrendView \(Page 454\)](#)
- [TubeArcObject \(Page 460\)](#)
- [TubeDoubleTeeObject \(Page 462\)](#)
- [TubePolyline \(Page 464\)](#)

TubeTeeObject (Page 467)
UserView (Page 481)
WindowSlider (Page 485)
StatusForce (Page 415)
SmartClientView (Page 413)
Slider (Page 410)
ScreenWindow (Page 405)
RoundButton (Page 399)
Polyline (Page 382)
Polygon (Page 379)
OptionGroup (Page 372)
MultiLineEdit (Page 345)
MediaPlayer (Page 343)
Listbox (Page 340)
Line (Page 337)
Bar (Page 258)
Button (Page 266)
OnlineTrendControl (Page 360)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)
FunctionTrendControl (Page 309)
AlarmView (Page 251)
AlarmControl (Page 236)

ActivateDynamic (RT Professional)

Description

Dynamically activates a trigger and the specified cycle for a property at runtime. This requires a VB script at the property as well as a trigger set to "On demand". Every time the trigger is activated a different activation cycle can be used.

Syntax

```
Expression.ActivateDynamic (ByVal Property name As String, ByVal  
Cycle name As Cycle)
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

Parameter	Description
Property name	Property to which trigger relates. Use the VBS name of the property.
Cycle name	Name of activation cycle, e.g. "CycleTime1s".

See also

[ChannelDiagnose \(Page 274\)](#)
[CheckBox \(Page 275\)](#)
[Circle \(Page 279\)](#)
[CircleSegment \(Page 282\)](#)
[CircularArc \(Page 285\)](#)
[Clock \(Page 287\)](#)
[Connector \(Page 293\)](#)
[DiskSpaceView \(Page 298\)](#)
[Ellipse \(Page 300\)](#)
[EllipseSegment \(Page 303\)](#)
[EllipticalArc \(Page 306\)](#)
[Gauge \(Page 320\)](#)
[GraphicIOField \(Page 324\)](#)
[GraphicView \(Page 327\)](#)
[HTMLBrowser \(Page 330\)](#)
[IOField \(Page 332\)](#)
[Rectangle \(Page 396\)](#)
[ScriptDiagnostics \(Page 408\)](#)
[SymbolicIOField \(Page 424\)](#)
[SymbolLibrary \(Page 429\)](#)
[TextField \(Page 440\)](#)
[TubeArcObject \(Page 460\)](#)
[TubeDoubleTeeObject \(Page 462\)](#)
[TubePolyline \(Page 464\)](#)
[TubeTeeObject \(Page 467\)](#)

UserView (Page 481)
WindowSlider (Page 485)
Slider (Page 410)
ScreenWindow (Page 405)
RoundButton (Page 399)
Polyline (Page 382)
Polygon (Page 379)
OptionGroup (Page 372)
MultiLineEdit (Page 345)
MediaPlayer (Page 343)
Listbox (Page 340)
Line (Page 337)
Bar (Page 258)
Button (Page 266)
OnlineTrendControl (Page 360)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
UserArchiveControl (Page 469)
FunctionTrendControl (Page 309)
AlarmControl (Page 236)

Add (RT Professional)

Description of TagSet Object

Adds a tag to the list. A tag may be added to the tag object by using name or reference.

syntax

```
Expression.Add [Tag]
```

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

VARIANT

Parameters	Description
Tag	Name of a WinCC tag or reference to a tag object to be added to the list.

Example:

In the following example, a TagSet object is generated and a tag is added.

```
'VBS170
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
```

Tag objects may also be added as follows.

```
'VBS171
Dim Tag
Set Tag = HMIRuntime.Tags("Motor2")
Dim group2
Set group2 = HMIRuntime.Tags.CreateTagSet
group2.Add Tag
```

Description of DataSet Object

Adds a value or object reference to the list.

Note

The Data Set Object does not support classes.

Objects of type Screen, Screens, ScreenItem, ScreenItems, Tag and TagSet cannot be included in the DataSet list.

For object references it must be ascertained that objects are multiread-enabled.

syntax

```
Expression.Add [vtName], [vtUserData]
```

Expression

Necessary. An expression which returns an object of type "DataSet".

Parameters

VARIANT

Parameters	Description
vtName	Name by which value or tag are to be added to list.
vtUserData	Value to be added to list.

Example:

In this example, a value is included in the DataSet list.

```
'VBS172  
HMIRuntime.DataSet.Add "Motor1",23
```

See also

DataSet (list) (Page 206)

OnlineTrendControl (Page 360)

AttachDB method (RT Professional)

Description

Executes the "Connect backup" key function of the control.

Syntax

```
Expression.AttachDB()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

FunctionTrendControl (Page 309)

AlarmControl (Page 236)

CalculateStatistic (RT Professional)

Description

Executes the "Calculate statistics" key function of the f(t) trend view and the table view.

Syntax

```
Expression.CalculateStatistic()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

[OnlineTrendControl](#) (Page 360)

[OnlineTableControl](#) (Page 348)

CopyRows (RT Professional)

Description

Executes the "Copy lines" key function of the control.

Syntax

```
Expression.CopyRows()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

[OnlineTrendControl](#) (Page 360)

[OnlineTableControl](#) (Page 348)

[UserArchiveControl](#) (Page 469)

[AlarmControl](#) (Page 236)

Create (RT Professional)

Description

Creates a new Alarm object.

Syntax

```
Expression.Create (VARIANT vtApplication)
```

Expression

Required. An expression which returns an object of type "Alarm".

Parameters

VARIANT

Parameters	Description
vtApplication	Name of alarm object (optional)

See also

OnlineTrendControl (Page 360)

CreateTagSet (RT Professional)

Description

Creates a new TagSet object. This object may be used for optimized multi-tag access.

syntax

```
Expression.CreateTagSet()
```

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

VARIANT

Example:

The following example shows how to create a TagSet object.

```
'VBS168  
'Build a Reference to the TagSet Object  
Dim group  
Set group = HMIRuntime.Tags.CreateTagSet
```

See also

OnlineTrendControl (Page 360)

CutRows (RT Professional)**Description**

Executes the "Cut rows" key function of the recipe view.

Syntax

```
Expression.CutRows()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

UserArchiveControl (Page 469)

DeactivateDynamic (RT Professional)**Description**

Deactivates the trigger of the "ActivateDynamic" method used for the specified property in/ during runtime.

Syntax

```
Expression.DeactivateDynamic(ByVal Property name As String)
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

Parameters	Description
Property name	Property, whose trigger is disabled. Use the VBS name of the property.

See also

[ChannelDiagnose \(Page 274\)](#)
[CheckBox \(Page 275\)](#)
[Circle \(Page 279\)](#)
[CircleSegment \(Page 282\)](#)
[CircularArc \(Page 285\)](#)
[Clock \(Page 287\)](#)
[Connector \(Page 293\)](#)
[DiskSpaceView \(Page 298\)](#)
[Ellipse \(Page 300\)](#)
[EllipseSegment \(Page 303\)](#)
[EllipticalArc \(Page 306\)](#)
[Gauge \(Page 320\)](#)
[GraphicIOField \(Page 324\)](#)
[GraphicView \(Page 327\)](#)
[HTMLBrowser \(Page 330\)](#)
[IOField \(Page 332\)](#)
[Rectangle \(Page 396\)](#)
[ScriptDiagnostics \(Page 408\)](#)
[SymbolicIOField \(Page 424\)](#)
[SymbolLibrary \(Page 429\)](#)
[TextField \(Page 440\)](#)
[TubeArcObject \(Page 460\)](#)
[TubeDoubleTeeObject \(Page 462\)](#)
[TubePolyline \(Page 464\)](#)
[TubeTeeObject \(Page 467\)](#)
[UserView \(Page 481\)](#)

WindowSlider (Page 485)
Slider (Page 410)
ScreenWindow (Page 405)
RoundButton (Page 399)
Polyline (Page 382)
Polygon (Page 379)
OptionGroup (Page 372)
MultiLineEdit (Page 345)
MediaPlayer (Page 343)
Listbox (Page 340)
Line (Page 337)
Bar (Page 258)
Button (Page 266)
OnlineTrendControl (Page 360)
OnlineTableControl (Page 348)
TrendRulerControl (Page 444)
FunctionTrendControl (Page 309)
AlarmControl (Page 236)

DeleteRows (RT Professional)

Description

Executes the "Delete rows" key function of the recipe view.

Syntax

```
Expression.DeleteRows()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)
UserArchiveControl (Page 469)

DetachDB (RT Professional)

Description

Executes the "Disconnect backup" key function of the control.

Syntax

```
Expression.DetachDB()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

FunctionTrendControl (Page 309)

AlarmControl (Page 236)

Edit (RT Professional)

Description

Executes the "Edit" key function of the table view.

Syntax

```
Expression.Edit()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

AlarmControl (Page 236)

Export (RT Professional)

Description

Executes the "Export log" or "Export data" key function of the control.

Syntax

```
Expression.Export()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

FunctionTrendControl (Page 309)

AlarmControl (Page 236)

GetColumn (RT Professional)

Description

Returns the column object of the recipe view designated by name or index as "ICCAxUAColumn" type.

Syntax

```
Ausdruck.GetColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the column of the recipe view

Example

```
'VBS312
Dim ctrl
Dim objColumn
Set ctrl = ScreenItems("RecipeControl")
Set objColumn = ctrl.GetColumn("Field1")
objColumn.Length = 30
Set objColumn = ctrl.GetColumn(3)
objColumn.Align = 2
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "Column" listing, for example, you write "objColumn.Align" instead of "objColumn.ColumnAlign".

See also

OnlineTrendControl (Page 360)

UserArchiveControl (Page 469)

GetColumnCollection (RT Professional)**Description**

Returns the list of all column objects of the recipe view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS313
Dim ctrl
Dim coll
Dim field
Set ctrl = ScreenItems("RecipeControl")
Set coll = ctrl.GetColumnCollection
HMIRuntime.Trace "Number of fields:" & coll.Count & vbCrLf
For Each field In coll
    HMIRuntime.Trace field.Name & vbCrLf
    HMIRuntime.Trace field.Type & vbCrLf
    HMIRuntime.Trace field.Length & vbCrLf
    HMIRuntime.Trace field.Caption & vbCrLf
Next
```

See also

[OnlineTrendControl](#) (Page 360)

[UserArchiveControl](#) (Page 469)

GetHistlistColumnCollection (RT Professional)**Description**

Returns the list of all column objects of the message view hit list as "ICCAxCollection" type.

Syntax

Expression.GetHitlistColumnCollection()

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS315
Dim ctrl
Dim coll
Dim hitlistcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetHitlistColumnCollection
HMIRuntime.Trace "Number of hitlist columns:" & coll.Count & vbCrLf
For Each hitlistcol In coll
    HMIRuntime.Trace hitlistcol.Index & vbCrLf
    HMIRuntime.Trace hitlistcol.Name & vbCrLf
    HMIRuntime.Trace hitlistcol.Sort & vbCrLf
    HMIRuntime.Trace hitlistcol.SortIndex & vbCrLf
Next
```

See also

[OnlineTrendControl \(Page 360\)](#)

[AlarmControl \(Page 236\)](#)

GetHitlistColumn (RT Professional)

Description

Returns the column object of the message view hit list designated by name or index as "ICCAxMessageColumn" type.

Syntax

```
Expression.GetHitlistColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of hitlist column

Example

```
'VBS314
Dim ctrl
Dim objHitlistColumn
Set ctrl = ScreenItems("AlarmControl")
Set objHitlistColumn = ctrl.GetHitlistColumn("Date")
objHitlistColumn.Sort = 2
Set objHitlistColumn = ctrl.GetHitlistColumn("AverageComeGo")
objHitlistColumn.Visible = FALSE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "HitlistColumn" listing, for example, you write "objHitlistColumn.Visible" instead of "objHitlistColumn.HitlistColumnVisible".

See also

[OnlineTrendControl \(Page 360\)](#)

[AlarmControl \(Page 236\)](#)

GetMessageBlock (RT Professional)

Description

Returns the message block of the message view designated by name or index as "ICCAxMessageBlock" type.

Syntax

```
Expression.GetMessageBlock(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of message block.

Example

```
'VBS316
Dim ctrl
Dim objMsgBlock
Set ctrl = ScreenItems("AlarmControl")
Set objMsgBlock = ctrl.GetMessageBlock("Date")
objMsgBlock.Align = 2
Set objMsgBlock = ctrl.GetMessageBlock("Number")
objMsgBlock.LeadingZeros = 4
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "MessageBlock" listing, for example, you write "objMsgBlock.Align" instead of "objMsgBlock.MessageBlockAlign".

See also

[OnlineTrendControl \(Page 360\)](#)

[AlarmControl \(Page 236\)](#)

GetMessageBlockCollection (RT Professional)

Description

Returns the list of all message block objects of the message view as "ICCAxCollection" type.

Syntax

```
Expression.GetMessageBlockCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS317
Dim ctrl
Dim coll
Dim msgblock
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageBlockCollection
For Each msgblock In coll
    msgblock.Align = 1
    msgblock.Length = 12
    msgblock.Selected = TRUE
Next
```


Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "MessageBlock" listing, for example, you write "msgblock.Align" instead of "msgblock.MessageBlockAlign".

See also

OnlineTrendControl (Page 360)

AlarmControl (Page 236)

GetMessageColumn (RT Professional)**Description**

Returns the column object of the message view designated by name or index as "ICCAxMessageColumn" type.

Syntax

```
Expression.GetMessageColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of column in message list.

Example

```
'VBS318
Dim ctrl
Dim objMessColumn
Set ctrl = ScreenItems("AlarmControl")
Set objMessColumn = ctrl.GetMessageColumn("Date")
objMessColumn.Visible = FALSE
Set objMessColumn = ctrl.GetMessageColumn("Number")
objMessColumn.Sort = 1
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "MessageColumn" listing, for example, you write "objMessColumn.Visible" instead of "objMessColumn.MessageColumnVisible".

See also

OnlineTrendControl (Page 360)

AlarmControl (Page 236)

GetMessageColumnCollection (RT Professional)

Description

Returns the list of all column objects of the message view as "ICCAxCollection" type.

Syntax

```
Expression.GetMessageColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS319
Dim ctrl
Dim coll
Dim msgcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageColumnCollection
HMIRuntime.Trace "Number of message columns:" & coll.Count & vbCrLf
For Each msgcol In coll
    HMIRuntime.Trace msgcol.Index & vbCrLf
    HMIRuntime.Trace msgcol.Name & vbCrLf
    HMIRuntime.Trace msgcol.Sort & vbCrLf
    HMIRuntime.Trace msgcol.SortIndex & vbCrLf
Next
```

See also

OnlineTrendControl (Page 360)

AlarmControl (Page 236)

GetOperatorMessage (RT Professional)

Description

Returns the operator message object of the message view designated by name or index as "ICCAxOperatorMessage" type.

Syntax

Expression.GetOperatorMessage(ByVal vIndex As Variant)

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of operator message

Example

```
'VBS320
Dim ctrl
Dim objOpMess
Set ctrl = ScreenItems("AlarmControl")
Set objOpMess = ctrl.GetOperatorMessage(0)
objOpMess.Source1 = "Number"
objOpMess.SourceType1 = 1
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "OperatorMessage" listing, for example, you write "objOpMess.Source1" instead of "objOpMess.OperatorMessageSource1".

See also

[OnlineTrendControl \(Page 360\)](#)

[AlarmControl \(Page 236\)](#)

GetOperatorMessageCollection (RT Professional)

Description

Returns the list of all operator message objects of the message view as "ICCAxCollection" type.

Syntax

```
Expression.GetOperatorMessageCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS321
Dim ctrl
Dim coll
Dim opmsg
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetOperatorMessageCollection
For Each opmsg In coll
    HMIRuntime.Trace opmsg.Index & vbCrLf
    HMIRuntime.Trace opmsg.Name & vbCrLf
    HMIRuntime.Trace opmsg.Number & vbCrLf
    HMIRuntime.Trace opmsg.Selected & vbCrLf
Next
```

See also

[OnlineTrendControl \(Page 360\)](#)

[AlarmControl \(Page 236\)](#)

GetRow (RT Professional)

Description

Returns the row object defined by its row number in the table-based controls as "ICCAxDataRow" type.

Syntax

```
Expression.GetRow(ByVal IRow As Long)
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

Long

Parameters	Description
IRow	Number of the desired line of the control.

Example

```
'VBS356
Dim coll
Dim ctrl
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("UAControl")
Set coll = ctrl.GetRowCollection
'enumerate and trace out row numbers
For lIndex = 1 To coll.Count
    HMIRuntime.trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & " "
    'enumerate and trace out column titles and cell texts
    For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
        HMIRuntime.trace ctrl.GetRow(0).CellText(lCellIndex) & " "
        HMIRuntime.trace ctrl.GetRow(lIndex).CellText(lCellIndex) & " "
    Next
    HMIRuntime.trace vbNewLine
Next
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

[TrendRulerControl \(Page 444\)](#)

[UserArchiveControl \(Page 469\)](#)

[AlarmControl \(Page 236\)](#)

GetRowCollection (RT Professional)

Description

Returns the list of all row objects of the table-based controls type "ICCAxDataRowCollection".

Syntax

```
Expression.GetRowCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Properties of the ICCAxDataRowCollection

The ICCAxDataRowCollection refers to runtime data. The data is read-only. It is not possible to add and edit the data.

The following properties are available for the ICCAxDataRowCollection:

- Count - Determines the number of rows in the collection.
- Item - Access to an individual row within the collection via the row number. Numbering runs from 1 to Count. A Row object is returned.

Example

```
'VBS357
Dim ctrl
Dim coll
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetRowCollection
HMIRuntime.Trace "Number of message rows:" & coll.Count & vbCrLf
'enumerate and trace out row numbers
For lIndex = 1 To coll.Count
    HMIRuntime.Trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & " "
    'enumerate and trace out column titles and cell texts
    For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
        HMIRuntime.Trace ctrl.GetMessageColumn(lCellIndex - 1).Name & " "
        HMIRuntime.Trace ctrl.GetRow(lIndex).CellText(lCellIndex) & " "
    Next
    HMIRuntime.Trace vbCrLf
Next
```

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AlarmControl (Page 236)

GetRulerBlock (RT Professional)**Description**

Returns the block object of the evaluation table designated by name or index as "ICCAxRulerBlock" type.

Syntax

```
Expression.GetRulerBlock(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the block in the evaluation table

Example

```
'VBS322
Dim ctrl
Dim objRulerBlock
Set ctrl = ScreenItems("RulerControl")
Set objRulerBlock = ctrl.GetRulerBlock(0)
objRulerBlock.Caption = "RulerBlock1"
Set objRulerBlock = ctrl.GetRulerBlock("Name")
objRulerBlock.Length = 10
```


Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "RulerBlock" listing, for example, you write "objRulerBlock.Caption" instead of "objRulerBlock.BlockCaption".

See also

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

GetRulerBlockCollection (RT Professional)**Description**

Returns the list of all block objects of the evaluation table as "ICCAxCollection" type.

Syntax

```
Expression.GetRulerBlockCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS323
Dim ctrl
Dim coll
Dim rulerblock
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerBlockCollection
For Each rulerblock In coll
    rulerblock.Align = 1
    rulerblock.Length = 12
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "RulerBlock" listing, for example, you write "rulerblock.Align" instead of "rulerblock.RulerBlockAlign".

See also

[OnlineTrendControl](#) (Page 360)

[TrendRulerControl](#) (Page 444)

GetRulerColumn (RT Professional)

Description

Returns the column object of the evaluation table designated by name or index as "ICCAxRulerColumn" type.

Syntax

```
Expression.GetRulerColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the column of the evaluation

Example

```
'VBS324
Dim ctrl
Dim objRulercol
Set ctrl = ScreenItems("RulerControl")
Set objRulercol = ctrl.GetRulerColumn("Name")
objRulercol.Sort = 0
Set objRulercol = ctrl.GetRulerColumn("ValueY")
objRulercol.Visible = FALSE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "RulerColumn" listing, for example, you write "objRulercol.Visible" instead of "objRulercol.ColumnVisible".

See also

[OnlineTrendControl \(Page 360\)](#)

[TrendRulerControl \(Page 444\)](#)

GetRulerColumnCollection (RT Professional)

Description

Returns the list of all column objects of the evaluation table as "ICCAxCollection" type.

Syntax

```
Expression.GetRulerColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS325
Dim ctrl
Dim coll
Dim rulercol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerColumnCollection
HMIRuntime.Trace "Number of ruler columns:" & coll.Count & vbCrLf
For Each rulercol In coll
    HMIRuntime.Trace rulercol.Index & vbCrLf
    HMIRuntime.Trace rulercol.Name & vbCrLf
    HMIRuntime.Trace rulercol.Sort & vbCrLf
    HMIRuntime.Trace rulercol.SortIndex & vbCrLf
Next
```

See also

[OnlineTrendControl \(Page 360\)](#)

[TrendRulerControl \(Page 444\)](#)

GetRulerData (RT Professional)

Description

Returns the value of the called trend at the ruler position.

Syntax

```
Expression.GetRulerData (ByVal RulerIndex As Long, pvValue As  
Variant, Optional pvTimeStamp As Variant, Optional pvFlags As  
Variant) Long
```

Expression

Necessary. An expression which returns an object of the "Trend" type.

Parameters

Parameters	Description
RulerIndex	0 =Ruler
pvValue	Value of X axis
pvTimeStamp	Time or value of the Y axis
pvFlags	Qualitycode

Example

```
'VBS326
Dim ctrl
Dim objTrend
Dim objIOField1
Dim objIOField2
Dim rulvalue
Dim runtime
Set ctrl = ScreenItems( "Control1" )
Set objTrend = ctrl.GetTrend( "Trend 1" )
Set objIOField1 = ScreenItems( "I/O Field1" )
Set objIOField2 = ScreenItems( "I/O Field2" )
objTrend.GetRulerData 0, rulvalue, runtime
objIOField1.OutputValue = rulvalue
objIOField2.OutputValue = runtime
```

See also

OnlineTrendControl (Page 360)

GetSelectedRow (RT Professional)

Description

Returns the selected row object of a table-based control as "ICCAxDataRow" type.

Syntax

```
Expression.GetSelectedRow()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Example

```
'VBS358
Dim ctrl
Dim lCellIndex
Dim lCellCount
Dim headingRow
Dim selectedRow
Set ctrl = ScreenItems("TableControl")
Set headingRow = ctrl.GetRow(0)
Set selectedRow = ctrl.GetSelectedRow
lCellCount = headingRow.CellCount
'enumerate and trace out column titles and cell texts
For lCellIndex = 1 To lCellCount
    HMIRuntime.trace headingRow.CellText(lCellIndex) & ":  "
    HMIRuntime.trace selectedRow.CellText(lCellIndex)
    HMIRuntime.trace vbNewLine
Next
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

[TrendRulerControl \(Page 444\)](#)

[UserArchiveControl \(Page 469\)](#)

[AlarmControl \(Page 236\)](#)

GetSelectedRows (RT Professional)

Description

Returns the selected row objects of a table-based control as type "ICCAxDataRow" for multiple selection.

Syntax

```
Expression.GetSelectedRows()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Example

```
'VBS359
Dim ctrl
Dim lCellIndex
Dim lCellCount
Dim lRowIndex
Dim lRowCount
Dim headingRow
Dim selectedRow
Dim selectedRows
Set ctrl = ScreenItems("TableControl")
Set headingRow = ctrl.GetRow(0)
Set selectedRows = ctrl.GetSelectedRows
lCellCount = headingRow.CellCount
lRowCount = selectedRows.Count
'enumerate selected rows
For lRowIndex = 1 To lRowCount
    Set selectedRow = selectedRows(lRowIndex)
    HMIRuntime.Trace "Row number: " & CStr(lRowIndex) & vbNewLine
    'enumerate and trace out column titles and cell texts
    For lCellIndex = 1 To lCellCount
        HMIRuntime.trace headingRow.CellText(lCellIndex) & ": "
        HMIRuntime.trace selectedRow.CellText(lCellIndex)
        HMIRuntime.trace vbNewLine
    Next
Next
Next
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Row" listing, for example, you write "objRow.CellCount" instead of "objRow.RowCellCount".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

[TrendRulerControl \(Page 444\)](#)

UserArchiveControl (Page 469)

AlarmControl (Page 236)

GetStatisticAreaColumn (RT Professional)

Description

Returns the column object of the statistic area window of the evaluation window designated by name or index as "ICCAxRulerColumn" type.

Syntax

```
Ausdruck.GetStatisticAreaColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of column of statistics area window.

Example

```
'VBS327
Dim ctrl
Dim objStatAreaCol
Set ctrl = ScreenItems("RulerControl")
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("DatasourceY")
objStatAreaCol.Visible = FALSE
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("ValueY(LL)")
objStatAreaCol.Sort = 1
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatisticAreaColumn" listing, for example, you write "objStatAreaCol.Visible" instead of "objStatAreaCol.ColumnVisible".

See also

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

GetStatisticAreaColumnCollection (RT Professional)**Description**

Returns the list of all column objects of the statistic area window of the evaluation table as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetStatisticAreaColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS328
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticAreaColumnCollection
HMIRuntime.Trace "Number of statistic Area columns:" & coll.Count & vbCrLf
For Each statcol In coll
    HMIRuntime.Trace statcol.Index & vbCrLf
    HMIRuntime.Trace statcol.Name & vbCrLf
    HMIRuntime.Trace statcol.Sort & vbCrLf
    HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
```

See also

[OnlineTrendControl \(Page 360\)](#)

[TrendRulerControl \(Page 444\)](#)

GetStatisticResultColumn (RT Professional)

Description

Returns the column object of the statistic window of the evaluation window designated by name or index as "ICCAxRulerColumn" type.

Syntax

```
Ausdruck.GetStatisticResultColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of column of statistics window.

Example

```
'VBS329
Dim ctrl
Dim objStatResCol
Set ctrl = ScreenItems("RulerControl")
Set objStatResCol = ctrl.GetStatisticResultColumn("MaxValue")
objStatResCol.Visible = FALSE
Set objStatResCol = ctrl.GetStatisticResultColumn("Average")
objStatResCol.Sort = 2
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatisticResultColumn" listing, for example, you write "objStatResCol.Visible" instead of "objStatResCol.ColumnVisible".

See also

OnlineTrendControl (Page 360)

TrendRulerControl (Page 444)

GetStatisticResultColumnCollection (RT Professional)

Description

Returns the list of all column objects of the statistic window of the evaluation table as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetStatisticResultColumnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS330
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticResultColumnCollection
HMIRuntime.Trace "Number of statistic result columns:" & coll.Count & vbCrLf
For Each statcol In coll
    HMIRuntime.Trace statcol.Index & vbCrLf
    HMIRuntime.Trace statcol.Name & vbCrLf
    HMIRuntime.Trace statcol.Sort & vbCrLf
    HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
```

See also

[OnlineTrendControl \(Page 360\)](#)

[TrendRulerControl \(Page 444\)](#)

GetStatusbarElement (RT Professional)

Description

Returns the element of the control status bar designated as name or index as type "ICCAxStatusbarElement".

Syntax

```
Ausdruck.GetStatusbarElement(ByVal vIndex As Variant)
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of status bar element.

Example

```
'VBS331
Dim ctrl
Dim objStatusBar
Set ctrl = ScreenItems( "Control1" )
Set objStatusBar = ctrl.GetStatusbarElement(1)
objStatusBar.Visible = FALSE
Set objStatusBar = ctrl.GetStatusbarElement(3)
objStatusBar.Width = 10
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatusbarElement" listing, for example, you write "objStatusBar.Visible" instead of "objStatusBar.StatusbarElementVisible".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

[TrendRulerControl \(Page 444\)](#)

[UserArchiveControl \(Page 469\)](#)

[FunctionTrendControl \(Page 309\)](#)

[AlarmControl \(Page 236\)](#)

GetStatusbarElementCollection (RT Professional)

Description

Returns the list of all status bar elements of the control as type "ICCAxCollection".

Syntax

```
Ausdruck.GetStatusbarElementCollection()
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS332
Dim ctrl
Dim coll
Dim statelement
Set ctrl = ScreenItems.Item("Controll")
Set coll = ctrl.GetStatusbarElementCollection
HMIRuntime.Trace "Number of statusbar elements:" & coll.Count & vbCrLf
For Each statelement In coll
    HMIRuntime.Trace statelement.Name & vbCrLf
    HMIRuntime.Trace statelement.Width & vbCrLf
    HMIRuntime.Trace statelement.Text & vbCrLf
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "StatusbarElement" listing, for example, you write "statelement.Name" instead of "statelement.StatusbarElementName".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

[TrendRulerControl \(Page 444\)](#)

UserArchiveControl (Page 469)

FunctionTrendControl (Page 309)

AlarmControl (Page 236)

GetTimeAxis (RT Professional)

Description

Returns the time object that is specified by name or index for the f(t) trend view as "ICCAxTimeAxis" type.

Syntax

```
Expression.GetTimeAxis(ByVal vIndex As Variant)
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of time axis.

Example

```
'VBS333
Dim ctrl
Dim objTimeAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTimeAxis = ctrl.GetTimeAxis(1)
objTimeAxis.Visible = FALSE
Set objTimeAxis = ctrl.GetTimeAxis("axis 2")
objTimeAxis.Label = "Time axis 2"
objTimeAxis.DateFormat = "dd.MM.yy"
objTimeAxis.TimeFormat = "HH:mm:ss.ms"
objTimeAxis.RangeType = 2
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeAxis.BeginTime = "06.04.2010 9:33:18"
'objTimeAxis.BeginTime = "04/06/2010 9:33:18"
objTimeAxis.MeasurePoints = 100
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "TimeAxis" listing, for example, you write "objTimeAx.Visible" instead of "objTimeAx.TimeAxisVisible".

See also

OnlineTrendControl (Page 360)

GetTimeAxisCollection (RT Professional)

Description

Returns a list of all time objects of the f(t) trend view as "ICCAxCollection" type.

Syntax

```
Expression.GetTimeAxisCollection()
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for ICCAxCollection:

- Count
- Item

The following functions are available for ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS334
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis1
Dim objTimeAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis1 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2010")
Set objTimeAxis2 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2011")
objTimeAxis1.TrendWindow = objTrendWnd.Name
objTimeAxis1.Label = "2010"
objTimeAxis1.RangeType = 1
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeAxis1.BeginTime = "01.01.2010 0:00:00"
'objTimeAxis1.BeginTime = "01/01/2010 0:00:00"
objTimeAxis1.EndTime = "31.12.2010 11:59:59"
'objTimeAxis1.EndTime = "12/31/2010 11:59:59"
objTimeAxis2.TrendWindow = objTrendWnd.Name
objTimeAxis2.Label = "2011"
objTimeAxis2.RangeType = 1
objTimeAxis2.BeginTime = "01.01.2011 0:00:00"
'objTimeAxis2.BeginTime = "01/01/2011 0:00:00"
objTimeAxis2.EndTime = "31.12.2011 11:59:59"
'objTimeAxis2.EndTime = "12/31/2011 11:59:59"
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis2.Name
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "TimeAxis" listing, for example, you write "objTimeAxis1.Label" instead of "objTimeAxis1.TimeAxisLabel".

See also

OnlineTrendControl (Page 360)

GetTimeColumn (RT Professional)

Description

Returns the time column object of the table view designated by name or index as "ICCAxTime Column" type.

Syntax

```
Ausdruck.GetTimeColumn(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of time column.

Example

```
'VBS335
Dim ctrl
Dim objTimeCol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn1")
objTimeCol.ShowDate = FALSE
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn2")
objTimeCol.Visible = FALSE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "TimeColumn" listing, for example, you write "objTimeColumn.ShowDate" instead of "objTimeColumn.TimeColumnShowDate".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

GetTimeColumnCollection (RT Professional)

Description

Returns the list of all time column objects of the table view as "ICCAxCollection" type.

Syntax

```
Expression.GetTimeColumnCollection()
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for ICCAxCollection:

- Count
- Item

The following functions are available for ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```

'VBS336
Dim ctrl
Dim objTimeCol1
Dim objTimeCol2
Dim coll
Dim timecol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol1 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2010")
Set objTimeCol2 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2011")
objTimeCol1.Caption = "2010"
objTimeCol1.RangeType = 1
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeCol1.BeginTime = "01.01.2010 0:00:00"
'objTimeCol1.BeginTime = "01/01/2010 0:00:00"
objTimeCol1.EndTime = "31.12.2010 11:59:59"
'objTimeCol1.EndTime = "12/31/2010 11:59:59"
objTimeCol2.Caption = "2011"
objTimeCol2.RangeType = 0
objTimeCol2.BeginTime = "01.01.2011 0:00:00"
'objTimeCol2.BeginTime = "01/01/2011 0:00:00"
objTimeCol2.TimeRangeFactor = 1
objTimeCol2.TimeRangeBase = 3600000
Set coll = ctrl.GetTimeColumnCollection
For Each timecol In coll
    timecol.Align = 1
    timecol.Length = 12
    timecol.BackColor = RGB(240,240,0)
    timecol.ForeColor = RGB(130,160,255)
Next

```

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

GetToolBarButton (RT Professional)

Description

Returns the button function designated by name or index on the control toolbar as "ICCAxToolBarButton" type.

Syntax

```
Ausdruck.GetToolBarButton(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of toolbar button function.

Example

```
'VBS337
Dim ctrl
Set ctrl = ScreenItems( "Control1" )
Dim toolbu
Set toolbu = ctrl.GetToolBarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "ToolBarButton" listing, for example, you write "toolbu.Index" instead of "toolbu.ToolBarButtonIndex".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

[TrendRulerControl \(Page 444\)](#)

[UserArchiveControl \(Page 469\)](#)

[FunctionTrendControl \(Page 309\)](#)

[AlarmControl \(Page 236\)](#)

GetToolBarButtonCollection (RT Professional)**Description**

Returns the list of all button functions of the control toolbar as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetToolBarButtonCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following methods are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS338
Dim ctrl
Dim coll
Dim toolbu
Set ctrl = ScreenItems( "Control1" )
Set coll = ctrl.GetToolBarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
For Each toolbu In coll
  HMIRuntime.Trace toolbu.Name & vbCrLf
  HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
  HMIRuntime.Trace "Authorization: " & toolbu.PasswordLevel & vbCrLf
Next
```

See also

[OnlineTrendControl \(Page 360\)](#)
[OnlineTableControl \(Page 348\)](#)
[TrendRulerControl \(Page 444\)](#)
[UserArchiveControl \(Page 469\)](#)
[FunctionTrendControl \(Page 309\)](#)
[AlarmControl \(Page 236\)](#)

GetTrend (RT Professional)

Description

Returns the f(t) or f(x) trend view designated by name or index as "ICCAxTrend" or "ICCAxFunctionTrend" type.

Syntax

```
Ausdruck.GetTrend(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of curve.

Example for Runtime Professional

```
'VBS339
Dim ctrl
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrend = ctrl.GetTrend( "Trend 1" )
objTrend.PointStyle = 1
objTrend.LineWidth = 4
Set objTrend = ctrl.GetTrend(2)
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag2"
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Trend" listing, for example, you write "objTrend.PointStyle" instead of "objTrend.TrendPointStyle".

See also

[OnlineTrendControl \(Page 360\)](#)

[FunctionTrendControl \(Page 309\)](#)

GetTrendCollection (RT Professional)

Description

Returns the list of all trends of the f(t) or f(x) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetTrendCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example for Runtime Professional

```
'VBS340
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag1"
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValAxis.Name
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "Trend" listing, for example, you write "objTrend.TagName" instead of "objTrend.TrendTagName".

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

GetTrendWindow (RT Professional)**Description**

Returns the trend view object of the f(t) trend view or the f(x) trend view designated by name or index as "ICCAxTrendWindow" type.

Syntax

```
Ausdruck.GetTrendWindow(ByVal vIndex As Variant)
```

Expression

Required An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of curve window.

Example for Runtime Professional

```
'VBS341
Dim ctrl
Dim objTrendWnd
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindow(1)
objTrendWnd.Visible = FALSE
Set objTrendWnd = ctrl.GetTrendWindow("trend window 2")
objTrendWnd.VerticalGrid = TRUE
objTrendWnd.FineGrid = TRUE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "TrendWindow" listing, for example, you write "objTrendWnd.Visible" instead of "objTrendWnd.TrendWindowVisible".

See also

[OnlineTrendControl](#) (Page 360)

[FunctionTrendControl](#) (Page 309)

GetTrendWindowCollection (RT Professional)

Description

Returns the list of all trend window objects of the f(t) trend display or the f(x) trend display as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetTrendWindowCollection()
```

Expression

Required. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example for Runtime Professional

```
'VBS342
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name
```

See also

[OnlineTrendControl \(Page 360\)](#)

[FunctionTrendControl \(Page 309\)](#)

GetValueAxis (RT Professional)

Description

Returns the value axis object of the f(t) trend view designated by name or index as "ICCAxValueAxis" type.

Syntax

```
Ausdruck.GetValueAxis(ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the value axis

Example

```
'VBS343
Dim ctrl
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objValAxis = ctrl.GetValueAxis(1)
objValAxis.Visible = FALSE
Set objValAxis = ctrl.GetValueAxis("axis 2")
objValAxis.Label = "Value axis 2"
objValAxis.ScalingType = 0
objValAxis.Precisions = 2
objValAxis.AutoRange = TRUE
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "ValueAxis" listing, for example, you write "objValueAx.Visible" instead of "objValueAx.ValueAxisVisible".

See also

OnlineTrendControl (Page 360)

GetValueAxisCollection (RT Professional)

Description

Returns the list of all value axis objects of the f(t) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetValueAxisCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS344
Dim ctrl
Dim objTrendWnd
Dim objValAxis1
Dim objValAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objValAxis1 = ctrl.GetValueAxisCollection.AddItem("myValueAxis1")
Set objValAxis2 = ctrl.GetValueAxisCollection.AddItem("myValueAxis2")
objValAxis1.TrendWindow = objTrendWnd.Name
objValAxis1.Label = "Value1"
objValAxis2.TrendWindow = objTrendWnd.Name
objValAxis2.inTrendColor = TRUE
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis2.Name
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "ValueAxis" listing, for example, you write "objValueAxis1.Label" instead of "objValueAxis1.ValueAxisLabel".

See also

OnlineTrendControl (Page 360)

GetValueColumn (RT Professional)

Description

Returns the value column object defined by name or index for the tabular view as "ICCAxValueColumn" type.

Syntax

```
Expression.GetValueColumn (ByVal vIndex As Variant)
```

Expression

Necessary. Expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of the value column of the f(t) trend view.

Example

```
'VBS345
Dim ctrl
Dim objValueColumn
Set ctrl = ScreenItems("TableControl")
Set objValueColumn = ctrl.GetValueColumn("ValueColumn1")
objValueColumn.Precisions = 4
Set objValueColumn = ctrl.GetValueColumn(2)
objValueColumn.ExponentialFormat = TRUE
```

Note

When you use the "Get..." methods to access properties from the control object list rather than with the control object, you have to omit the prefix of the property with the name of the list.

For the "ValueColumn" listing, for example, you write "objValueColumn.Precisions" instead of "objValueColumn.ValueColumnPrecisions".

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

GetValueColumnCollection (RT Professional)

Description

Returns the list of all value column objects of the f(t) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetValueColulmnCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS346
Dim ctrl
Dim objValCol1
Dim objValCol2
Dim coll
Dim valcol
Set ctrl = ScreenItems("TableControl")
Set objValCol1 = ctrl.GetValueColumnCollection.AddItem("ValueColumn1")
Set objValCol2 = ctrl.GetValueColumnCollection.AddItem("ValueColumn2")
objValCol1.Caption = "Value Archive"
objValCol1.Provider = 1
objValCol1.TagName = "ProcessValueArchive\arch1"
objValCol1.TimeColumn = "TimeColumn1"
objValCol2.Caption = "Value Tag"
objValCol2.Provider = 2
objValCol2.TagName = "tagxx"
objValCol2.TimeColumn = "TimeColumn2"
Set coll = ctrl.GetValueColumnCollection
For Each valcol In coll
    valcol.Align = 2
    valcol.Length = 10
    valcol.AutoPrecisions = TRUE
Next
```

See also

[OnlineTrendControl \(Page 360\)](#)

[OnlineTableControl \(Page 348\)](#)

GetXAxis (RT Professional)

Description

Returns the X axis object of the f(x) trend view designated by name or index as "ICCAxValueAxis" type.

Syntax

```
Ausdruck.GetXAxis (ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of X axis.

Example

```
'VBS347
Dim ctrl
Dim objXAx
Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAx = ctrl.GetXAxis(1)
objXAx.Visible = FALSE
Set objXAx = ctrl.GetXAxis("axis 2")
objXAx.Label = "X axis 2"
objXAx.ScalingType = 0
objXAx.Precisions = 2
objXAx.Color = RGB(109,109,109)
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "XAxis" listing, for example, you write "objXAx.Visible" instead of "objXAx.XAxisVisible".

See also

[OnlineTrendControl \(Page 360\)](#)

[FunctionTrendControl \(Page 309\)](#)

GetXAxisCollection (RT Professional)

Description

Returns the list of all X axis objects of the f(x) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetXAxisCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS348
Dim ctrl
Dim objXAxis1
Dim objXAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAxis1 = ctrl.GetXAxisCollection.AddItem("myXAxis1")
objXAxis1.Label = "temperature"
Set objXAxis2 = ctrl.GetXAxisCollection.AddItem("myXAxis2")
objXAxis2.Label = "pressure"
Set coll = ctrl.GetXAxisCollection
HMIRuntime.Trace "Number of XAxis:" & coll.Count & vbCrLf
For Each axes In coll
    HMIRuntime.Trace axes.Name & vbCrLf
    HMIRuntime.Trace axes.Label & vbCrLf
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "XAxis" listing, for example, you write "objXAxis1.Label" instead of "objXAxis1.XAxisLabel".

See also

[OnlineTrendControl \(Page 360\)](#)

[FunctionTrendControl \(Page 309\)](#)

GetYAxis (RT Professional)

Description

Returns the Y axis object of the f(x) trend view designated by name or index as "ICCAxValueAxis" type.

Syntax

```
Ausdruck.GetYAxis (ByVal vIndex As Variant)
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

VARIANT

Parameters	Description
vIndex	Index or name of Y axis.

Example

```
'VBS349
Dim ctrl
Dim objYAx
Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAx = ctrl.GetYAxis(1)
objYAx.Visible = FALSE
Set objYAx = ctrl.GetYAxis("axis 2")
objYAx.Label = "Y axis 2"
objYAx.Align = 0
objYAx.Precisions = 3
objYAx.EndValue = 90.000
objYAx.BeginValue = 10.000
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "YAxis" listing, for example, you write "objYAx.Visible" instead of "objYAx.YAxisVisible".

See also

[OnlineTrendControl \(Page 360\)](#)

[FunctionTrendControl \(Page 309\)](#)

GetYAxisCollection (RT Professional)

Description

Returns the list of all Y axis objects of the f(x) trend view as "ICCAxCollection" type.

Syntax

```
Ausdruck.GetYAxisCollection()
```

Expression

Necessary. An expression that returns an object of the "ScreenItem" type.

Parameters

--

Features and functions of the ICCAxCollection

The following properties are available for the ICCAxCollection:

- Count
- Item

The following functions are available for the ICCAxCollection:

- AddItem(vName) As Object
- RemoveItem(vIndex)

Example

```
'VBS350
Dim ctrl
Dim objYAxis1
Dim objYAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAxis1 = ctrl.GetXAxisCollection.AddItem("myYAxis1")
objYAxis1.Label = "temperature"
Set objYAxis2 = ctrl.GetXAxisCollection.AddItem("myYAxis2")
objYAxis2.Label = "pressure"
Set coll = ctrl.GetYAxisCollection
HMIRuntime.Trace "Number of YAxis:" & coll.Count & vbCrLf
For Each axes In coll
    HMIRuntime.Trace axes.Name & vbCrLf
    HMIRuntime.Trace axes.Label & vbCrLf
Next
```

Note

When you use the "Get..." methods to access properties from the Control object list rather than with the Control object, you have to omit the prefix of the property with the name of the list.

For the "YAxis" listing, for example, you write "objYAxis1.Label" instead of "objYAxis1.YAxisLabel".

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

1.5.6.2 Methods H-R (Panels, Comfort Panels, RT Advanced, RT Professional)**HideAlarm (RT Professional)****Description**

Executes the "Hide alarm" button function of the alarm view.

Syntax

```
Expression.HideAlarm()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

InsertData (RT Professional)**Description**

Adds data to the called trend.

Syntax

```
Expression.InsertData(dblAxisX As Variant, dblAxisY As Variant)
```

Expression

Necessary. An expression which returns an object of the "Trend" type.

Parameters

Parameters	Description
dblAxisX	Value of X axis
dblAxisY	Value of Y axis

Example

```
'VBS300
Dim lngFactor
Dim dblAxisX
Dim dblAxisY
Dim objTrendControl
Dim objTrend
Set objTrendControl = ScreenItems("Control1")
Set objTrend = objTrendControl.GetTrend("Trend 1")
For lngFactor = -100 To 100
    dblAxisX = CDbl(lngFactor * 0.02)
    dblAxisY = CDbl(dblAxisX * dblAxisX + 2 * dblAxisX + 1)
    objTrend.InsertData dblAxisX, dblAxisY
Next
```

Item (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns an element from a list.

Syntax

```
Expression.Item(Index)
```

Expression

Necessary. An expression which returns a list.

Parameters

Index

The name or the index number of an element of the list:

- ScreenItems-Listing: Use an object name, e.g. "HmiRuntime.Screens(1).ScreenItems("Circle")", or the index number.
- Screens-Listing: Use either the name or the index number.
- SmartTags-Listing: You can only use the tag name as index in the SmartTags list. A counting of all tags is not possible.

If the transferred value does not correspond with an element in the list, an error occurs. The return value has the value "Nothing".

```
On Error Resume Next
Dim screen
Set screen = HmiRuntime.Screens("Screen_1")
If (screen is Nothing)
then...
Else...
End If
```

Best practice for optimizing auto-completion support is to use combined addressing by means of screen name and object name, e.g.

```
"HmiRuntime.Screens("Screen").ScreenItems("Circle")".
```

Example

The item method is the default method for lists. Therefore, the results are the same for the following two examples:

```
'VBS_Example_Item
HMIruntime.Screens.Item(1)
HMIruntime.Screens(1)
```

Both instructions reference the respective base screen.

See also

[ScreenItems \(list\) \(Page 220\)](#)

[ScreenItem \(Page 218\)](#)

LockAlarm (RT Professional)

Description

Executes the "Disable alarm" button function of the alarm view.

Syntax

```
Expression.LockAlarm()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

LoopInAlarm (RT Professional)

Description

Executes the "Loop in alarm" button function of the alarm view.

Syntax

```
Expression.LoopInAlarm()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

MoveAxis (RT Professional)

Description

Executes the "Move axes area" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.MoveAxis()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

MoveToFirst (RT Professional)

Description

Executes the "First line" button function of the control.

Syntax

```
Expression.MoveToFirst()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

UserArchiveControl (Page 469)

MoveToFirstLine (RT Professional)

Description

Executes the "First alarm" button function of the alarm view.

Syntax

```
Expression.MoveToFirstLine()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

MoveToFirstPage (RT Professional)

Description

Executes the "First page" button function of the alarm view.

Syntax

```
Expression.MoveToFirstPage()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

MoveToLast (RT Professional)

Description

Executes the "Last data record" button function of the control.

Syntax

```
Expression.MoveToLast()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

UserArchiveControl (Page 469)

MoveToLastLine (RT Professional)

Description

Executes the "Last alarm" button function of the alarm view.

Syntax

```
Expression.MoveToLastLine()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

AlarmControl (Page 236)

MoveToLastPage (RT Professional)

Description

Executes the "Last page" button function of the alarm view.

Syntax

```
Expression.MoveToLastPage()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

MoveToNext (RT Professional)

Description

Executes the "Next data record" button function of the control.

Syntax

```
Expression.MoveToNext()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

UserArchiveControl (Page 469)

MoveToNextLine (RT Professional)

Description

Executes the "Next alarm" button function of the alarm view.

Syntax

```
Expression.MoveToNextLine()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

MoveToNextPage (RT Professional)

Description

Executes the "Next page" button function of the alarm view.

Syntax

```
Expression.MoveToNextPage()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

MoveToPrevious (RT Professional)

Description

Executes the "Previous data record" button function of the control.

Syntax

```
Expression.MoveToPrevious()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

UserArchiveControl (Page 469)

MoveToPreviousLine (RT Professional)

Description

Executes the "Previous alarm" button function of the alarm view.

Syntax

```
Expression.MoveToPreviousLine()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

MoveToPreviousPage (RT Professional)

Description

Executes the "Previous page" button function of the alarm view.

Syntax

```
Expression.MoveToPreviousPage()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

NextColumn (RT Professional)

Description

Executes the "Next column" button function of the table view.

Syntax

```
Expression.NextColumn()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTableControl (Page 348)

NextTrend (RT Professional)

Description

Executes the "Next trend" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.NextTrend()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

OneToOneView (RT Professional)

Description

Executes the "Original view" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.OneToOneView()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

PasteRows (RT Professional)**Description**

Executes the "Insert rows" button function of the recipe view.

Syntax

```
Expression.PasteRows()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

PreviousColumn (RT Professional)**Description**

Executes the "Previous column" button function of the table view.

Syntax

```
Expression.PreviousColumn()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTableControl (Page 348)

PreviousTrend (RT Professional)

Description

Executes the "Previous trend" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.PreviousTrend()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

Print (RT Professional)

Description

Executes the "Print" button function of the control.

Syntax

```
Expression.Print()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

FunctionTrendControl (Page 309)

AlarmControl (Page 236)

QuitHorn (RT Professional)

Description

Executes the "Alarm annunciator acknowledgment" button function of the alarm view.

Syntax

```
Expression.QuitHorn()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

QuitSelected (RT Professional)

Description

Executes the "Single acknowledgment" button function of the alarm view.

Syntax

```
Expression.QuitSelected()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

QuitVisible (RT Professional)

Description

Executes the "Group acknowledgment" button function of the alarm view.

Syntax

```
Expression.QuitVisible()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

Read (RT Professional)

Description of Tag Object

Reads out the status of a tag (tag object) shortly after the moment it was called. At the same time, the tag object is provided with the values read. Upon reading a tag, its value, quality code and time stamp are determined. The "LastError" property can be used to determine whether the call was successful.

The "Name" and "Tagprefix" properties are not changed by this.

If the value of the tag is read successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	Tag values
Name	Tag name (unchanged)
QualityCode	Quality level
Timestamp	Current tag time stamp
LastError	0
ErrorDescription	" "

If the value of the tag is not read successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	VT_Empty
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	Read operation error codes
ErrorDescription	Error description on LastError

syntax

```
Expression.Read([Readmode])
```

Expression

Necessary. An expression which returns a tag object. The return value of the Read method is the value of the tag read out.

Parameters

The optional "Readmode" parameter enables the distinction between two types of reading:

Parameters	Description
0	The tag value is read from the process image (cache). 0 is the default value.
1	The value of a tag is read directly from AS or channel (direct).

If the "Readmode" parameter is omitted, the value is read from the process image by default. The return value of the Read method is the tag value read out as VARIANT.

Reading From the Process Image

When reading from the process image, the tag is logged on and, from that moment, polled cyclically from the PLC. The login cycle is dependent on the configured trigger. The value is read from the tag image by WinCC. For Close Picture, the tag actions are ended again. The call is characterized by the following:

- The value is read by WinCC from the tag image.
- The call is faster in comparison to direct reading (except with the first call: The first call basically takes longer because the value from the PLC must be read out and logged on.)
- The duration of the call is not dependent on the bus load or AS.

Behavior in actions with a tag trigger

All of the tags contained in the tag trigger are already known with Open Picture and are registered with the defined monitoring time. Since all tags are requested at once, the best possible optimization can be targeted from the channel. If a tag, contained in the trigger, is requested with Read during an action, the value already exists and is transferred directly to

the call. If a tag is requested which is not contained in the trigger, the behavior is the same as with a standard trigger.

Behavior in actions with a cyclic trigger

tags are registered with half of the cycle time with the first call. For every other call, the value is present.

Behavior in event-driven actions

The tag is registered in the "upon change" mode with the first call. Process tags that are registered in the "upon change" mode correspond with a cyclic read job with a cycle time of 1s.

If an event (e.g. mouse click) requests a value asynchronously, the tag is transferred to the tag image. The tag is requested cyclically from the AS as of this point in time and therefore increases the basic load. To bypass this increase in the basic load, the value can also be read synchronously. The synchronous call causes a one-off increase in the communication load but the tag is not transferred to the tag image.

Direct reading

In the case of direct reading, the current value is returned. The tag is not registered cyclically, the value is requested from the AS one time only. Direct reading has the following properties:

- The value is read explicitly from the AS.
- The call takes longer compared to reading from the process image.
- The duration of the call is dependent on the bus load and AS, amongst other things.

Example

Reading a tag directly from AS or channel

```
'VBS100
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read(1)      'Read direct
MsgBox vntValue
```

Reading a tag from the process image

```
'VBS101
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read        'Read from cache
MsgBox vntValue
```

Description of TagSet Object

The TagSet object offers the option of reading several tags in one call.

Functionality here is mostly identical with that of a tag object. In the following, only deviations thereof are described.

Expression

Necessary. An expression which returns an object of type "TagSet".

Reading From the Process Image

The TagSet object offers the advantage of requesting several tags in one read command. The tags are registered in the process image as a group, improving performance in the process.

Direct reading

Since one call may process several read commands, performance is enhanced in comparison to single calls.

Example

The following example shows how tags are included in the TagSet list, how tag values are imported and subsequently read.

```
'VBS174
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Read
HMIRuntime.Trace "Motor1: " & group("Motor1").Value & vbNewLine
HMIRuntime.Trace "Motor2: " & group("Motor2").Value & vbNewLine
```

If the optional parameter "Readmode" is set to 1, the process tags are not registered but read directly from AS or channel.

```
group.Read 1
```

ReadTags (RT Professional)

Description

Executes the "Read tags" button function of the recipe view.

Syntax

```
Expression.ReadTags ()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

Refresh (RT Professional)**Description**

Drawing all visible pictures again.

syntax

```
Expression.Refresh
```

Expression

Necessary. An expression which returns a "Screens" or "Screen" type object.

Parameter

--

Examples

The first example forces all visible pictures to be drawn again:

```
'VBS149  
HMIRuntime.Screens.Refresh
```

The second example forces the basic picture to be immediately redrawn:

```
'VBS150  
HMIRuntime.Screens(1).Refresh
```


See also

Screen (Page 215)

HMIRuntime (Page 208)

Remove (RT Professional)

Description of TagSet Object

Removes a tag from the TagSet list. The tag may be removed by name or reference to a tag object.

Syntax

```
Expression.Remove [Tag]
```

Expression

Required An expression that returns an object of type "TagSet".

Parameters

VARIANT

Parameters	Description
Tag	Name of a WinCC tag or reference to a tag object to be removed from the list.

Example

The following example shows how several tags are included in the TagSet list, and how to remove a tag again.

```
'VBS175
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Remove "Motor1"
```

Description of DataSet Object

Deletes the element specified in parameter "Name" from a list.

Syntax

```
Expression.Remove [Name]
```

Expression

Required An expression which returns an object of type "DataSet".

Parameters

VARIANT

Parameters	Description
Name	Name of the object to be removed from the list.

Example

The example shows how to remove the object "motor1" from the list.

```
'VBS166  
HMIRuntime.DataSet.Remove("motor1")
```

Description of objects Logging, AlarmLogs, DataLogs

The method deletes previously swapped-in log segments from the Runtime project.

Log segments that were deleted using the "Remove" method are removed from the "Common logging" folder of the project.

The call may require a somewhat longer time period, depending on the log data. This may block the processing of subsequent scripts. Blockage of actions within the picture may be avoided if you start the call in a Global Scripting action, such as starting the action through a trigger tag.

The process of disconnecting and clearing logs creates a CPU load. This will affect performance.

Note

Calling up the "Remove" method is presently only possible at the server. There is an example, however, which shows how the method may be started by the client from a server.

Syntax

Objects Logging, AlarmLogs

```
Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [ServerPrefix]
```

Expression

Required An expression that returns an object of type "Logging" or "AlarmLogs".

Object DataLogs

Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [Type] [ServerPrefix]

Expression

Required An expression that returns an object of type "DataLogs".

Parameters

TimeFrom

Point in time, from which the logs are to be cleared.

When specifying the time information, a short form is also possible. This is described in the "Time format" section.

TimeTo

Time up to which log segments are to be cleared.

When specifying the time information, a short form is also possible. This is described in the "Time Format" section.

Timeout

Timeout in milliseconds.

If you enter "-1" as a value, the wait will be infinite. If you enter a value of "0", there will be no wait.

Type

Type of log.

The parameter can (optionally) be used only to delete log segments of the tag logging.

The following values can be entered:

Assigned value	Type	Description
1	hmiDataLogFast	Tag Logging Fast data
2	hmiDataLogSlow	Tag Logging Slow data
3	hmiDataLogAll	Tag Logging Fast and Slow data

ServerPrefix

Reserved for future versions.

Return value

If an error occurred during deletion of the log segments, the method will return an error alarm. Additional information may be found under the subject heading "Error alarms from database area".

Time format

The format for specifying time information is defined as follows: YYYY-MM-DD hh:mm:ss, where YYYY represents the year, MM the month, DD the day, hh the hour, mm the minute and ss the second. For example, the time of 2 minutes and one second past 11 o'clock on July 26, 2004 is displayed as follows: 2004-07-26 11:02:01.

For parameters "TimeFrom" and "TimeTo" the statement of data and time is also possible in short form. Not all format fields must be filled in this case. The short form means that the information on date and time may be lacking one or several parameters, beginning with the value for seconds. For example, the time may be specified in the "YYYY-MM" or "YYYY-MM-DD hh" format. Using the statement "TimeFrom" = "2004-09" and "TimeTo" = "2004-10-04" all log segments between September 2004 up to and including October 4.

Example

In the following example, log segments within a certain time period that were swapped in (again) after the fact are removed and the return value is output as a trace.

```
'VBS182
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("2004-08-22","2004-09-22",-1) &
vbNewLine
```

In the following example, all log segments that were swapped-in (again) after the fact are removed and the return value is output as a trace.

```
'VBS183
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("", "", -1) & vbNewLine
```

See also

- Logging (Page 213)
- DataSet (list) (Page 206)
- DataLogs (list) (Page 204)
- AlarmLogs (list) (Page 201)

RemoveAll (RT Professional)

Description of TagSet Object

Deletes all tags from a TagSet list.

syntax

```
Expression.RemoveAll
```

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

--

Example:

The following example shows how several tags are included in the TagSet list, and how to remove all tags again.

```
'VBS176
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.RemoveAll
```

Description of DataSet Object

Deletes all values or object references from a DataSet list.

syntax

Expression.RemoveAll

Expression

Necessary. An expression which returns an object of type "DataSet".

Parameters

--

Example:

The example shows how all objects are removed from the list.

```
'VBS167
HMIRuntime.DataSet.RemoveAll
```

See also

[DataSet \(list\) \(Page 206\)](#)

Restore (RT Professional)

Description of objects Logging, AlarmLogs, DataLogs

The method adds swapped-out log segments to the Runtime project.

Upon swapping-in, the log segments are copied to the "Common logging" folder of the project. Therefore the appropriate storage capacity must be available.

The call may require a somewhat longer time period, depending on the log data. This may block the processing of subsequent scripts. Blockage of actions within the picture may be avoided if you start the call in a Global Scripting action, such as starting the action through a trigger tag.

The connecting/copying of the logs generates a CPU load because the SQL server experiences additional load, especially if signature checking is activated. Copying of log segments will slow down hard disk access.

If signature checking is activated, an error message is returned if an unsigned or modified log is to be swapped in. There is always only one error alarm returned, even if several errors occurred during a swap-in process. Additionally, a WinCC system alarm is generated for each log segment. An entry is added to the Windows event viewer in the "Application" section. This provides the opportunity to check which log segments are creating the error.

- With an unsigned log, the return value "0x8004720F" is returned. The event viewer contains the entry "Validation of database <db_name> failed! No signature found!".
The log is swapped in.
- With a changed log, the return value "0x80047207" is returned. The event viewer contains the entry "Validation of database <db_name> failed!".
The log is not swapped in.

Note

Calling up the "Restore" method is presently only possible at the server. There is an example, however, which shows how the method may be started by the client from a server.

Syntax

Objects Logging, AlarmLogs

```
Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut]  
[ServerPrefix]
```

Expression

Required An expression that returns an object of type "Logging" or "AlarmLogs".

Object DataLogs

```
Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut] [Type]  
[ServerPrefix]
```

Expression

Required An expression that returns an object of type "DataLogs".

Parameters**SourcePath**

Path to log data.

TimeFrom

Point in time, from which the logs are to be swapped in.

When specifying the time information, a short form is also possible. This is described in the "Time format" section.

TimeTo

Time up to which log segments are to be swapped in.

When specifying the time information, a short form is also possible. This is described in the "Time Format" section.

Timeout

Timeout in milliseconds.

If you enter "-1" as a value, the wait will be infinite. If you enter a value of "0", there will be no wait.

Type

Type of log.

The parameter can (optionally) be used only to store log segments of the tag logging.

The following values can be entered:

Assigned value	Type	Description
1	hmiDataLogFast	Tag Logging Fast data
2	hmiDataLogSlow	Tag Logging Slow data
3	hmiDataLogAll	Tag Logging Fast and Slow data

ServerPrefix

Reserved for future versions.

Return value

If an error occurred while swapping in log segments, the method will return an error message. Additional information may be found under the subject heading "Error alarms from database area".

Time format

The format for specifying time information is defined as follows: YYYY-MM-DD hh:mm:ss, where YYYY represents the year, MM the month, DD the day, hh the hour, mm the minute and ss the second. For example, the time of 2 minutes and one second past 11 o'clock on July 26, 2004 is displayed as follows: 2004-07-26 11:02:01.

For parameters "TimeFrom" and "TimeTo" the statement of data and time is also possible in short form. Not all format fields must be filled in this case. The short form means that the information on date and time may be lacking one or several parameters, beginning with the value for seconds. For example, the statement may be in the form of "YYYY-MM" or "YYYY-MM-DD hh". Using the statement "TimeFrom" = "2004-09" and "TimeTo" = "2004-10-04" all log segments between September 2004 up to and including October 4.

Example

In the following example, all log segments since the start of the specified time period are swapped in again, and the return value is output as a trace.

```
'VBS184
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:\Folder", "2004-09-14", "", -1) &
vbNewLine
```

In the following example, all Tag Logging Slow log segments in the specified time period are swapped in again, and the return value is output as a trace.

```
'VBS185
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.DataLogs.Restore("D:\Folder", "2004-09-14
12:30:05", "2004-09-20 18:30", -1, 2) & vbNewLine
```

In the following example, all Alarm Logging log segments up to the specified time period are swapped in again, and the return value is output as a trace.

```
'VBS186
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.AlarmLogs.Remove("", "2004-09-20", -1) &
vbNewLine
```

See also

[Logging \(Page 213\)](#)

[DataLogs \(list\) \(Page 204\)](#)

[AlarmLogs \(list\) \(Page 201\)](#)

1.5.6.3 Methods S-Z (Panels, Comfort Panels, RT Advanced, RT Professional)

SelectAll (RT Professional)

Description

Selects all rows in a table-based control.

Syntax

```
Expression.SelectAll()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AlarmControl (Page 236)

SelectedStatisticArea (RT Professional)

Description

Executes the "Set statistics range" button function of the table view.

Syntax

```
Expression.SelectedStatisticArea()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTableControl (Page 348)

SelectRow (RT Professional)**Description**

Selects a specific row in a table-based control.

Syntax

```
Expression.SelectRow ByVal IRow As Long, Optional bExtendSelection  
As Boolean
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

Parameters	Description
IRow	Number of the row to be selected.
bExtendSelection	Indicates as an option whether the current selection will be extended. Is only relevant if multiple selections are possible.

Example

- Row 1 is currently selected. If `SelectRow 4, True` is called, then row 1 and row 4 are selected.
- Row 1 is currently selected. If `SelectRow 4, False` or `SelectRow 4` is called without an optional parameter, then only row 4 is selected.

See also

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AlarmControl (Page 236)

ServerExport (RT Professional)**Description**

Executes the "Export log" button function of the recipe view.

Syntax

```
Expression.ServerExport()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

ServerImport (RT Professional)**Description**

Executes the "Import log" button function of the recipe view.

Syntax

```
Expression.ServerImport()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

SetHTML (RT Professional)**Description**

Writes a HTML code in the display range of the HTML browser.

Syntax

```
Epression.SetHTML string
```

Expression

Required. An expression which returns an "HTMLBrowser" type object.

Example

```
Dim objBrowser
Set objBrowser= HMIRuntime.Screens("Screen_1").ScreenItems("HTML-Browser_1")
objBrowser.SetHTML "<h1>This is a title</h1><p>This is a <b>bold</b> text.</p>"
```

See also

HTMLBrowser (Page 330)

SetOperationLock (Panels, Comfort Panels, RT Advanced)**Description**

Temporarily switches off the operability of a screen window. The respective screen window can be defined as transparent. You determine the level of transparency yourself.

Note

This function is available for all screen windows except for the root screen.

Note

The contents of the temporary inoperable screen window are not updated on a PC with Windows 7 operating system.

The contents of the screen window are updated continuously on a PC with Windows 8 or Windows 10.

Syntax

```
Expression.SetOperationLock(ByVal LockState as Bool, ByVal  
TransparencyLevel as Long)
```

Expression

Required. An expression that returns an object of the "ScreenItem" type.

Parameters

Parameters	Description
True	The operability of the screen window was switched off temporarily.
False	The operability of the screen window was switched off temporarily.
Value	Value for the transparency of the screen window

Example

In the following example, you make the contents of the screen window "MyLockedWindow" inoperable in the Screen_1 screen by setting the "SetOperationLock" parameter to "true":

```
HmiRuntime.Screens("Screen_1").ScreenItems("MyLockedWindow").SetOperationLock true, 20
```

The number "20" determines the transparency of the screen window.

If you set the "SetOperationLock" parameter to "false", the "MyLockedWindow" screen window can be operated again.

```
HmiRuntime.Screens("Screen_1").ScreenItems("MyLockedWindow").SetOperationLock true, 20
```

ShowColumnSelection (RT Professional)

Description

Executes the "Select columns" button function of the table view.

Syntax

```
Expression.ShowColumnSelection()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTableControl (Page 348)

ShowComment (RT Professional)

Description

Executes the "Comment dialog" button function of the alarm view.

Syntax

```
Expression.ShowComment()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

AlarmControl (Page 236)

ShowDisplayOptionsDialog (RT Professional)

Description

Executes the "Display options dialog" button function of the alarm view.

Syntax

```
Expression.ShowDisplayOptionsDialog()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowEmergencyQuitDialog (RT Professional)

Description

Executes the "Emergency acknowledgment" button function of the alarm view.

Syntax

```
Expression.ShowEmergencyQuitDialog()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowHelp (RT Professional)**Description**

Executes the "Help" button function of the control.

Syntax`Expression.ShowHelp()`**Expression**

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

FunctionTrendControl (Page 309)

AlarmControl (Page 236)

ShowHideList (RT Professional)**Description**

Executes the "List of alarm to hide" button function of the alarm view.

Syntax`Expression.ShowHideList()`

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowHitList (RT Professional)

Description

Executes the "Hit list" button function of the alarm view.

Syntax

`Expression.ShowHitList()`

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowInfoText (RT Professional)

Description

Executes the "About dialog" button function of the alarm view.

Syntax

`Expression.ShowInfoText()`

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowLockDialog (RT Professional)**Description**

Executes the "Lock dialog" button function of the alarm view.

Syntax`Expression.ShowLockDialog()`**Expression**

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowLockList (RT Professional)**Description**

Executes the "Lock list" button function of the alarm view.

Syntax`Expression.ShowLockList()`**Expression**

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowLongTermArchiveList (RT Professional)

Description

Executes the "Historical alarm list (long-term)" button function of the alarm view.

Syntax

```
Expression.ShowLongTermArchiveList()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowMessageList (RT Professional)

Description

Executes the "Alarm list" button function of the alarm view.

Syntax

```
Expression.ShowMessageList()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowPercentageAxis (RT Professional)

Description

Executes the "Relative axis" button function of the f(t) trend view.

Syntax

```
Expression.ShowPercentageAxis()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTrendControl (Page 360)

ShowPropertyDialog (RT Professional)

Description

Executes the "Configuration dialog" button function of the control.

Syntax

```
Expression.ShowDialog()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

FunctionTrendControl (Page 309)

AlarmControl (Page 236)

ShowSelectArchive (RT Professional)

Description

Executes the "Select data connection" button function of the recipe view.

Syntax

```
Expression.ShowSelectArchive()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

ShowSelection (RT Professional)

Description

Executes the "Selection dialog" button function of the recipe view.

Syntax

```
Expression.ShowSelection()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

ShowSelectionDialog (RT Professional)

Description

Executes the "Selection dialog" button function of the alarm view.

Syntax

```
Expression.ShowSelectionDialog()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowSelectTimeBase (RT Professional)

Description

Executes the "Timebase dialog" button function of the recipe view.

Syntax

```
Expression.ShowSelectTimeBase()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

ShowShortTermArchiveList (RT Professional)

Description

Executes the "Historical alarm list (short-term)" button function of the alarm view.

Syntax

```
Expression.ShowShortTermArchiveList()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowSort (RT Professional)

Description

Executes the "Sorting dialog" button function of the recipe view.

Syntax

```
Expression.ShowSort()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

ShowSortDialog (RT Professional)

Description

Executes the "Sorting dialog" button function of the alarm view.

Syntax

```
Expression.ShowSortDialog()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowTagSelection (RT Professional)

Description

Executes the "Select data connection" button function of the control.

Syntax

```
Expression.ShowTagSelection()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

FunctionTrendControl (Page 309)

ShowTimebaseDialog (RT Professional)

Description

Executes the "Timebase dialog" button function of the alarm view.

Syntax

```
Expression.ShowTimebaseDialog()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

ShowTimeSelection (RT Professional)

Description

Executes the "Select time range" button function of the control.

Syntax

```
Expression.ShowTimeSelection()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

FunctionTrendControl (Page 309)

ShowTrendSelection (RT Professional)

Description

Executes the "Select trends" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.ShowTrendSelection()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

FunctionTrendControl (Page 309)

StartStopUpdate (RT Professional)

Description

Executes the "Start" or "Stop" button function of the control.

Syntax

```
Expression.StartStopUpdate()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

OnlineTableControl (Page 348)

FunctionTrendControl (Page 309)

Stop (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Closes WinCC Runtime.

Syntax

```
Expression.Stop()
```

Expression

Required. An expression which returns an object of the type "HMIRuntime".

Parameters

--

See also

HMIRuntime (Page 208)

Trace (Panels, Comfort Panels, RT Advanced, RT Professional)

Description

Returns a user-defined text through the operating system channel for debug alarms.

The methods HMIRuntime.Trace works only in a PC-based environment. The text transferred as parameter can be displayed using the diagnostics tools "GSC Diagnostics" or "ApDiag". You can use the "ShowSystemAlarm" system function if you need to run a trace without using external tools.

Note

"ApDiag" diagnostics tool

The "ApDiag" diagnostics tool is installed by default with Runtime. The diagnostics tool is available at the following path: "C:\Program Files (x86)\Siemens\Automation\SCADA-RT_V11\WinCC\Tools".

For more information, refer to the Internet (<https://support.industry.siemens.com/cs/de/en/view/102777629>).

Syntax

```
Expression.Trace"STRING"
```

Expression

Required An expression which returns an "HMIRuntime" type object.

Parameters

STRING

The text which is issued as a debug alarm. The transferred text can be displayed using the diagnostics tools "GSC Diagnostics" or "ApDiag". You can use the "ShowSystemAlarm" system function if you need to run a trace without using external tools.

Example

In the following example a debug alarm is issued:

```
'VBS example trace  
HMIRuntime.Trace "Customized error message"
```

See also

HMIRuntime (Page 208)

UnhideAlarm (RT Professional)

Description

Executes the "Unhide alarm" button function of the alarm view.

Syntax

```
Expression.UnhideAlarm()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

UnlockAlarm (RT Professional)

Description

Executes the "Unlock alarm" button function of the alarm view.

Syntax

```
Expression.UnlockAlarm()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

AlarmControl (Page 236)

UnselectAll (RT Professional)**Description**

Removes all selections from the cells of a table-based control.

Syntax

```
Expression.UnselectAll()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AlarmControl (Page 236)

UnselectRow (RT Professional)**Description**

Removes the selections from a specific cell of a table-based control.

Syntax

```
Expression.UnselectRow(ByVal IRow As Long)
```

Expression

Required. An expression that returns an object of the "ScreenItem" type.

Parameters

Long

Parameters	Description
IRow	Number of rows whose selection will be removed.

See also

OnlineTableControl (Page 348)

TrendRulerControl (Page 444)

UserArchiveControl (Page 469)

AlarmControl (Page 236)

Write (RT Professional)

Description of Tag Object

Writes a value to a tag. The "LastError" property can be used to determine whether the call was successful.

If the value of the tag is set successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	Tag values set by the user (unchanged)
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0
LastError	0
ErrorDescription	" "

If the value of the tag is not set successfully, the properties of the tag object are assigned the following values:

Property	Allocation
Value	Tag values set by the user (unchanged)
Name	Tag name (unchanged)
QualityCode	Bad Out of Service
Timestamp	0

Property	Allocation
LastError	Write operation error codes
ErrorDescription	Error description on LastError

syntax

```
Expression.Write [Value],[Writemode]
```

Expression

Necessary. An expression which returns a tag object.

Parameters

The value to be written can be transferred directly to the method as a parameter. If the parameter is not specified, the value in the "Value" property is used. The "Writemode" option parameter can be used to select whether the tag value should be written synchronously or asynchronously. If the "Writemode" parameter is not used, writing is performed asynchronously as its default value.

During the writing process, no information is supplied on the status of the tags.

The "Value" property contains the value which was set before or during the writing operation, therefore it may not correspond to the real current value of the tag. If the data on the tag should be updated, use the Read method.

Parameters	Description
Value (optional)	The tag value is specified. The specified value overwrites the value in the "Value" property in the tag object. The tag value is not specified. The tag receives the current value from the "Value" property of the tag object.
Writemode (optional)	0 or empty: The tag value is written asynchronously. 0 is the default value. 1: The tag value is written synchronously.

On asynchronous writing, it is written immediately into the tag image. The user does not receive any feedback if the value has been written in the programmable controller, too.

In the case of synchronous writing (direct to the PLC), the writing operation actually occurs when the PLC is ready to operate. The user receives a check-back message if the writing operation was not successful.

Example

Asynchronous writing

```
'VBS104
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write
MsgBox objTag.Value
```

or

```
'VBS105
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5
MsgBox objTag.Value
```

Synchronous writing

```
'VBS106
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write ,1
MsgBox objTag.Value
```

or

```
'VBS107
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5, 1
MsgBox objTag.Value
```

Description of TagSet Object

The TagSet object offers the option of writing several tags in one call.

Functionality here is mostly identical with that of a tag object. In the following, only deviations thereof are described.

Expression

Necessary. An expression which returns an object of type "TagSet".

Parameters

In order to write different values, the "Value" property of individual tag objects must be set, and write must be called thereafter without the "Value" parameter. Since the write commands are grouped into one call, it results in improved performance compared to single calls.

In a TagSet object, it is not possible to pass on a value using the "Write" method. Individual values must be set using the "Value" property of the individual tag objects.

Example

The following example shows how tags are included in the TagSet list, how tag values are set and subsequently written.

```
'VBS173
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Wert1"
group.Add "Wert2"
group("Wert1").Value = 3
group("Wert2").Value = 9
group.Write
```

If you set the optional parameter "Writemode" equal to 1, the process tags are written synchronously (directly to AS).

```
group.Write 1
```

WriteTags (RT Professional)

Description

Executes the "Write tags" button function of the recipe view.

Syntax

```
Expression.WriteTags()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

UserArchiveControl (Page 469)

ZoomArea (RT Professional)**Description**

Executes the "Zoom area" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.ZoomArea()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

ZoomInOut (RT Professional)**Description**

Executes the "Zoom +/-" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.ZoomInOut()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

ZoomInOutTime (RT Professional)

Description

Executes the "Zoom time axis +/-" button function of the f(t) trend view.

Syntax

```
Expression.ZoomInOutTime()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTrendControl (Page 360)

ZoomInOutValues (RT Professional)

Description

Executes the "Zoom value axis +/-" button function of the f(t) trend view.

Syntax

```
Expression.ZoomInOutValues()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameter

--

See also

OnlineTrendControl (Page 360)

ZoomInOutX (RT Professional)

Description

Executes the "Zoom X axis +/-" button function of the f(x) trend view.

Syntax

```
Expression.ZoomInOutX()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

FunctionTrendControl (Page 309)

ZoomInOutY (RT Professional)

Description

Executes the "Zoom Y axis +/-" button function of the f(x) trend view.

Syntax

```
Expression.ZoomInOutY()
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

--

See also

FunctionTrendControl (Page 309)

ZoomMove (RT Professional)

Description

Executes the "Move trend area" button function of the f(t) and f(x) trend views.

Syntax

```
Expression.ZoomMove ( )
```

Expression

Required. An expression that returns an object of the type "ScreenItem".

Parameters

- -

See also

OnlineTrendControl (Page 360)

FunctionTrendControl (Page 309)

C scripting (RT Professional)

2.1 System functions (RT Professional)

2.1.1 ActivateNextScreen (RT Professional)

Description

WinCC saves the names of the screens opened by the user during runtime as well as the sequence in which these screens were opened.

Can only be used in C scripting.

The maximum size of the screen buffer is specified in the "Runtime settings > Screens > Screen buffer" editor.

The ActivateNextScreen system function now opens the screen that was opened before the last call of ActivatePreviousScreen.

Syntax

```
BOOL ActivateNextScreen();
```

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The ActivateNextScreen function is used in the following program code to call the next screen and to save the return value to the b_error tag.

The saved return value can be processed in the following code.

```
{
BOOL b_error;

//Open next screen
b_error = ActivateNextScreen();

if(b_error)
{
    // User defined code if
    // function succeeds without error
    ...
}
else
{
    // User defined code in case of error
    ...
}
...
}
```

2.1.2 ActivatePreviousScreen (RT Professional)

Description

WinCC saves the names of the screens opened by the user during runtime as well as the sequence in which these screens were opened.

The system function can be used in C scripting only.

The maximum size of the screen buffer is specified in the "Runtime settings > Screens > Screen buffer" editor.

The ActivatePreviousScreen system function now opens the screen which was open before the currently open screen.

Syntax

```
BOOL ActivatePreviousScreen();
```

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The ActivatePreviousScreen function is used in the following program code to call the previous screen and to save the return value to the b_error tag.

The saved return value can be processed in the following code.

```
{  
  BOOL b_error;  
  
  //Open previous screen  
  b_error = ActivatePreviousScreen();  
  
  if(b_error)  
  {  
    // User defined code if  
    // function succeeds without error  
    ...  
  }  
  else  
  {  
    // User defined code in case of error  
    ...  
  }  
  ...  
}
```

2.1.3 ActivateScreen (RT Professional)

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent area or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent area when the system function is called up, the permanent area maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The following program code activates "Screen_2" with the ActivateScreen function, for example, when you click a button.

```
Sub ActivateScreen_2()  
  
'User-defined code  
' i. e. when pressing a button  
  
ActivateScreen "Screen_2",0
```

2.1.4 ActivateScreenInScreenWindow (RT Professional)

Description

Performs a screen change to the specified screen in a specified screen window.

Use in the function list

ActivateScreenInScreenWindow (Screen name, Screen window, New screen name)

Use in user-defined functions

ActivateScreenInScreenWindow (Screen_name, Screen_window, New_screen_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Screen name

Name of the screen to be displayed in the screen window.

Screen window

Name of the screen window in which the new screen is to be displayed.

New screen name

Name of the new screen to be displayed in the screen window.

Example

The ActivateScreenInScreenWindow function is used in the following program code to activate the "Screen_2" screen when you click a button.

```
{  
// User defined code  
// i.e. when pressing a button  
ActivateScreenInScreenWindow (GetParentScreen(screenName),  
GetParentScreenWindow(screenName), "Screen_2");  
...  
}
```

2.1.5 ActivateStartScreen (RT Professional)

Description

Opens the configured start screen.

Can only be used in C scripting.

Syntax

BOOL ActivateStartScreen();

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The ActivateStartScreen function is used in the following program code to call the configured start screen and to save the return value to the b_error tag.

The saved return value can be processed in the following code.

```
{
BOOL b_error;

//Open start screen
b_error = ActivateStartScreen();

if(b_error)
{
    // User defined code if
    // function succeeds without error
    ...
}
else
{
    // User defined code in case of error
    ...
}
...
}
```

2.1.6 ActivateStoredScreen (RT Professional)

Description

Opens the screen saved with the StoreScreen system function.

Can only be used in C scripting.

Syntax

```
BOOL ActivateStoredScreen();
```

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The `ActivateStoredScreen` function is used in the following program code to call the stored screen and to save the return value to the `b_error` tag.

The saved return value can be processed in the following code.

```
{  
  BOOL b_error;  
  
  //Open stored screen  
  b_error = ActivateStoredScreen();  
  
  if(b_error)  
  {  
    // User defined code if  
    // function succeeds without error  
    ...  
  }  
  else  
  {  
    // User defined code in case of error  
    ...  
  }  
  ...  
}
```

See also

[StoreScreen](#) (Page 1486)

2.1.7 DecreaseTag (RT Professional)

Description

Subtracts the given value from the tag value.

$X = X - a$

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function for events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual tag value is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

Example

The following program code decrements the value at the varX tag by the value at the value tag. The value entered is saved to the old_value tag and output along with the new varX value.

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Decrease tag
DecreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function DecreaseTag: %i\r\n", old_value, varX);
...
}
```

2.1.8 GetLocalScreen (RT Professional)

Description

Returns a pointer to the screen name.

Can only be used in C scripting.

Syntax

```
char* GetLocalScreen(char* Screen name);
```

Parameters

Screen name

Pointer to the name of the screen.

Return value

Pointer to the name of the screen.

Note

The syntax of the transferred call parameter Screen name must correspond to that formed by the graphics system for the screen paths:

<Screen_name>.<Screen_window_name>.<Screen_name>.<Screen_window_name>.<Screen_name>...

Principle

A "Screen_A" contains a "Screen_window_B". Another screen is called in the screen window and visualized as minimized image, etc.

Calling screens in the screen windows

1. "Screen_window_B" displays "Screen_C".
"Screen_C" contains "Screen_window_D".
2. "Screen_window_D" displays "Screen_E".

Pointer to the screen name

returns system function `GetLocalScreen(Screen name)` in the next step with the following values:

- "Screen_E", if the system function is called in "Screen_E"
- "Screen_C", if the system function is called in "Screen_C"
- "Screen_A", if the system function is called in "Screen_A"

Example

The following program code saves the return value of the function GetLocalScreen to the pszScrName tag. If the return value is valid (not NULL), it is saved with maximum_MAX_PATH characters to the szScrName tag.

```
{
char* pszScrName = NULL;
char szScrName[_MAX_PATH+1];

//Get the Local Screen
pszScrName = GetLocalScreen(lpszScreenName);

//Copy the string
if (pszScrName != NULL)
{
    strncpy(szScrName,pszScrName,_MAX_PATH);
// print local screen name
    printf ("Local screen name: %s\r\n", szScrName);
}
...
}
```

2.1.9 GetLinkedTag (RT Professional)

Description

Returns the name of the tag which is logically linked to the specified object property.

Can only be used in C scripting.

Syntax

char* GetLinkedTag(char* Screen Name, char* Object, char* Name of Property);

Parameters

Screen Name

Pointer to the name of the screen.

Object

Pointer to the name of the object.

Name of Property

Pointer to the name of the object property.

Return value

Pointer to the name of the tag which is logically linked to the specified object property.

Note

Return value for user data types on the faceplate

If a user data type is directly interconnected on the faceplate, no tag name is returned. A tag name is only returned if the structure elements of the user data type are interconnected individually on the faceplate.

Example

The following program code saves the return value of function "GetLinkedTag" to the "pszVarName" tag. If the return value is valid, it is saved with the maximum path length to the szVAarName tag. The saved return value can be processed in a following code.

```
char CFunktion()
{
    char* pszVarName = NULL;
    char szVarName[_MAX_PATH+1]; //Get the TagName
    pszVarName = GetLinkedVariable("gs_stand_graph_00","StatischerText6","Visible"); //Copy
    the string
    if (strcmp (pszVarName,"") != 0)
    {
        strncpy(szVarName,pszVarName,_MAX_PATH);
    }
    else printf("Das Attribut 'visible' ist nicht dynamisiert\r\n");
    //User defined code where the
    //user can do something with the returnvalue
    ...
}
```

2.1.10 GetLanguageByLocaleID (RT Professional)

Description

Determines the current Runtime language.

Can only be used in C scripting.

Syntax

DWORD GetLanguageByLocaleID ();

Return value

Language ID.

The following assignments apply (hexadecimal language code):

Symbolic name	Value(hexadecimal)	Abbreviation
LANG_ARABIC	0x0401	
LANG_AFRIKAANS	0x0436	
LANG_ALBANIAN	0x041C	
LANG_BASQUE	0x042D	
LANG_BULGARIAN	0x0402	
LANG_BYELORUSSIAN	0x0423	
LANG_CATALAN	0x0403	
LANG_CHINESE	0x0804	
LANG_CROATIAN	0x041A	
LANG_CZECH	0x0405	CSY
LANG_DANISH	0x0406	DAN
LANG_DUTCH	0x0413	NLD
LANG_ENGLISH	0x0409	ENU
LANG_ESTONIAN	0x0425	
LANG_FAEROESE	0x0438	
LANG_FARSI	0x0429	
LANG_FINNISH	0x040B	FIN
LANG_FRENCH	0x040C	FRA
LANG_GERMAN	0x0407	DEU
LANG_GREEK	0x0408	
LANG_HEBREW	0x040D	
LANG_HUNGARIAN	0x040E	HUN
LANG_ICELANDIC	0x040F	ISL
LANG_INDONESIAN	0x0421	
LANG_ITALIAN	0x0410	ITA
LANG_JAPANESE	0x0411	
LANG_KOREAN	0x0412	
LANG_LATVIAN	0x0426	
LANG_LITHUANIAN	0x0427	
LANG_NORWEGIAN	0x0414	NOR
LANG_POLISH	0x0415	PLK
LANG_PORTUGUESE	0x0416	PTB
LANG_ROMANIAN	0x0418	
LANG_RUSSIAN	0x0419	RUS
LANG_SLOVAK	0x041B	SKY
LANG_SLOVENIAN	0x0424	
LANG_SORBIAN	0x042E	
LANG_SPANISH	0x040A	ESP

Symbolic name	Value(hexadecimal)	Abbreviation
LANG_SWEDISH	0x041D	SVE
LANG_THAI	0x041E	
LANG_TURKISH	0x041F	TRK
LANG_UKRAINIAN	0x0422	

Example

The following program code reads out the current Runtime language and saves the return value in the `rt_language` tag.

The saved return value can be processed in the following code (in this case, with `printf` output).

```
{
DWORD rt_language;

//Get the current language
rt_language = GetLanguageByLocaleID ();

//print language code
printf ("Language code: %d\r\n", rt_language);
...
}
```

2.1.11 GetParentScreen (RT Professional)

Description

The function determines the name of the parent screen from a specified screen path.

Can only be used in C scripting.

Syntax

```
char* GetParentScreen(char* Screen name);
```

Parameters

Screen name

Pointer to the name of the screen. The syntax of the transferred call parameter Screen name must correspond to that formed by the graphics system for the screen paths:

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

Note

The "." character is used to distinguish between screen and screen object names. The ":" character is used for the hierarchy structuring. Therefore, use only the "-" or "_" as a delimiter in the name designations.

Return value

Name of the parent screen.

Principle

A "Screenwindow_1" screen window is in a "Screen_1" screen. In the screen window, the screen becomes "Screen_2" with the "Screenwindow_2" screen window, etc.

Screen path: Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen returns:

- "Screen_2", if the system function is called in the "Screen_3" screen.
- "Screen_1", if the system function is called in the "Screen_2" screen.

GetParentScreenwindow returns:

- "Screenwindow_2", if the system function is called in the "Screen_3" screen.
- "Screenwindow_1", if the system function is called in the "Screen_2" screen.

Example

The following program code saves the return value of function `GetParentScreen` to the `pszScrName` tag. If the return value is valid (not `NULL`), it is saved with maximum `_MAX_PATH` characters to the `szScrName` tag.

```
{
char* pszScrName = NULL;
char szScrName[_MAX_PATH+1];

//Get the Parent Screen
pszScrName = GetParentScreen(lpszScreenName);

//Copy the string
if (pszScrName != NULL)
{
    strncpy(szScrName,pszScrName,_MAX_PATH);
// print Screen name
    printf ("Screen name: %s\r\n", szScrName);
}
...
}
```

2.1.12 `GetParentScreenWindow` (RT Professional)

Description

The function determines the name of the parent screen window from a transferred screen path.
Can only be used in C scripting.

Syntax

```
char* GetParentScreenWindow(char* Screen name);
```

Parameters

Screen name

Pointer to the name of the screen. The syntax of the transferred Screen name call parameter must correspond to that formed by the graphics system for the screen paths:

<Screen name>.<screen window name>:<screen name>.<screen window name>:<screen name>...The "." character is used to distinguish between screen and screen object names.

The ":" character is used for the hierarchy structuring. Therefore, use only the "-" or "_" as a delimiter in the name designations.

Note

The "." character is used to distinguish between screen and screen object names. The ":" character is used for the hierarchy structuring. Therefore, use only the "-" or "_" as a delimiter in the name designations.

Return value

Name of the parent screen window.

Principle

A "Screenwindow_1" screen window is in a "Screen_1" screen. In the screen window, the screen becomes "Screen_2" with the "Screenwindow_2" screen window, etc.

Screen path: Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen returns:

- "Screen_2", if the system function is called in the "Screen_3" screen.
- "Screen_1", if the system function is called in the "Screen_2" screen.

GetParentScreenwindow returns:

- "Screenwindow_2", if the system function is called in the "Screen_3" screen.
- "Screenwindow_1", if the system function is called in the "Screen_2" screen.

Example

The following program code saves the return value of the function `GetParentScreenWindow` to the `pszScrName` tag. If the return value is valid (not `NULL`), it is saved with maximum `_MAX_PATH` characters to the `szScrName` tag.

```
{
char* pszScrName = NULL;
char szScrName[_MAX_PATH+1];

//Get the Parent Screen Window
pszScrName = GetParentScreenWindow(lpszScreenName);

//Copy the string
if (pszScrName != NULL)
{
    strncpy(szScrName,pszScrName,_MAX_PATH);
// print name of the parent screen window
    printf ("Parent screen window: %s\r\n", szScrName);
}
...
}
```

2.1.13 GetProp (RT Professional)

2.1.13.1 GetPropBOOL (RT Professional)

Description

Returns the current status of a property of data type "BOOL".

Can only be used in C scripting.

Syntax

`BOOL GetPropBOOL(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property)`

Parameters

Screen name

Screen name

Object

Object name You must set the parameter `Object = NULL` if the function call relates to a screen object property.

Name of the Property

Name of the object property

Return value**TRUE**

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code reads out whether the object is visible or not. The value is saved to the b_error tag.

```
{
BOOL b_error;

//Get the property Visible
b_error = GetPropBOOL("gs_graph_iofield","IOField1","Visible");

if(b_error)
{
    // User defined code if the
    // object is visible
    ...
}
else
{
    // User defined code if the
    // object is not visible
    ...
}
}
```

A specific code is processed, depending on the return value.

2.1.13.2 GetPropChar (RT Professional)**Description**

Returns the value of a property of data type "Char".

Can only be used in C scripting.

Syntax

Char* GetPropChar(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property)

Parameters

Screen name

Screen name

Object

Object name You must set the parameter Object = NULL if the function call relates to a screen object property.

Name of the Property

Name of the object property

Return value

Value of the property in data type "Char".

Example

The following program code reads the tooltip text from the object using the GetPropChar function and processes it as follows:

1. Saves the return value in the pszProp tag
2. Validation of the return value: Step 3 follows if the value is valid (not NULL).
3. The first 13 characters of the character sequence are saved in the szProp tag.

```
{
char* pszProp = NULL;
char szProp[14];

//Get the property Tooltiptext
pszProp = GetPropChar("lpszScreenName","IOField1","Tooltiptext");

if(pszProp != NULL)
{
//Copy the string and trim
strncpy(szProp,pszProp,13);
// print trimmed string
printf ("Short description of tooltip: %s\r\n", szProp);
}
...
}
```

The saved return value can be processed in the following code.

2.1.13.3 **GetPropDouble (RT Professional)**

Description

Returns the value of a property of data type "Double".
Can only be used in C scripting.

Syntax

double GetPropDouble(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property)

Parameters

Screen name

Screen name

Object

Object name You must set the parameter Object = NULL if the function call relates to a screen object property.

Name of the Property

Name of the object property

Return value

Value of the property in data type "Double".

Example

The following program code reads the "BackColor" property (background color of the "Button_1" object) by means of GetPropDouble function and processes the return value as follows:

1. Saving the return value to the szprop tag
2. Validation of the return value: Step 3 follows if the value is valid (not NULL).
3. Output of the background color

```
{
double szProp = NULL;

//Get the property of the IO Field 1
szprop = GetPropDouble("Screen_1","Button_1","BackColor");

if(szProp != NULL)
{
// print output value
printf ("Background color: %s\r\n", szProp);}
...
}
```

2.1.13.4 GetPropLong (RT Professional)

Description

Returns the status of a property of data type "long".
Can only be used in C scripting.

Syntax

long GetPropLong(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property)

Parameters

Screen name

Screen name

Object

Object name You must set the parameter Object = NULL if the function call relates to a screen object property.

Name of the Property

Name of the object property

Return value

Value of the property in data type "long".

Example

The following program code reads the CaptionBackColor property (background color of the TemperatureField object) by means of GetPropLong function and processes the return value as follows:

1. Saving the return value to the szProp tag
2. Validation of the return value: Step 3 follows if the value is valid (not NULL).
3. Output

```
{
long szProp = NULL;

//Get the property Tooltiptext
szProp = GetPropLong("Screen_1", "TemperatureField", "CaptionBackColor");

if(szProp != NULL)
{
// print caption
printf ("Caption of window: %d\r\n", szProp);
}
...
}
```

The saved return value can be processed in the following code.

2.1.14 GetServerTagPrefix (RT Professional)

Description

In order for a WinCC client in a distributed system to access tags of the associated server, the tag names must be expanded to include the server prefix.

Note

This system function is not supported at present.

A pointer of type "char" to server prefix, tag prefix, and window prefix is returned in each case.

The user is not permitted to change the memory (including no strcat) or release it.

Can only be used in C scripting.

Syntax

```
void GetServerTagPrefix (char** ppszServerPrefix, char** ppszTagPrefix, char**
ppszWindowPrefix);
```

Return value

ppszServerPrefix

Pointer to a pointer that references the server prefix

ppszTagPrefix

Pointer to a pointer that references the tag prefix

ppszWindowPrefix

Pointer to a pointer that references the window prefix

Example

The following program code retrieves the **server prefix**, **tag prefix** and **window prefix** checks their validity. If an error occurs, a text is output and the function is exited. If the check is successful a tag name is created and returned. Processing is executed as follows:

1. Declaration of pointer ppszServerPrefix, ppszTagPrefix and ppszWindowPrefix for the three prefixes
2. Initialization of the nServerPrefixLen, nTagPrefixLen and nTagLen tags
They serve as a buffer for the string length of the prefixes to be read out.
3. Initialization of the myTagName tag
4. Reading out of server prefix, tag prefix and window prefix
5. **Case distinction:** Server prefix
 - **No** server prefix returned: a text is output and the function is exited.
 - Server prefix returned: Its length is determined and saved in the nServerPrefixLen tag.
6. If a tag prefix is returned, its length is determined and saved in the nTagPrefixLen tag.
7. Determines the length of the tag name and saves it in the nTagLen tag.
8. **Case distinction:** Permissible length for tag name
 - Permissible length exceeded: a text is output and the function is exited.
 - Permissible length **not** exceeded: The tag name required for a client environment is compiled.

```

{
char* pszServerPrefix;
char* pszTagPrefix;
char* pszWindowPrefix;
int nServerPrefixLen = 0;
int nTagPrefixLen = 0;
int nTagLen = 0;
char myTagName[MAX_DM_VAR_NAME+1];

//Initialize the return value
memset(myTagName,0,MAX_DM_VAR_NAME + 1);

//Get the serverprefix the tagprefix and the windowprefix
GetServerTagPrefix(&pszServerPrefix, &pszTagPrefix, &pszWindowPrefix);

//If a serverprefix exists
if (pszServerPrefix)
{
    //Get the length of the string
    nServerPrefixLen = strlen(pszServerPrefix);
}
Else
{
    printf("No server prefix was returned.");
    return;
}

//If a tagprefix exists
if (pszTagPrefix)
{
    //Get the length of the string
    nTagPrefixLen = strlen(pszTagPrefix);
}

//Get the length of the tag
nTagLen = strlen("TagName");

//Check if the lenght of the
//ServerPrefix+TagPrefix+VarName + the double points < MAX_DM_VAR_NAME)
if (nServerPrefixLen + nTagPrefixLen + nTagLen+2 < MAX_DM_VAR_NAME)
{
    sprintf(myTagName,"%s::%s%s",pszServerPrefix,pszTagPrefix,"TagName");
    //User defined code where the
    //user can do something with the returnvalue
    ...
}
Else
{
    printf("The resulting string is too long.");
    return;
}
}

```

2.1.15 GetTag (RT Professional)

2.1.15.1 GetTag functions (RT Professional)

Description

The GetTagXXX function calculates the value of a tag of the specified data type.

Can only be used in C scripting:

The following table shows the different GetTag functions for reading the tag value:

Type	Function name	Parameter	PLC data type	HMI data type
BOOL	GetTagBit	Tag Tag_Name	Binary tag	Bool
BYTE	GetTagByte	Tag Tag_Name	Unsigned 8-bit	USInt
char*	GetTagChar	Tag Tag_Name	Text tag 8-bit or text tag 16-bit	String or WString
SYSTEM-TIME	GetTagDateTime	Tag Tag_Name	DTL	DateTime
double	GetTagDouble	Tag Tag_Name	Floating point 64-bit	LReal
DWORD	GetTagDWord	Tag Tag_Name	Unsigned 32-bit	UDInt
float	GetTagFloat	Tag Tag_Name	Floating point 32-bit	Real
BOOL	GetTagRaw	Tag Tag_Name, BYTE* pValue, DWORD size	Raw data type	Raw
signed char	GetTagSByte	Tag Tag_Name	Signed 8-bit	SInt
long int	GetTagSDWord	Tag Tag_Name	Signed 32-bit	DInt
short int	GetTagSWord	Tag Tag_Name	Signed 16-bit	Int
WORD	GetTagWord	Tag Tag_Name	Unsigned 16-bit	UInt

Syntax

```
<Type><FunctionName><(Parameter)>;
```

Example: BYTE GetTagByte (Tag Tag_Name);

Parameter

Tag_Name

Tag name

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

Value of the tags in the specified type.

System function "GetTagChar" returns a pointer to a string that contains the tag value.

System function "GetTagRaw" returns TRUE or FALSE:

TRUE: System function completed without errors.

FALSE: An error has occurred.

Example

The following program code uses the GetTagByte function to read out the value of the gs_tag_byte tag and saves it to the bvalue tag.

```
{  
BYTE bvalue;  
  
//Get the current value of the tag  
bvalue = GetTagByte("gs_tag_byte");  
  
// print value  
printf ("Value of gs_tag_byte: %d\r\n", bvalue);  
...  
}
```

The saved return value can be processed in the following code.

2.1.15.2 GetTagDateTime function (RT Professional)

Function

Determines the value of a tag of data type "Date/Time".

Syntax

SYSTEMTIME GetTagDateTime(Tag Tag_Name);

Parameter

Tag_Name

Name of the tag

Return value

Value of the tag in the data type "Date/Time".

2.1.15.3 GetTagMultiStateQCWait functions (RT Professional)

Description

The values, status, and quality code of several tags are established and saved in the respective addresses in the specified format. The values are read explicitly from the AS.

Two DWORD arrays are transferred, in the member of which the status and quality codes of the individual tags will be located after the system function has been called. The size of the arrays must be large enough to provide sufficient storage space for the status and quality codes.

Can only be used in C scripting.

Syntax

```
BOOL GetTagMultiStateQCWait (DWORD* pdwState, DWORD* pdwQualityCode, const char* pFormat, ...);
```

Parameters

pdwState

Field in which the status of the individual tags is created after the system function cycle.

pdwQualityCode

Field in which the quality code of the individual tag is created after the system function cycle.

pFormat

Format description (type) for all requested tags, followed by the name and address of the value for each tag.

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code uses the GetTagMultiStateQCWait function to read the value from the "gs_tag_XXX" tag and saves these values to the lvalue1, lvalue2 tags, etc.

1. Preprocessor definition for "DATA_SIZE" (in this case for 5 tags)
2. Creating the DWord fields
 - dwState Tag status field
 - dwQc Field for quality codes
3. Tag definition for buffering
4. Execution of the function GetTagMultiStateQCWait
The value read is written to the addresses of the tags.

```
{
#define DATA_SIZE 5
DWORD dwState[DATA_SIZE];
DWORD dwQC[DATA_SIZE];

//define all Datas
BOOL lValue1;
long lValue2 ;
char* szValue3;
double dblValue4 ;
WORD lValue5 ;

//Set the tags
GetTagMultiStateQCWait(dwState,dwQC,"%d%d%s%f%d",
  "gs_tag_bit",&lValue1,
  "gs_tag_SByte",&lValue2,
  "gs_tag_char",&szValue3,
  "gs_tag_float",&dblValue4,
  "gs_tag_word",&lValue5);

//User defined code where the
//user can do something with the returnvalue
...
}
```

The saved return value can be processed in the following code.

2.1.15.4 GetTagMultiStateWait functions (RT Professional)

Description

The values and status of several tags are determined and saved in the respective addresses in the specified format. The values are read explicitly from the AS.

The system function must be given a DWORD array, in the member of which the status of the individual tags will be located after the system function has been called. The size of the array must be large enough to provide sufficient storage space for the status.

Can only be used in C scripting.

Syntax

```
BOOL GetTagMultiStateWait(DWORD* pdwState, const char* pFormat);
```

Parameters

pdwState

Field in which the tag status is saved.

pFormat

Format description (type) for all requested tags, followed by the name and address of the value for each tag.

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code uses the GetTagMultiStateWait function to read the value from the "gs_tag_XXX" tag and saves these values to the lvalue1, lvalue2 tags, etc.

1. Preprocessor definition for "DATA_SIZE" (in this case for 5 tags)
2. Creating DWord field dwState for the tag status
3. Tag definition for buffering
4. Execution of the function GetTagMultiStateWait
The value read is written to the addresses of the tags.

```

{
#define DATA_SIZE 5
DWORD dwState[DATA_SIZE];

//define all Datas
BOOL lValue1;
long lValue2 ;
char* szValue3;
double dblValue4 ;
WORD lValue5 ;

//Set the tags
GetTagMultiStateWait(dwState,"%d%d%s%f%d",
    "gs_tag_bit",&lValue1,
    "gs_tag_SByte",&lValue2,
    "gs_tag_char",&szValue3,
    "gs_tag_float",&dblValue4,
    "gs_tag_word",&lValue5);

//User defined code where the
//user can do something with the returnvalue
...
}

```

The saved return value can be processed in the following code.

2.1.15.5 GetTagMultiWait functions (RT Professional)

Description

The values of several tags are determined and saved in the respective addresses in the specified format. The value is read explicitly from the AS. The system function uses SysMalloc to allocate memory to the tag value.

Can only be used in C scripting.

Syntax

```
BOOL GetTagMultiWait(const char* pFormat,...);
```

Parameters

pFormat

Format description for all requested tags along with name and address of the value for each tag.

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The GetTagMultiWait function is used in the following program code to read several tags of different types:

1. Declaration of three tags as memory for three different tag types
2. Declaration of the boolean tag ok for buffering the return value (TRUE/FALSE)
3. Reading the three tags and saving the values to the corresponding addresses.
The return value of the function is saved to the ok tag.
4. Output of the three tags with tag type prefix

```
DWORD dwVar1Value;  
char* szVar2Value;  
//memory for values allocated via SysMalloc  
  
double dbVar3Value;  
  
BOOL ok;  
  
ok=GetTagMultiWait("%d%s%f", "Ernie_word", &dwVar1Value,  
    "Ernie_char", &szVar2Value,  
    "Ernie_double", &dbVar3Value);  
  
printf("Word %d, String %s, Double %f\r\n",  
    dwVar1Value, szVar2Value, dbVar3Value);
```

2.1.15.6 GetTagState functions (RT Professional)

Description

The GetTagStateXXX function calculates the value of a tag of the specified data type. It also returns the status of the tags.

Can only be used in C scripting.

The following table lists the different GetTagStateXXX functions for reading the tag value:

Type	Function name	Parameters	PLC data type	HMI data type
BOOL	GetTagBitState	Tag Tag_Name, PDWORD lp_dwstate	Binary tag	Bool
BYTE	GetTagByteState	Tag Tag_Name, PDWORD lp_dwstate	Unsigned 8-bit	Byte
char*	GetTagCharState	Tag Tag_Name, PDWORD lp_dwstate	Text tag 8-bit or text tag 16-bit	String
double	GetTagDoubleState	Tag Tag_Name, PDWORD lp_dwstate	Floating point 64-bit	Double
DWORD	GetTagDWord-State	Tag Tag_Name, PDWORD lp_dwstate	Unsigned 32-bit	UInteger
float	GetTagFloatState	Tag Tag_Name, PDWORD lp_dwstate	Floating point 32-bit	Float
BOOL	GetTagRawState	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	Raw data type	Raw
signed char	GetTagSByteState	Tag Tag_Name, PDWORD lp_dwstate	Signed 8-bit	Byte
long int	GetTagSDWord-State	Tag Tag_Name, PDWORD lp_dwstate	Signed 32-bit	Integer
short int	GetTagSWord-State	Tag Tag_Name, PDWORD lp_dwstate	Signed 16-bit	Short
WORD	GetTagWordState	Tag Tag_Name, PDWORD lp_dwstate	Unsigned 16-bit	UShort

Syntax

<Type><FunctionName><(Parameter)>;

Example: BOOL GetTagBitState (Tag Tag_Name, PDWORD lp_dwstate);

Parameters

Tag_Name

Tag name

lp_dwstate

Pointer to a DWORD to which the tag status is saved on completion of the system function cycle.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

Value of the tags in the specified type.

The system function "GetTagCharState" returns a pointer to a value of the tag of data type "char".

System function "GetTagRawState" returns TRUE or FALSE:

TRUE: System function completed without errors.

FALSE: An error has occurred.

Example

The following program code uses the GetTagBitState function to read out the value of the gs_tag_bit tag and saves it to the bValue tag.

The status is stored in the address of the dwState tag.

A specific code will be executed, depending on the return value in bValue (TRUE/FALSE).

```
{
DWORD dwState;
BOOL bValue;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
bValue = GetTagBitState("gs_tag_bit",&dwState);

//Create a string which includes the tag value
if (bValue)
{
    // User defined code if the
    // value of the tag is true
    ...
}
else
{
    // User defined code if the
    // value of the tag is false
    ...
}
}
```

See also

Constants (Page 1554)

Structure definition DM_VAR_UPDATE_STRUCT (Page 1538)

Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1539)

2.1.15.7 GetTagStateQC functions (RT Professional)

Description

The GetTagStateQC function calculates the value of a tag of the specified data type. It also returns the status and quality code of the tags.

Can only be used in C scripting.

The following table shows the different GetTagStateQC functions for reading the tag value:

Type	Function name	Parameters	PLC data type	HMI data type
BOOL	GetTagBitStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Binary tag	Bool
BYTE	GetTagByteStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Unsigned 8-bit	UByte
char*	GetTagCharStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Text tag 8-bit or text tag 16-bit	String
double	GetTagDoubleStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Floating point 64-bit	Double
DWORD	GetTagDWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Unsigned 32-bit	UInteger
float	GetTagFloatStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Floating point 32-bit	Float
BOOL	GetTagRawStateQC	Tag Tag_Name, BYTE pValue[], DWORD size, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Raw data type	Raw
signed char	GetTagSByteStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Signed 8-bit	Byte
long int	GetTagSDWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Signed 32-bit	Integer

Type	Function name	Parameters	PLC data type	HMI data type
short int	GetTagSWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Signed 16-bit	Short
WORD	GetTagWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Unsigned 16-bit	UShort

Syntax

<Type><FunctionName><(Parameter)>;

Example: BOOL GetTagBitStateQC (Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name

Tag name

lp_dwstate

Pointer to a DWORD to which the tag status is saved on completion of the system function cycle.

pdwQualityCode

Pointer on a DWORD in which the quality code of the tag is created after the system function cycle.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

Value of the tags in the specified type.

The system function "GetTagCharStateQC" returns a pointer to a value of the tag of data type "char".

The system function "GetTagRawStateQC" returns TRUE or FALSE:

TRUE: System function completed without errors.

FALSE: An error has occurred.

Example

The following program code uses the GetTagBitStateQC function to read the value from the gs_tag_bit tag and saves it to the ok tag.

The status and quality code is saved to the dwState and dwQC addresses of the tag.

The return value of the ok tag (TRUE/FALSE) determines the specific code that is executed.

```
{
DWORD dwState;
DWORD dwQC;
BOOL ok;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
ok = GetTagBitStateQC("gs_tag_bit",&dwState,&dwQC);

//Create a string which includes the tag value
if (ok)
{
    // succeeded, print tag value
    printf ("Value at &dwState,&dwQC: %d\r\n", &dwState);
    printf ("Value at &dwQC: %d\r\n", &dwQC);
    ...
}
else
{
    // failed
    printf ( "Error - function failed." );
}
}
```

See also

Structure definition DM_VAR_UPDATE_STRUCT (Page 1538)

Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1539)

2.1.15.8 GetTagStateQCWait functions (RT Professional)**Description**

The GetTagStateQCWait function calculates the value of a tag of the specified data type. The value is read explicitly from the AS. It also returns the status and quality code of the tags.

Can only be used in C scripting.

The following table lists the different GetTagStateQCWait functions for reading the tag value:

Type	Function name	Parameters	PLC data type	HMI data type
BOOL	GetTagBitStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Binary tag	Bool
BYTE	GetTagByteStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Unsigned 8-bit	UByte
char*	GetTagCharStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Text tag 8-bit or text tag 16-bit	String
double	GetTagDoubleStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Floating point 64-bit	Double
DWORD	GetTagDWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Unsigned 32-bit	UInteger
float	GetTagFloatStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Floating point 32-bit	Float
BOOL	GetTagRawStateQCWait	Tag Tag_Name, BYTE pValue[], DWORD size, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Raw data type	Raw
signed char	GetTagSByteStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Signed 8-bit	Byte
long int	GetTagSDWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Signed 32-bit	Integer
short int	GetTagSWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Signed 16-bit	Short
WORD	GetTagWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQuality-Code	Unsigned 16-bit	UShort

Syntax

<Type><FunctionName><(Parameter)>;

Example: BOOL GetTagBitStateQC (Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode);

Parameters

Tag_Name

Tag name

lp_dwstate

Pointer to a DWORD to which the tag status is saved on completion of the system function cycle.

pdwQualityCode

Pointer on a DWORD in which the quality code of the tag is created after the system function cycle.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

Value of the tags in the specified type.

The system function "GetTagCharStateQCWait" returns a pointer to a value of the tag of data type "char".

The system function "GetTagRawStateQCWait" returns TRUE or FALSE:

TRUE: System function completed without errors.

FALSE: An error has occurred.

Example

The following program code uses the GetTagBitStateQCWait function to read the value from the gs_tag_bit tag and saves it to the bValue tag.

The status and quality code is saved to the dwState and dwQC addresses of the tag.

A specific code will be executed, depending on the return value in bValue (TRUE/FALSE).

```
{
DWORD dwState;
DWORD dwQC;
BOOL bValue;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
bValue = GetTagBitStateQCWait("gs_tag_bit",&dwState,&dwQC);

//Create a string which includes the tag value
if (bValue)
{
    // User defined code if the
    // value of the tag is true
    ...
}
else
{
    // User defined code if the
    // value of the tag is false
    ...
}
}
```

See also

Structure definition DM_VAR_UPDATE_STRUCT (Page 1538)

Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1539)

2.1.15.9 GetTagStateWait functions (RT Professional)

Description

The GetTagStateWait function calculates the value of a tag of the specified data type. The value is read explicitly from the AS. It also returns the status of the tags.

Can only be used in C scripting.

The following table lists the different GetTagStateWait functions for reading the tag value:

Type	Function name	Parameters	PLC data type	HMI data type
BOOL	GetTagBitStateWait	Tag Tag_Name, PDWORD lp_dwstate	Binary tag	Bool
BYTE	GetTagByteState- Wait	Tag Tag_Name, PDWORD lp_dwstate	Unsigned 8-bit	UByte

Type	Function name	Parameters	PLC data type	HMI data type
char*	GetTagCharStateWait	Tag Tag_Name, PDWORD lp_dwstate	Text tag 8-bit or text tag 16-bit	String
double	GetTagDoubleStateWait	Tag Tag_Name, PDWORD lp_dwstate	Floating point 64-bit	Double
DWORD	GetTagDWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	Unsigned 32-bit	UInteger
float	GetTagFloatStateWait	Tag Tag_Name, PDWORD lp_dwstate	Floating point 32-bit	Float
BOOL	GetTagRawStateWait	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	Raw data type	Raw
signed char	GetTagSByteStateWait	Tag Tag_Name, PDWORD lp_dwstate	Signed 8-bit	Byte
long int	GetTagSDWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	Signed 32-bit	Integer
short int	GetTagSWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	Signed 16-bit	Short
WORD	GetTagWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	Unsigned 16-bit	UShort

Syntax

<Type><FunctionName><(Parameter)>

Example: BOOL GetTagBitStateWait (Tag Tag_Name, PDWORD lp_dwstate)

Parameters

Tag_Name

Tag name

lp_dwstate

Pointer to a DWORD to which the tag status is saved on completion of the system function cycle.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

Value of the tags in the specified type.

The system function "GetTagCharStateWait" returns a pointer to a value of the tag of data type "char".

The system function "GetTagRawState" returns TRUE or FALSE:

TRUE: System function completed without errors.

FALSE: An error has occurred.

Example

The following program code uses the GetTagBitStateWait function to read the value from the gs_tag_bit tag and saves it to the bValue tag.

The status is stored in the address of the dwState tag.

A specific code will be executed, depending on the return value in bValue (TRUE/FALSE).

```
{
DWORD dwState;
BOOL bValue;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
bValue = GetTagBitStateWait("gs_tag_bit",&dwState);

//Create a string which includes the tag value
if (bValue)
{
    // User defined code if the
    // value of the tag is true
    ...
}
else
{
    // User defined code if the
    // value of the tag is false
    ...
}
}
```

See also

Structure definition DM_VAR_UPDATE_STRUCT (Page 1538)

Structure definition DM_VAR_UPDATE_STRUCTEX (Page 1539)

2.1.15.10 GetTagValue functions (RT Professional)

Description

Enables the transfer of a value in the form of a variant. Determines the pointer on the results structure that contains the value.

Can only be used in C scripting.

Syntax

```
BOOL GetTagValue(LPDM_VARKEY lpdmVarKey, LPDM_VAR_UPDATE_STRUCT  
lpdmresult, LPCMN_ERROR lpdmError);
```

Parameters

lpdmVarKey

Pointer to a structure of data type "DM_VARKEY"

lpdmresult

Pointer to a structure of data type "DM_VAR_UPDATE_STRUCT"

lpdmError

Pointer on the structure that contains the error description.

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The GetTagValue function is used in the following program code to calculate the value in varKey.

A specific code will be executed, depending on the return value in keyFound (TRUE/FALSE).

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error;

BOOL keyFound;

//Get the tag value
keyFound = GetTagValue(&varKey, &result, &error);

if (keyFound)
{
    // print tag value
    printf ("Value of varKey: %d\r\n", &varKey);
    ...
}
else
{
    // failed
    printf ( "Error - function failed." );
    ...
}
}
```

2.1.15.11 GetTagValueStateQC functions (RT Professional)

Description

Enables the transfer of a value in the form of a variant. Determines the pointer on the results structure that contains the value. It also returns the status and quality code of the tags.

The system function can be used in C scripting only.

Syntax

```
BOOL GetTagValueStateQC (LPDM_VARKEY lpdmVarKey,
LPDM_VAR_UPDATE_STRUCTEX lpdmresult, LPCMN_ERROR lpdmError);
```

Parameters

lpdmVarKey

Pointer to a structure of data type "DM_VARKEY"

lpdmresult

Pointer to a structure of data type "DM_VAR_UPDATE_STRUCTEX"

lpdmError

Pointer on the structure that contains the error description.

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The GetTagValueStateQC function is used in the following program code to calculate the value in varKey.

The status and quality code is saved to the dwState and dwQC addresses of the tag.

A specific code will be executed, depending on the return value in keyFound (TRUE/FALSE).

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCTEX result;
CMN_ERROR error;
// to clarify (DM_VAR_UPDATE_STRUCTEX already contains QualityCode)
DWORD dwState;
DWORD dwQC;

BOOL keyFound;

//Get the tag value
keyFound = GetTagValueStateQC(&varKey, &result, &error,&dwState,&dwQC);

if (keyFound)
{
    // User defined code if the
    // value of the tag is true
    ...
}
else
{
    // User defined code if the
    // value of the tag is false
    ...
}
}
```

2.1.15.12 GetTagValueStateQCWait functions (RT Professional)

Description

Enables the transfer of a value in the form of a variant. Determines the pointer on the results structure that contains the value. The value is read explicitly from the AS. It also returns the status and quality code of the tags.

Can only be used in C scripting.

Syntax

```
BOOL GetTagValueStateQCWait (LPDM_VARKEY lpdmVarKey,  
LPDM_VAR_UPDATE_STRUCTEX lpdmresult, LPCMN_ERROR lpdmError);
```

Parameters

lpdmVarKey

Pointer to a structure of data type "DM_VARKEY"

lpdmresult

Pointer to a structure of data type "DM_VAR_UPDATE_STRUCTEX"

lpdmError

Pointer on the structure that contains the error description.

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code uses the function for reading GetTagValueStateQCWait
The status and quality code is saved to the dwState and dwQC addresses of the tag.

A specific code will be executed, depending on the return value in keyFound (TRUE/FALSE).

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCTEX result;
CMN_ERROR error;
// to clarify (DM_VAR_UPDATE_STRUCTEX already contains QualityCode)
DWORD dwState;
DWORD dwQC;

BOOL keyFound;

//Get the tag value
keyFound = GetTagValueStateQC(&varKey, &result, &error,&dwState,&dwQC);

if (keyFound)
{
    // User defined code if the
    // value of the tag is true
    ...
}
else
{
    // User defined code if the
    // value of the tag is false
    ...
}
}
```

2.1.15.13 GetTagValueWait functions (RT Professional)

Description

Enables the transfer of a value in the form of a variant. Determines the pointer on the results structure that contains the value. The value is read directly from the AS.

Can only be used in C scripting.

Syntax

```
BOOL GetTagValueWait(LPDM_VARKEY lpdmVarKey, LPDM_VAR_UPDATE_STRUCT
lpdmresult, LPCMN_ERROR lpdmError);
```

Parameters

lpdmVarKey

Pointer to a structure of data type "DM_VARKEY"

lpdmresult

Pointer to a structure of data type "DM_VAR_UPDATE_STRUCT"

lpdmError

Pointer on the structure that contains the error description

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The GetTagValueWait function is used in the following program code to calculate the value in varKey.

A specific code will be executed, depending on the return value in keyFound (TRUE/FALSE).

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error;

BOOL keyFound;

//Get the tag value
keyFound = GetTagValueWait(&varKey, &result, &error);

if (keyFound)
{
    // succeeded, key found
    // print tag value
    printf ("Value of varKey: %d\r\n", &varKey);
    ...
}
else
{
    // failed
    printf ( "Error - function failed." );
}
    ...
}
```

2.1.15.14 GetTagWait functions (RT Professional)

Description

The GetTagWaitXXX function calculates the value of a tag of the specified data type. The value is read explicitly from the AS.

Can only be used in C scripting.

The following table shows the different GetTagWait functions for reading the tag value:

Type	Function name	Parameters	PLC data type	HMI data type
BOOL	GetTagBitWait	Tag Tag_Name	Binary tag	Bool
BYTE	GetTagByteWait	Tag Tag_Name	Unsigned 8-bit	UByte
char*	GetTagCharWait	Tag Tag_Name	Text tag 8-bit or text tag 16-bit	String
double	GetTagDoubleWait	Tag Tag_Name	Floating point 64-bit	Double
DWORD	GetTagDWordWait	Tag Tag_Name	Unsigned 32-bit	UInteger
float	GetTagFloatWait	Tag Tag_Name	Floating point 32-bit	Float
BOOL	GetTagRawWait	Tag Tag_Name, BYTE* pValue, DWORD size	Raw data type	Raw
char	GetTagSByteWait	Tag Tag_Name	Signed 8-bit	Byte
long int	GetTagSDWordWait	Tag Tag_Name	Signed 32-bit	Integer
short int	GetTagSWordWait	Tag Tag_Name	Signed 16-bit	Short
WORD	GetTagWordWait	Tag Tag_Name	Unsigned 16-bit	UShort

Syntax

<Type><FunctionName><(Parameter)>;

Example: BYTE GetTagByteWait (Tag Tag_Name);

Parameters

Tag_Name

Tag name

pValue

Pointer to a byte field containing the value of the raw data tags.

Return value

Value of the tags in the specified type.

System function "GetTagCharWait" returns a pointer to a string that contains the tag value.

System function "GetTagRawWait" returns TRUE or FALSE:

TRUE: System function completed without errors.

FALSE: An error has occurred.

Example

The following program code uses the GetTagByteWait function to read the value from the gs_tag_byte tag and saves it to the bvalue tag.

```
{
BYTE bvalue;

//Get the current value of the tag
bvalue = GetTagByteWait("gs_tag_byte");

// print value
printf ("Value of gs_tag_byte: %d\r\n", bvalue);
...
}
```

The saved return value can be processed in the following code.

2.1.16 IncreaseTag (RT Professional)

Description

Adds the given value to the value of the tags.

$X = X + a$

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function for events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual tag value is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag

The tag to which the given value is added.

Value

The value that is added.

Example

The following program code increments the value of the varX tag by the value in the value tag. The value entered is saved to the old_value tag and output along with the new varX value.

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Increase tag
IncreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function IncreaseTag: %i\r\n", old_value, varX);
...
}
```

2.1.17 InquireLanguage (RT Professional)

Description

Determines all languages configured in the text library for the runtime.

Use Pointer to a counter to specify the storage location for language IDs.

The system function can be used in C scripting only.

Syntax

DWORD* InquireLanguage (DWORD* Pointer to a counter);

Parameters

Pointer to a counter

Pointer to the number of language IDs found

Return value

Pointer to a field containing the established language IDs.

The following assignments apply (hexadecimal language code):

Symbolic name	Value(hexadecimal)	Abbreviation
LANG_ARABIC	0x0401	
LANG_AFRIKAANS	0x0436	
LANG_ALBANIAN	0x041C	
LANG_BASQUE	0x042D	
LANG_BULGARIAN	0x0402	
LANG_BYELORUSSIAN	0x0423	
LANG_CATALAN	0x0403	
LANG_CHINESE	0x0404	
LANG_CROATIAN	0x041A	
LANG_CZECH	0x0405	CSY
LANG_DANISH	0x0406	DAN
LANG_DUTCH	0x0413	NLD
LANG_ENGLISH	0x0409	ENU
LANG_ESTONIAN	0x0425	
LANG_FAEROESE	0x0438	
LANG_FARSI	0x0429	
LANG_FINNISH	0x040B	FIN
LANG_FRENCH	0x040C	FRA
LANG_GERMAN	0x0407	DEU
LANG_GREEK	0x0408	
LANG_HEBREW	0x040D	
LANG_HUNGARIAN	0x040E	HUN
LANG_ICELANDIC	0x040F	ISL
LANG_INDONESIAN	0x0421	
LANG_ITALIAN	0x0410	ITA
LANG_JAPANESE	0x0411	
LANG_KOREAN	0x0412	
LANG_LATVIAN	0x0426	
LANG_LITHUANIAN	0x0427	
LANG_NORWEGIAN	0x0414	NOR
LANG_POLISH	0x0415	PLK
LANG_PORTUGUESE	0x0416	PTB
LANG_ROMANIAN	0x0418	
LANG_RUSSIAN	0x0419	RUS
LANG_SLOVAK	0x041B	SKY
LANG_SLOVENIAN	0x0424	
LANG_SORBIAN	0x042E	
LANG_SPANISH	0x040A	ESP

Symbolic name	Value(hexadecimal)	Abbreviation
LANG_SWEDISH	0x041D	SVE
LANG_THAI	0x041E	
LANG_TURKISH	0x041F	TRK
LANG_UKRAINIAN	0x0422	

Example

The following program code uses the InquireLanguage function to query the language configured at runtime and processes these as follows:

1. Saving the queried language ID to the tag language
2. Saving the number of languages to the tag count
3. Formatted output of the ID and number of languages

```
{
DWORD count;
DWORD* language;
int i;

//Count the installed languages
language = InquireLanguage(&count);

printf("##### INQUIRE LANGUAGE #####");
//Print out the count of languages
printf ( "\r\nCount Languages=%d\r\n", count );

//print out which languages are installed
for (i=1;i<=count; i++)
{
printf ( "\r\n%d.language=%x", i,*language++);
}
}
```

The saved return value can be processed in the following code.

2.1.18 InverseLinearScaling (RT Professional)

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function $X = (Y - b) / a$.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InvertLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

The saved return value can be processed in the following code.

2.1.19 InvertBit (RT Professional)

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the Boolean tag bStatus and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
  BOOL bStatus = 0;
  BOOL bSaved = bStatus;

  //Invert variable
  invertBit(bStatus);

  //print current and saved value
  printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

2.1.20 InvertBitInTag (RT Professional)

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters**Tag**

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
    ...
}
```

2.1.21 IsUserAuthorized (RT Professional)**Description**

Authentication of users.

Can only be used in C scripting.

Syntax

BOOL IsUserAuthorized (DWORD AuthorizationNumber);

Parameters

AuthorizationNumber

Authorization (numerical) to be verified.

Return value

TRUE

The user has the specified authorization.

FALSE

The user does not have the specified authorization.

Example

The following program code authenticates users by means of IsUserAuthorized function and writes the value to the boolean tag ok.

```
{
BOOL ok;
DWORD authnumber;

//Check authorization and return value
ok = IsUserAuthorized (authnumber);

//error handling
if(ok)
{
    // user authorized
    printf ( "User is authorized." );
}
else
{
    // user not authorized
    printf ( "Authorization failed." );
}
...
}
```

The return value is output.

2.1.22 LinearScaling (RT Professional)

Description

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function $Y = (a * X) + b$.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n", Yvar);
...
}
```

The saved return value can be processed in the following code.

2.1.23 ReportJob (RT Professional)

Description

A print job, or the preview for a print job is started depending on the value of the Name of method parameter.

Can only be used in C scripting.

Syntax

void ReportJob(LPCSTR Print job name, LPCSTR Name of method)

Parameters

Print job name

Pointer to the name of the print job

Name of method

Defines whether to start the print job or a print job preview:

- PRINTJOB: Print job is started.
- PREVIEW: Preview of the print job is started.

Example

The following program code executes a preview or print job for each content of the printmethod tag. The printmethod tag is read by an I/O field that is connected to the strPrintJobMethod tag.

```
{
char* pszPrintjobName;
char* printmethod;

//Get the print method
printf("Input PRINTJOB for printing or PREVIEW for a quick view");
printmethod = GetTagChar("strPrintJobMethod")

//Print job or show preview
ReportJob(&PrintjobName, printmethod);

//error handling
if(printmethod=="PRINTJOB")
{
    // message for printing completed
    printf("printing done");
    ...
}
else
{
    // User defined code if the
    // job is a preview or failed
    ...
}
}
```

2.1.24 ResetBit (RT Professional)

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code sets the value of the Boolean tag bStatus to 0 by means of the ResetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
  BOOL bStatus = 1;
  BOOL bSaved = bStatus;

  //Reset bit
  ResetBit (bStatus);

  //print current and saved value
  printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

2.1.25 ResetBitInTag (RT Professional)

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit to 0 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBit(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult

//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
...
}
```

2.1.26 Set_Focus (RT Professional)**Function**

Sets the focus on the specified object.

Syntax

BOOL Set_Focus(LPCTSTR IpszPictureName, LPCTSTR IpszObjectName)

Parameters

lpszPictureName

Screen name

lpszObjectName

Object name

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

2.1.27 SetBit (RT Professional)

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the Boolean tag bStatus to 1 by means of the SetBit function and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit(myTag)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult


//Programming language: C
{
  BOOL bStatus = 0;
  BOOL bSaved = bStatus;

  //Set bit
  SetBit (bStatus);

  //print current and saved value
  printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

2.1.28 SetBitInTag (RT Professional)**Description**

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bStatusWord tag and outputs the result along with the original bSaved value.

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=2
SetBitInTag(myTag, bitposition)
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult

//Programming language: C
{
BYTE bStatusWord;
BYTE bSaved = bStatusWord;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bStatusWord, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bSaved);
...
}
```

2.1.29 SetLanguageByLocaleID (RT Professional)**Description**

Changes the language setting in Runtime.

Can only be used in C scripting.

Syntax

BOOL SetLanguageByLocaleID (DWORD dwLocaleID);

Parameters

dwLocaleID

Language ID of the language to be set

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code uses the SetLanguage function to set the current runtime language to German and saves the return value to the ok tag.

```
{
BOOL ok;
DWORD old_language;
DWORD new_language;

//Get the current language and save it
old_language = GetLanguageByLocaleID ();

//Set the current language to German
ok = SetLanguage(0x0407);

//Get the current language and save it
new_language = GetLanguageByLocaleID ();

//error handling
if(ok)
{
    // succeeded
    printf ( "RT language is now German." );
}
else
{
    // failed
    printf ( "RT language was not updated." );
}
//print language code
printf ("Former language code: %d\r\n", old_language);
printf ("Current language code: %d\r\n", new_language);
}
```

The saved return value can be processed in the following code.

See also

GetLanguageByLocaleID (Page 1384)

2.1.30 SetProp (RT Professional)**2.1.30.1 SetPropBOOL (RT Professional)****Description**

Sets an object property of data type "BOOL".

Can only be used in C scripting.

Syntax

BOOL SetPropBOOL(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property, BOOL Value)

Parameters

Screen name

Screen name

Object

Object name You must set the parameter Object = NULL if the function call relates to a screen object property.

Name of the Property

Name of the object property

Value

Value assigned to the object property of data type "BOOL".

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code uses the SetPropBool function to set the property of the gs_graph_iofield object to "Visible". The return value is saved to the ok tag.

```
{
BOOL ok;

//Set the visibility TRUE
ok = SetPropBOOL("gs_graph_iofield","IOField1","Visible",TRUE);

//error handling
if(ok)
{
    // succeeded
    printf ( "IO field is visible." );
}
else
{
    // failed
    printf ( "Error - visibility not set" );
}
...
}
```

The saved return value can be processed in the following code.

2.1.30.2 SetPropChar (RT Professional)

Description

Sets an object property of data type "char".

Can only be used in C scripting.

Syntax

BOOL SetPropChar(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property, char* Value)

Parameters

Screen name

Screen name

Object

Object name You must set the parameter Object = NULL if the function call relates to a screen object property.

Name of the Property

Name of the object property

Value

Value assigned to the object property of data type "char".

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code uses the SetPropChar function to set the Tooltiptext property of the gs_graph_iofield object to the "Tooltiptext 1" value. The return value is saved to the ok tag.

```
{
BOOL ok;

//Set the property Tooltiptext
ok = SetPropChar("Screen_1","gs_graph_iofield","ToolTipText","Tooltiptext 1");

//error handling
if(ok)
{
    // succeeded
    printf ( "Property of Tooltiptext is now Tooltiptext 1." );
}
else
{
    // failed
    printf ( "Error - property not set" );
}
...
}
```

The saved return value can be processed in the following code.

2.1.30.3 SetPropDouble (RT Professional)

Description

Sets an object property of data type "Double".

Can only be used in C scripting.

Syntax

BOOL SetPropDouble(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property, double Value)

Parameters

Screen name

Screen name

Object

Object name You must set the parameter Object = NULL if the function call relates to a screen object property.

Name of the Property

Name of the object property

Value

Value assigned to the object property of data type "double".

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code accesses an object property in the screenName screen. In this example, the SetPropDouble function sets the property (radius) of "Circle_1" to the value 10. The return value is saved to the ok tag.

```
{
BOOL ok;

//Set the property of circle
ok = SetPropDouble(screenName,"Circle_1","Radius",10);

//error handling
if(ok)
{
    // succeeded
    printf ( "Radius was set." );
}
else
{
    // failed
    printf ( "Error - radius not set" );
}
...
}
```

The saved return value can be processed in the following code.

2.1.30.4 SetPropLong (RT Professional)

Description

Sets an object property of data type "long".

Can only be used in C scripting.

Syntax

BOOL SetPropLong(LPCTSTR Screen name, LPCTSTR Object, LPCTSTR Name of the property, long Value)

Parameters

Screen name

Screen name

Object

Object name You must set the parameter Object = NULL if the function call relates to a screen object property.

Name of the Property

Name of the object property

Value

Value assigned to the object property of data type "long".

Return value**TRUE**

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code uses the SetPropLong function to change the foreground color of an object: In "Screen_1", the "ForeColor" property of the "Button1" object is set to the value 65333 (red). The return value is saved to the ok tag.

```
{  
  BOOL ok;  
  
  //Set the property Tooltiptext  
  ok = SetPropLong("Screen_1", "Button1", "ForeColor", 65333);  
  
  //error handling  
  if(ok)  
  {  
    // succeeded  
    printf ( "Color was set." );  
  }  
  else  
  {  
    // failed  
    printf ( "Error - color not set" );  
  }  
  ...  
}
```

The saved return value can be processed in the following code.

2.1.31 SetPropertyByConstant (RT Professional)

Description

Specifies the value of an object property as a string.

Use in the function list

SetPropertyByConstant (Screen name, screen object, name of the property, value)

Use in user-defined functions

SetPropertyByConstant

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to change the property of a screen, the parameter "Object" must be empty. For this purpose, use the following syntax, for example:

SetPropertyByConstant ("<Screen name>", "", "<Property name>", "<Value>")

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property is changed.

Name of the property

Name of the property that will be changed.

Value

The value assigned to the property.

Example

The SetPropertyByConstant function is used in the following program code to change an object property: In the "Trends" screen, the "ToolBarButtonClick" property of the "Control_1" object is set to the value 26.

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolBarButtonClick", "26"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolBarButtonClick", "26");

// User defined code
...
}
```

Example: Changing a screen property

The SetPropertyByConstant function is used in the following program code to change a screen property: In the "Trends" screen, the "Backcolor" property is set to the value 255.

```
'Programming language: VBS
'Name of the picture: Trends

SetPropertyByConstant "Trends", "Trends", "Backcolor", "255"

'User defined code
...


{
//Programming language: C
//Name of the picture: Trends

SetPropertyByConstant ("Trends", "Trends", "Backcolor", "255");

// User defined code
...
}
```

Alternatively, use the password ZERO or a space-string instead of the second parameter (object).

2.1.32 SetPropertyByProperty (RT Professional)

Description

Specifies the value of an object property with another object property.

Use in the function list

SetPropertyByProperty (Screen name, object, name of the property, target screen name, target object, target property name)

Use in user-defined functions

SetPropertyByProperty (Screen_name, Screen_object, Property_name, Source_screen_name, Source_screen_object, Source_property_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to specify the property of a screen with another screen property, the parameters "Object" and "Destination object" must be empty. For this purpose, use the following syntax, for example:

SetPropertyByProperty ("<Screen name>", "", "<Property name>", "<Destination_Screen name>", "", "<Destination_Property name>")

Parameters

Screen name

Name of the screen that contains the object.

Object

Name of the object whose property value is transferred to the target object.

Name of the property

Name of the property whose value is transferred to the target object.

Target screen name

Name of the screen that contains the destination object.

Target object

Name of the target object for which a property is changed.

Destination property name

Name of the property that will be changed.

Example

The following program code uses the SetPropertyByProperty function to set the property "ToolBarButtonClick" of the object "Control_1" in the original screen Trend_1": on the corresponding property in the destination screen "Trend_2".

```
'Programming language: VBS
'Name of source picture: Trend_1
'Name of target picture: Trend_2
'Name of the f(t) trend view control: Control_1

SetPropertyByProperty "Trend_1", "Control_1", "ToolBarButtonClick", "Trend_2",
"Control_2", "ToolBarButtonClick"

'User defined code
...

{
//Programming language: C
//Name of source picture: Trend_1
//Name of target picture: Trend_2
//Name of the f(t) trend view control: Control_1

SetPropertyByProperty ("Trend_1", "Control_1", "ToolBarButtonClick", "Trend_2",
"Control_2", "ToolBarButtonClick");

// User defined code
...
}
```

2.1.33 SetPropertyByTag (RT Professional)**Description**

Specifies the value of an object property with a tag value.

Use in the function list

SetPropertyByTag (Screen name, screen object, name of the property, tag name)

Use in user-defined functions

SetPropertyByTag (Screen_name, Screen_object, Property_name, Tag_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to specify the property of a screen by means of a tag value, the "Object" parameter must be empty. For this purpose, use the following syntax, for example:

SetPropertyByTag ("<Screen name>", "", "<Property name>", "<Tag name>")

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object containing the property to be set with the tag value.

Name of the property

Name of the property that is set with the tag value.

Tag name

Name of the tag that contains the value of the property.

Example

The SetPropertyByTag function is used in the following program code to change an object property: When you click on the object, the object name and the screen in which the object is located are transferred. The CaptionText in the screen window contains the value of the HMI_value_1 tag.

```
'Programming language: VBS
SetPropertyByTag screenName, objectName, "CaptionText", "HMI_value_1"

'User defined code
...

{
//Programming language: C
SetPropertyByTag (screenName, objectName, "CaptionText", "HMI_value_1");

// User defined code
...
}
```

Example

The SetPropertyByTag function is used in the following program code to change an object property: In the "Trends" screen, the "ToolBarButtonClick" property of the "Control_1" object is set to the value 26.

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolBarButtonClick", "26"

'User defined code
...

{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolBarButtonClick", "26");

// User defined code
...
}
```

2.1.34 SetPropertyByTagIndirect (RT Professional)

Description

Writes the value of an indirect addressed tag to an object property. The tag transferred as parameter contains the name of a second tag the value of which is read.

Use in the function list

SetPropertyByTagIndirect (Screen name, screen object, name of the property, , tag name)

Use in user-defined functions

SetPropertyByTagIndirect (Screen_name, Screen_object, Property_name, Tag_name)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

If you want to specify the property of a screen by means of a tag, the "Object" parameter must be empty. For this purpose, use the following syntax, for example:

SetPropertyByTagIndirect ("<Screen name>", "", "<Property name>", "<Tag name>")

Parameters

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property you want to change.

Name of the property

Name of the property to be changed.

Tag name

Name of the tag which, in turn, contains the name of the tag whose value is read.

Example

The SetPropertyByTagIndirect function is used in the following program code to change an object property: .

```
'Programming language: VBS
SetPropertyByTagIndirect GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1"

'User defined code
...

{
//Programming language: C
SetPropertyByTagIndirect (GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1");

// User defined code
...
}
```

2.1.35 SetTag (RT Professional)

2.1.35.1 SetTag functions (RT Professional)

Description

The SetTagXXX function calculates the value of a tag of the specified data type.

Can only be used in C scripting.

The following table shows the different SetTag functions for setting the tag value:

Function name	Parameter	PLC data type	HMI data type
SetTagBit	Tag Tag_Name, short int value	Binary tag	Bool
SetTagByte	Tag Tag_Name, BYTE value	Unsigned 8-bit	USInt
SetTagDateTime	Tag Tag_Name, SYSTEMTIME value	DTL	DateTime
SetTagChar	Tag Tag_Name, LPSTR value	Text tag 8-bit or text tag 16-bit	String or WString
SetTagDouble	Tag Tag_Name, double value	Floating point 64-bit	LReal
SetTagDWord	Tag Tag_Name, DWORD value	Unsigned 32-bit	UDInt
SetTagFloat	Tag Tag_Name, float value	Floating point 32-bit	Real
SetTagRaw	Tag Tag_Name, BYTE* pValue, DWORD size	Raw data type	Raw
SetTagSByte	Tag Tag_Name, char value	Signed 8-bit	SInt
SetTagSDWord	Tag Tag_Name, long int value	Signed 32-bit	DInt
SetTagSWord	Tag Tag_Name, short int value	Signed 16-bit	Int
SetTagWord	Tag Tag_Name, WORD value	Unsigned 16-bit	UInt

Note

Error message when using a tag of WString type

If you use a WString type tag in a SetTag system function that was converted into a C script, an error message is generated in the Global Script diagnostics window when the function is executed. The function is correctly executed despite the error message.

The following SetTag functions are affected:

- SetTagWithOperatorEvent
- SetTagIndirect
- SetTagByTagIndirect
- SetTagIndirectByTagIndirect
- SetTagByProperty
- SetTagIndirectByProperty

Syntax

```
BOOL<FunctionName><(Parameter)>;
```

Example: `BOOL SetTagBit (Tag Tag_Name, short int value);`

Parameter

Tag_Name

Tag name

value

Value of the tags in the specified data type.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

TRUE

System function completed without errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

Example

The following program code uses the SetTagBit function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTagBit("gs_tag_bit", TRUE);
//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );
    bvalue = GetTagBit("gs_tag_bit");
    printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

2.1.35.2 SetTagDateTime (RT Professional)

Function

Sets the value of a tag of data type "Date/Time".

Syntax

BOOL SetTagDateTime(Tag Tag_Name, SYSTEMTIME value);

Parameter

Tag_Name

Tag name

value

Value of the tag in the data type "Date/Time".

Return value

TRUE

The function itself has been completed without any errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

2.1.35.3 SetTagMultiStateWait functions (RT Professional)

Description

Sets the values of several tags. The system function will end only after the AS has reported the acceptance of the value.

The system function can be used in C scripting only.

The system function must be given a DWORD array, in the member of which the status of the individual tags will be located after the system function has been called. The size of the array must be large enough to provide sufficient storage space for the status.

Syntax

```
BOOL SetTagMultiStateWait(DWORD* pdwState, const char* pFormat,...);
```

Parameters

pdwState

Field in which the tag status is saved.

pFormat

Format description for all the requested tags along with the name and value for each tag.

Return value

TRUE

System function completed without errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

Example

The following program code uses the SetTagMultiStateWait function to set the value of several tags.

1. Creation of a DWord arrays of the necessary size (number of tags)
2. Creation of the tags that will contain the values to be transferred to the WinCC tags by means of the SetTagMultiStateWait function
3. Writing the values of the recently declared tags to the WinCC tags:
 - gs_tag_bit including the tag value "IValue1"
 - gs_tag_SByte including the tag value in the address "&IValue2"
 - etc.

```
{
#define DATA_SIZE 5
DWORD dwData[DATA_SIZE];

//define all tags
BOOL lValue1;
long lValue2;
char szValue3[_MAX_PATH];
float lValue4;
char lValue5;

// Fill the tags with the values
// you want to set into the WinCC tags
...

//Set the WinCC tags
SetTagMultiStateWait(dwData,"%d%d%s%f%d","gs_tag_bit",lValue1,
"gs_tag_SByte",lValue2,
"gs_tag_char",szValue3,
"gs_tag_float",lValue4,
"gs_tag_word",lValue5);
...
}
```

The saved return value can be processed in the following code.

2.1.35.4 SetTagMultiWait functions (RT Professional)

Description

The values of several tags are set in the specified format. The system function will end only after the AS has reported the acceptance of the value.

Can only be used in C scripting.

Syntax

```
BOOL SetTagMultiWait(const char* pFormat,...);
```

Parameters

pFormat

Format description for all the requested tags along with the name and value for each tag.

Return value

TRUE

System function completed without errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

Example

The following program code uses the SetTagMultiWait function to change the value of several tags. The return value is saved to the ok tag.

```
{
BOOL ok;
//memory for values allocated via SysMalloc
DWORD dwVar1Value;
char* szVar2Value;
double dbVar3Value;

//settings
ok=SetTagMultiWait("%d%s%f", "Ernie_word", 16,
    "Ernie_char", "Hallo Welt",
    "Ernie_double", 55.4711);

//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );

    // Get values and print
    GetTagMultiWait("%d%s%f", "Ernie_word", &dwVar1Value,
        "Ernie_char", &szVar2Value,
        "Ernie_double", &dbVar3Value);

    printf("Word %d, String %s, Double %f\r\n",
        dwVar1Value, szVar2Value, dbVar3Value);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

Example

The GetTagMultiWait function is used in the following program code to read several tags of different types:

1. Declaration of three tags as memory for three different tag types
2. Declaration of the boolean tag ok for buffering the return value (TRUE/FALSE)
3. Reading the three tags and saving the values to the corresponding addresses.
The return value of the function is saved to the ok tag.
4. Output of the three tags with tag type prefix

```

DWORD dwVar1Value;
char* szVar2Value;
//Speicher für den Variablenwert wird
//durch die Funktion mit SysMalloc angelegt
double dbVar3Value;

BOOL ok;

ok=GetTagMultiWait("%d%s%f", "Ernie_word", &dwVar1Value,
    "Ernie_char", &szVar2Value,
    "Ernie_double", &dbVar3Value);

printf("Word %d, String %s, Double %f\r\n",
    dwVar1Value, szVar2Value, dbVar3Value);

```

2.1.35.5 SetTagState functions (RT Professional)

Description

Sets the value of a tag of a specified data type. It also returns the status of the tags.

Can only be used in C scripting.

The following table lists the different SetTagStateXXX functions for reading the tag value:

Function name	Parameters	PLC data type	HMI data type
SetTagBitState	Tag Tag_Name, short int value, PDWORD lp_dwstate	Binary tag	Bool
SetTagByteState	Tag Tag_Name, BYTE value, PDWORD lp_dwstate	Unsigned 8-bit	UByte
SetTagCharState	Tag Tag_Name, LPSTR value, PDWORD lp_dwstate	Text tag 8-bit or text tag 16-bit	String
SetTagDoubleState	Tag Tag_Name, double value, PDWORD lp_dwstate	Floating point 64-bit	Double
SetTagDWordState	Tag Tag_Name, DWORD value, PDWORD lp_dwstate	Unsigned 32-bit	UInteger
SetTagFloatState	Tag Tag_Name, float value, PDWORD lp_dwstate	Floating point 32-bit	Float
SetTagRawState	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	Raw data type	Raw
SetTagSByteState	Tag Tag_Name, signed char value, PDWORD lp_dwstate	Signed 8-bit	Byte

Function name	Parameters	PLC data type	HMI data type
SetTagSDWord-State	Tag Tag_Name, long int value, PDWORD lp_dwstate	Signed 32-bit	Integer
SetTagSWord-State	Tag Tag_Name, short int value, PDWORD lp_dwstate	Signed 16-bit	Short
SetTagWord-State	Tag Tag_Name, WORD value, PDWORD lp_dwstate	Unsigned 16-bit	UShort

Syntax

BOOL <FunctionName><(Parameter)>;

Example: BOOL SetTagBitState (Tag Tag_Name, short int value, PDWORD lp_dwstate);

Parameters

Tag_Name

Tag name

value

Value of the tags of a specified type.

lp_dwstate

Pointer to a DWORD to which the tag status is saved on completion of the system function cycle.

pFormat

Format description for all the requested tags along with name and value for each tag.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

TRUE

System function completed without errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

Example

The following program code uses the SetTagBitState function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag. "&dwstate" is the address of the tag to which the tag status is saved.

The saved return value can be processed in the following code.

```
{
DWORD dwstate;
BOOL ok;

//Load dwState with default values
dwstate = 0xFFFFFFFF;
//Set the value of the tag to TRUE
//dwstate is the tag state
ok = SetTagBitState("gs_tag_bit",TRUE,&dwstate);

//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );
    printf ("Status of gs_tag_bit: %d\r\n", dwstate);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

2.1.35.6 SetTagStateWait functions (RT Professional)**Description**

Sets the value of a tag of a specified data type. The system function will end only after the AS has reported the acceptance of the value. It also returns the status of the tags.

Can only be used in C scripting.

The following table lists the different SetTagStateWait functions for setting the tag value:

Function name	Parameters	PLC data type	HMI data type
SetTagBitState-Wait	Tag Tag_Name, short int value, PDWORD lp_dwstate	Binary tag	Bool
SetTagByteState-Wait	Tag Tag_Name, BYTE value, PDWORD lp_dwstate	Unsigned 8-bit	UByte

Function name	Parameters	PLC data type	HMI data type
SetTagCharStateWait	Tag Tag_Name, LPSTR value, PDWORD lp_dwstate	Text tag 8-bit or text tag 16-bit	String
SetTagDoubleStateWait	Tag Tag_Name, double value, PDWORD lp_dwstate	Floating point 64-bit	Double
SetTagDWordStateWait	Tag Tag_Name, DWORD value, PDWORD lp_dwstate	Unsigned 32-bit	UInteger
SetTagFloatStateWait	Tag Tag_Name, float value, PDWORD lp_dwstate	Floating point 32-bit	Float
SetTagRawStateWait	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	Raw data type	Raw
SetTagSByteStateWait	Tag Tag_Name, char value, PDWORD lp_dwstate	Signed 8-bit	Byte
SetTagSDWordStateWait	Tag Tag_Name, long int value, PDWORD lp_dwstate	Signed 32-bit	Integer
SetTagSWordStateWait	Tag Tag_Name, short int value, PDWORD lp_dwstate	Signed 16-bit	Short
SetTagWordStateWait	Tag Tag_Name, WORD value, PDWORD lp_dwstate	Unsigned 16-bit	UShort

Syntax

BOOL<FunctionName><(Parameter)>;

Example: BOOL SetTagBitStateWait (Tag Tag_Name, short int value, PDWORD lp_dwstate);

Parameters

Tag_Name

Tag name

value

Value of the tags of a specified type.

lp_dwstate

Pointer to a DWORD to which the tag status is saved on completion of the system function cycle.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

TRUE

System function completed without errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

Example

The following program code uses the SetTagBitStateWait function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag. "&dwstate" is the address of the tag to which the tag status is saved.

The saved return value can be processed in the following code.

```
{
DWORD dwstate;
BOOL ok;

//Load dwState with default values
dwstate = 0xFFFFFFFF;
//Set the value of the tag to TRUE
//dwstate is the tag state
ok = SetTagBitStateWait("gs_tag_bit",TRUE,&dwstate);

//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );
    printf ("Status of gs_tag_bit: %d\r\n", dwstate);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

2.1.35.7 SetTagValue functions (RT Professional)

Description

Enables the transfer of a value in the form of a variant and sets the pointer to the value of data type "Variant".

Can only be used in C scripting.

Syntax

```
BOOL SetTagValue (LPDM_VARKEY lpdmVarKey, LPVARIANT lpdmValue, PDWORD dwState, LPCMN_ERROR lpdmError);
```

Parameters

lpdmVarKey

Pointer to a structure of data type "DM_VARKEY"

lpdmValue

Pointer to a structure of data type "Variant". Refer to relevant technical references for the description of data type VARIANT.

lpdmError

Pointer on the structure that contains the error description.

Return value

TRUE

System function completed without errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

Example

The SetTagValue function is used in the following program code to transfer the value in varKey.

A specific code will be executed, depending on the return value in keyFound (TRUE/FALSE).

```
{
// tags for setting the value
DM_VARKEY varKey;
LPVARIANT value;
LPCMN_ERROR error1;

// tags for getting the value
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error;

BOOL keyFound;

ok = SetTagValue(&varKey, &value, &error1);

if (keyFound)
{
    // succeeded, get the new value
    GetTagValue(&varKey, &result, &error);
    // print tag value
    printf ("Value of varKey: %d\r\n", &varKey);
    ...
}
else
{
    // failed
    printf ( "Error - function failed." );
    ...
}
}
```

2.1.35.8 SetTagValueWait functions (RT Professional)

Description

Enables the transfer of a value in the form of a variant and sets the pointer to the value of data type "Variant". The system function will end only after the AS has reported the acceptance of the value.

Can only be used in C scripting.

Syntax

BOOL SetTagValueWait(LPDM_VARKEY lpdmVarKey, LPVARIANT lpdmValue, PDWORD dwState, LPCMN_ERROR lpdmError);

Parameters

lpdmVarKey

Pointer to a structure of data type "DM_VARKEY"

lpdmValue

Pointer to a structure of data type "Variant". Refer to relevant technical references for the description of data type VARIANT.

dwState

Status of the tags that is returned after the system function cycle.

lpdmError

Pointer on the structure that contains the error description.

Return value

TRUE

System function completed without errors.

There is no check to see if the tag was written without any errors.

FALSE

An error has occurred.

Example

The SetTagValueWait function is used in the following program code to transfer the value in varKey.

A specific code will be executed, depending on the return value in keyFound (TRUE/FALSE).

```
{
// tags for setting the value
DM_VARKEY varKey;
LPVARIANT value;
LPCMN_ERROR error1;

// tags for getting the value
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error;

BOOL keyFound;

ok = SetTagValueWait(&varKey, &value, &error1);

if (keyFound)
{
    // succeeded, get the new value
    GetTagValueWait(&varKey, &result, &error);
    // print tag value
    printf ("Value of varKey: %d\r\n", &varKey);
    ...
}
else
{
    // failed
    printf ( "Error - function failed." );
    ...
}
}
```

2.1.35.9 SetTagWait functions (RT Professional)

Description

Sets the value of a tag of a specified data type. The system function will end only after the AS has reported the acceptance of the value.

Can only be used in C scripting.

The following table lists the different SetTagWait functions for setting the tag value:

Function name	Parameters	PLC data type	HMI data type
SetTagBitWait	Tag Tag_Name, short int value	Binary tag	Bool
SetTagByteWait	Tag Tag_Name, BYTE value	Unsigned 8-bit	UByte
SetTagCharWait	Tag Tag_Name, char* value	Text tag 8-bit or text tag 16-bit	String
SetTagDoubleWait	Tag Tag_Name, double value	Floating point 64-bit	Double

Function name	Parameters	PLC data type	HMI data type
SetTagDWord-Wait	Tag Tag_Name, DWORD value	Unsigned 32-bit	UInteger
SetTagFloat-Wait	Tag Tag_Name, float value	Floating point 32-bit	Float
SetTagRaw-Wait	Tag Tag_Name, BYTE* pValue, DWORD size	Raw data type	Raw
SetTagSByte-Wait	Tag Tag_Name, char value	Signed 8-bit	Byte
SetTagSDWord-Wait	Tag Tag_Name, long int value	Signed 32-bit	Integer
SetTagSWord-Wait	Tag Tag_Name, short int value	Signed 16-bit	Short
SetTagWord-Wait	Tag Tag_Name, WORD value	Unsigned 16-bit	UShort

Syntax

BOOL <FunctionName><(Parameter)>;

Example: BOOL SetTagBitWait (Tag Tag_Name, short int value);

Parameters

Tag_Name

Tag name

value

Value of the tags of a specified type.

pValue

Pointer to a byte field containing the value of the raw data tags.

size

Length of the byte field in bytes

Return value

TRUE

System function completed without errors.

However, no check is made to verify that the tag was written without any errors.

FALSE

An error has occurred.

Example

The following program code uses the SetTagBitWait function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTagBitWait("gs_tag_bit", TRUE);
//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );
    bvalue = GetTagBitWait("gs_tag_bit");
    printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

2.1.35.10 SetTag (RT Professional)

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
    // succeeded
    printf ( "Function has run through.\r\n" );
    bvalue = GetTagBit("gs_tag_bit");
    printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
    // failed
    printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

2.1.35.11 SetTagByProperty (RT Professional)

Description

Specifies a tag value with the value of an object property. The change is also logged in the alarm system.

Use in the function list

SetTagIndirectByProperty (tag name, screen name, screen object, name of the property, with or without operator event)

Use in user-defined functions

SetTagByProperty (Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of the tag whose value is specified by the object property.

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property supplies the tag value.

Name of the property

Name of the property that supplies the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code returns the value of the selected text when you click in a combo box.

```
{
char* rt_value;

SetTagByProperty (rt_value, screenName, objectName, "SelectedText",
hmiWithoutOperatorEvent);

...
}
```

2.1.35.12 **SetTagByTagIndirect (RT Professional)**

Description

Writes the value of an indirect addressed tag to a tag. The tag transferred as parameter contains the name of a second tag the value of which is read. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByTagIndirect (name of the tag, tag name, with or without operator input alarm)

Use in user-defined functions

SetTagByTagIndirect (Tag_name, Source_tag_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Name of the tag

Name of the tag whose value is to be set.

Tag name

Name of a string tag which contains the name of a tag which supplies the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator input alarm

1 (hmiWithOperatorEvent) = With operator input alarm

Example

The following program code writes the value of the tag Tag4 to the tag Tag1.

```
{  
SetTag ("IndirectRead", "Tag4");  
SetTagByTagIndirect ("Tag1", "IndirectRead", hmiWithoutOperatorEvent);  
  
...  
}
```

2.1.35.13 SetTagIndirect (RT Professional)

Description

Writes a value to an indirect addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirect (tag name, , value, with or without operator event)

Use in user-defined functions

SetTagIndirect (Tag_name, Value, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Value

Value which is to be written.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code writes the value "value" to the tag Tag3.

```
{
int value;

SetTag ("IndirectWrite", "Tag3");
SetTagIndirect ("IndirectWrite", "value", hmiWithoutOperatorEvent);
...
}
```

2.1.35.14 SetTagIndirectByProperty (RT Professional)

Description

Writes the value of an object property to an indirectly addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByProperty (Tag name, Screen name, Screen object, Name of the property, With or without operator event)

Use in user-defined functions

SetTagIndirectByProperty (Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Screen name

Name of the screen that contains the object.

Screen object

Name of the object whose property supplies the value.

Name of the property

Name of the property which supplies the value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without input alarm

1 (hmiWithOperatorEvent) = With operator input alarm

Example

The following program code writes the value of the object property "Background color" to the tag Tag2.

```
{
SetTag ("IndirectWrite", "Tag2");
SetTagIndirectByProperty ("IndirectWrite", screenName, objectName, "BackColor",
hmiWithoutOperatorEvent);
...
}
```

2.1.35.15 SetTagIndirectByTagIndirect (RT Professional)

Description

Writes the value of an indirectly addressed tag to an indirectly addressed tag. The tag transferred as output parameter contains the name of a second tag the value of which is changed by the function. The tag transferred as parameter contains the name of a second tag the value of which is read. The change in the alarm system is logged via an operator input alarm.

Use in the function list

SetTagIndirectByTagIndirect (Tag name, Tag name, With or without operator input alarm)

Use in user-defined functions

SetTagIndirectByTagIndirect (Tag_name, Source_tag_name,
With_or_without_operator_event)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of a string tag which contains the name of a tag whose value is to be changed.

Tag name

Name of the indirect tag that returns the tag value.

With or without operator event

0 (hmiWithoutOperatorEvent) = Without operator event

1 (hmiWithOperatorEvent) = With operator event

Example

The following program code writes the value of the tag "Tag4" to the tag "Tag2".

```
{
SetTag ("IndirectWrite", "Tag2");
SetTag ("IndirectRead", "Tag4");
SetTagIndirectByTagIndirect ("IndirectWrite", "IndirectRead");
...
}
```

2.1.35.16 SetTagIndirectWithOperatorEvent (RT Professional)

Description

Specifies the indirect name for a tag. The change is also logged in the alarm system.

Use in the function list

SetTagIndirectWithOperatorInputAlarm (Tag name (output), LpValue)

Use in user-defined functions

SetTagIndirectWithOperatorEvent

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name (output)

Name of the tag to which the tag name is written.

LpValue

Name of the tag written to the tag.

2.1.35.17 SetTagWithOperatorEvent (RT Professional)

Description

Specifies the value for a tag. The change is also logged in the alarm system.

Use in the function list

SetTagWithOperatorEvent (tag name, value)

Use in user-defined functions

SetTagWithOperatorEvent (Tag_name, Value)

Can be used if the configured device supports user-defined functions. For more information, refer to "AUTOHOTSPOT".

Parameters

Tag name

Name of the tag whose value is to be set.

Value

Value written to the tag.

Example

The following program code transfers the value from the "value" tag to the "result" tag when you click the corresponding button.

```
'Programming language: VBS
SetTagWithOperatorEvent result, value
...
```

```
//Programming language: C
{
SetTagWithOperatorEvent ("result", "value");
...
}
```

2.1.36 StartProgram (RT Professional)

Description

Starts the specified program with the specified parameters.

The system function can be used in C scripting only

Syntax

```
void StartProgram(Program_name,Program_parameter,Display_mode,
Wait_for_program_end);
```

Parameters

Program_name

Path and name of the program to be started.

Program_parameters

Parameters to be used for start-up. Information on the possible parameters can be found in the description of the program to be started.

Display_mode

Defines the display mode in which the program is started:

0 (hmiShowNormal) = Display in window

1 (hmiShowMinimized) = Display in minimized window

2 (hmiShowMaximized) = Display in maximized window

3 (hmiShowMinimizedAndInactive) = Display in inactive and minimized window

Wait_for_program_end

The parameter is not evaluated by WinCC Runtime Professional.

Example

The following program code starts the calc.exe program in the minimized window.

```
{
BOOL Wait_for_program_end;
float number;

//start the program calc.exe
StartProgram("C:\\Winnt\\system32\\calc.exe", number, hmiShowMinimized,
Wait_for_program_end);
...
}
```

2.1.37 StopRuntime (RT Professional)

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{  
  
//Stop runtime and shutdown  
StopRuntime (hmiStopRuntimeAndOperationSystem);  
  
}
```

The saved return value can be processed in the following code.

2.1.38 StoreScreen (RT Professional)

Description

Saves the current screen. This screen can be opened with the ActivateStoredScreen system function.

Can only be used in C scripting.

Syntax

BOOL StoreScreen();

Return value

TRUE

System function completed without errors.

FALSE

An error has occurred.

Example

The following program code writes the return value of the StoreScreen function to the screen_stored tag and calls the stored screen, provided that this was saved without errors.

```
{
BOOL screen_stored;
screen_stored = StoreScreen();
//user defined code
...
//error handling
if(screen_stored)
{
    // succeeded
    ActivateStoredScreen();
    printf ( "Stored screen is now activated.\r\n" );
}
else
{
    // failed
    printf ( "Error - no screen stored." );
}
...
}
```

See also

ActivateStoredScreen (Page 1378)

2.1.39 TriggerOperatorEvent (RT Professional)

Description

The TriggerOperatorEvent system function triggers an operator input alarm.

Syntax

```
int TriggerOperatorEvent( DWORD dwFlags, DWORD dwMsgNum, char* lpszObjectName,
DWORD dwMyTextID, double doValueOld, double doValueNew, char* pszComment );
```

Parameter

dwFlags

FLAG_COMMENT_PARAMETER (0x001): Specifies that the comment is input by means of the parameter.

FLAG_COMMENT_DIALOG (0x003): Specifies that the comment is input by means of a dialog.

FLAG_TEXTID_PARAMETER (0x100): Specifies that the comment is specified by means of a Text ID.

dwMsgNum

Number of the operator input alarm that is triggered.

lpszObjectName

Pointer to the name of the tag with the old value and the new value.

dwMyTextID

ID of the text to be used as comment.

doValueOld

Old value.

doValueNew

New value.

pszComment

Pointer to the text to be used as comment.

Return value

TRUE

The system function has been completed without any errors.

FALSE

An error has occurred.

2.1.40 UA (Recipe) (RT Professional)

2.1.40.1 uaArchiveClose (RT Professional)

Description

Terminates the connection to the current recipe.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveClose (
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using `uaQueryArchive` or `uaQueryArchiveByName`.

Return value

TRUE

Successful closing of the recipe

FALSE

Error

2.1.40.2 **uaArchiveDelete (RT Professional)**

Description

Deletes the data from a recipe. The configured recipe, however, is retained.
Can only be used in C scripting.

Syntax

```
BOOL uaArchiveDelete (
    UAHARCHIVE hArchive,
    LPCSTR pszWhere )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using `uaQueryArchive` or `uaQueryArchiveByName`.

LPCSTR pszWhere

This string contains the SQL selection expression. It defines which data records are to be deleted. The expression is equivalent to the SQL instruction "DELETE FROM <archive> WHERE pszWhere".

Warning! If this string is empty, the entire recipe will be deleted.

Return value

TRUE

Successful deletion of the recipe

FALSE

Error

2.1.40.3 uaArchiveExport (RT Professional)

Description:

Exports the current recipe to a log in CSV format.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveExport (  
    UAHARCHIVE hArchive,  
    LPCSTR pszDestination,  
    LONG lType,  
    LONG lOptions )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using UaQueryArchive or UaQueryArchiveByName.

LPCSTR pszDestination

File name of the target archive.

LONG lType

Data format of the target archive. Two formats are available:

- UA_FILETYPE_DEFAULT = 0: Default file format = CSV
- UA_FILETYPE_CSV = 1: CSV file format

LONG lOptions

Reserved for future expansions. Must be 0.

Return value

TRUE

Successful export of the recipe

FALSE

Error

2.1.40.4 uaArchiveGetCount (RT Professional)

Description

Reads the number of data records.

Can only be used in C scripting.

Syntax

```
LONG uaArchiveGetCount(  
    UAHARCHIVE hArchive,  
    LONG * plCount )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName .

LONG plCount

Pointer to a tag in which the number of data records should be stored.

Return value

Number of data records.

0 = Log is empty or error has occurred. Must be queried by means of uaGetLastError() .

2.1.40.5 uaArchiveGetFieldLength (RT Professional)

Description

Reads the length of a field in the current data record.

Can only be used in C scripting.

Syntax

```
LONG uaArchiveGetFieldLength(  
    UAHARCHIVE hArchive,  
    LONG lField )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first field.

Return value

Length of the current field.

2.1.40.6 uaArchiveGetFieldName (RT Professional)

Description

Reads the name of a field in the current data record.

Can only be used in C scripting.

Syntax

```
VOID uaArchiveGetFieldName (  
    UAHARCHIVE hArchive,  
    LONG lField,  
    LPCSTR pszName,  
    LONG cMaxLen )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first field.

LPCSTR pszName

Field Name

LONG cMaxLen

Maximum length

2.1.40.7 uaArchiveGetFields (RT Professional)**Description**

Reads the number of configured data fields, which also includes the "ID", "Last User", and "Last Access" fields. In the Runtime calls, the indexes of the configured fields are indicated 1 to N. The field ID has the index 0. The fields "Last User" and "Last Access" are appended to the end of the configured fields.

Can only be used in C scripting.

Syntax

```
LONG uaArchiveGetFields (
    UAHARCHIVE hArchive )
```

Parameters**UAHARCHIVE hArchive**

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

Number of configured fields.

2.1.40.8 uaArchiveGetFieldType (RT Professional)**Description**

Reads the type of a field in the current data record.

Can only be used in C scripting.

Syntax

```
LONG uaArchiveGetFieldType (
    UAHARCHIVE hArchive,
    LONG lField )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first field.

Return value

Type of the current field.

The symbolic definitions of the field types are:

UA_FIELDTYPE_INTEGER

UA_FIELDTYPE_DOUBLE

UA_FIELDTYPE_STRING

UA_FIELDTYPE_DATETIME

2.1.40.9 uaArchiveGetFieldValueDate (RT Professional)

Description

Reads the date and time of a field in the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveGetFieldValueDate (  
    UAHARCHIVE hArchive,  
    LONG lField,  
    LPSYSTEMTIME pstDateTime )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName .

LONG lField

Field number, with lField = 1 addressing the first field.

LPSYSTEMTIME pstDateTime

Pointer to a tag of the type SYSTEMTIME

Return value

TRUE

Successful reading of date and time

FALSE

Error

2.1.40.10 uaArchiveGetFieldValueDouble (RT Professional)

Description

Reads the Double value of a field in the current data record.

The system function can be used in C scripting only.

```
BOOL uaArchiveGetFieldValueDouble (
    UAHARCHIVE hArchive,
    LONG lField,
    double* pdValue )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first field.

double* pdValue

Pointer to tag for the current field content.

Return value

TRUE

Successful reading of field value

FALSE

Error

2.1.40.11 uaArchiveGetFieldValueFloat (RT Professional)

Description

Reads Float value of a field in the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveGetFieldValueFloat (  
    UAHARCHIVE hArchive,  
    LONG lField,  
    FLOAT* pfValue )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first field.

FLOAT* pfValue

Pointer to Float tag for the current field content.

Return value

TRUE

Successful reading of field value

FALSE

Error

2.1.40.12 uaArchiveGetFieldValueLong (RT Professional)

Description

Reads the Long Integer value of a field in the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveGetFieldValueLong (
    UAHARCHIVE hArchive,
    LONG lField,
    LONG* pdValue )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first field.

LONG* pdValue

Pointer to Long tag for the current field content.

Return value

TRUE

Successful reading of field value

FALSE

Error

2.1.40.13 uaArchiveGetFieldValueString (RT Professional)

Description

Reads the string of a field in the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveGetFieldValueString (
    UAHARCHIVE hArchive,
    LONG lField,
    LPSTR pszString,
    LONG cMaxLen )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first field.

LPCSTR pszString

Field value as string.

LONG cMaxLen

Maximum length of the string.

Return value

TRUE

Successful reading of field value

FALSE

Error

2.1.40.14 uaArchiveGetFilter (RT Professional)

Description

Reads the filter of the current data record. Additional information can be found in the appendix under "SQL Statements".

Can only be used in C scripting.

Syntax

```
VOID uaArchiveGetFilter (  
    UAHARCHIVE hArchive,  
    LPSTR pszFilter,  
    LONG cMaxLen )
```

The system function can be used in C scripting only.

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LPSTR pszFilter

Read filter.

LONG cMaxLen

Maximum length

2.1.40.15 uaArchiveGetID (RT Professional)

Description

uaArchiveGetID reads the ID of the recipe.

The recipe ID serves internal purposes and may differ from the number given in the recipe.

Can only be used in C scripting.

Syntax

```
LONG uaArchiveGetID (  
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

ID of the recipe.

2.1.40.16 uaArchiveGetName (RT Professional)

Description

Reads the name of the recipe.

Can only be used in C scripting.

Syntax

```
VOID uaArchiveGetName (  
    UAHARCHIVE hArchive,  
    LPSTR pszName,  
    LONG cMaxLen )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LPSTR pszName

Pointer to buffer for recipe name.

LONG cMaxLen

Maximum length

Example

```
char Filling [40];  
uaArchiveGetName( hArchive, bottling, 39 );
```

2.1.40.17 uaArchiveGetSor (RT Professional)

Description

uaArchiveGetSort reads the sorting of the recipe.

Can only be used in C scripting.

Syntax

```
VOID uaArchiveGetSort (  
    UAHARCHIVE hArchive,  
    LPSTR pszSort,  
    LONG cMaxLen )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LPCSTR pszSort

Sorting

LONG cMaxLen

Maximum length

2.1.40.18 uaArchiveImport (RT Professional)

Description

uaArchiveImport imports a recipe with CSV data format. The structure of the destination recipe must be identical to that of the imported recipe.

Syntax

```
BOOL uaArchiveImport (  
    UAHARCHIVE hArchive,  
    LPCSTR pszSource,  
    LONG lType,  
    LONG lOptions )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LPCSTR pszSource

File name of the source archive.

LONG lType

Data format of the source archive. Two formats are available:

UA_FILETYPE_DEFAULT = 0: Default file format = CSV

UA_FILETYPE_CSV = 1: CSV file format

LONG lOptions

Reserved for future expansions. Must be 0.

Return value

TRUE

Successful import of the recipe.

FALSE

Error

2.1.40.19 uaArchiveInsert (RT Professional)

Description

Inserts the local data record buffer into the current database. Before you call uaArchiveInsert, use the "uaArchiveSetFieldValue..." system functions to enter the data in the fields of the local data buffer so that the new data record contains useful data.
Use system function "uaArchiveSetFieldValueLong" to enter the internal column "ID" in the current data record. Can only be used in C scripting.

Syntax

```
BOOL uaArchiveInsert (  
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful insertion of the data record.

2.1.40.20 uaArchiveMoveFirst (RT Professional)

Description

Goes to the first data record.
Can only be used in C scripting.

Syntax

```
BOOL uaArchiveMoveFirst (  
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful jumping in the recipe

FALSE

Error

2.1.40.21 uaArchiveMoveLast (RT Professional)

Description

Goes to the last data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveMoveLast (  
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName .

Return value

TRUE

Successful jumping in the recipe

FALSE

Error

2.1.40.22 uaArchiveMoveNext (RT Professional)

Description

Goes to the next data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveMoveNext (
```

```
UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful jumping in the recipe

FALSE

Error

2.1.40.23 uaArchiveMovePrevious (RT Professional)

Description

Goes to the previous data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveMovePrevious (  
UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful jumping in the recipe

FALSE

Error

2.1.40.24 uaArchiveOpen (RT Professional)

Description

uaArchiveOpen must be called before all RT functions (e.g. uaArchiveMoveFirst, uaArchiveMoveLast, uaArchiveMoveNext, uaArchiveMovePrevious, uaArchiveDelete, uaArchiveUpdate, uaArchiveInsert, uaArchiveGetID, uaArchiveGetFields, uaArchiveGetFieldType, uaArchiveGetFieldValueDate, uaArchiveGetFieldValueDouble, uaArchiveGetFieldValueFloat, uaArchiveGetFieldValueLong, uaArchiveGetFieldValueString, uaArchiveSetFieldValueDate, uaArchiveSetFieldValueDouble, uaArchiveSetFieldValueFloat, uaArchiveSetFieldValueLong and uaArchiveSetFieldValueString).

Note

Sort and filter recipes

You may apply the "uaArchiveSetSort" and "uaArchiveSetFilter" system functions to a recipe without having to open this recipe by means of "uaArchiveOpen".

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveOpen (  
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful opening of the recipe

FALSE

Error

2.1.40.25 uaArchiveReadTagValues (RT Professional)

Description

Reads the current value from the field tag.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveReadTagValues (  
    UAHARCHIVE hArchive,  
    LONG* pnFields,  
    LONG cFields,  
    LONG lOptions )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG* pnFields

Reserved for future applications (NULL)

LONG cFields

Number of field indices transferred (size of array pnFields).

Reserved for future applications (0)

LONG lOptions

Reserved for future applications (0)

In the case of all other values of lOptions, the data is inserted at the position of the pointer.

Return value

TRUE

Successful reading in the recipe

FALSE

Error

2.1.40.26 uaArchiveReadTagValuesByName (RT Professional)

Description

Reads the tag values in the current data.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveReadTagValuesByName (
```



```

UAHARCHIVE hArchive,
LPCSTR pszFields,
LONG lOptions )

```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using `uaQueryArchive` or `uaQueryArchiveByName`.

LPCSTR pszFields

Reserved for future applications (NULL)

LONG lOptions

Reserved for future applications (0)

Return value

TRUE

Successful reading in the recipe

FALSE

Error

2.1.40.27 uaArchiveRequery (RT Professional)

Description

On completion of the call of `uaArchiveSetFilter` and `uaArchiveSetSort`, reload the recipe by means of `uaArchiveRequery`.

Note

Sort and filter recipes

You may apply the "uaArchiveSetSort" and "uaArchiveSetFilter" system functions to a recipe without having to open this recipe by means of "uaArchiveOpen". In this case you do not have to call the system function "uaArchiveRequery".

Call `uaArchiveRequery` in the following situations as well:

- If you have made inputs in the recipe view.
- If you have made inputs in the "Recipes" editor that are to be transferred to the table cell.

Syntax

```

BOOL uaArchiveRequery(

```

UAHARCHIVE hArchive)

Can only be used in C scripting.

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful requery

FALSE

Error

2.1.40.28 uaArchiveSetFieldValueDate (RT Professional)

Description

Writes the date and time into a field of the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveSetFieldValueDate (
    UAHARCHIVE hArchive,
    LONG lField,
    LPSYSTEMTIME pstDateTime )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first configured field. The ID field is addressed with lField = 0.

LPSYSTEMTIME pstDateTime

Date and Time

Return value

TRUE

Successful writing of date and time

FALSE

Error

2.1.40.29 uaArchiveSetFieldValueDouble (RT Professional)

Description

Writes a Double value into a field of the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveSetFieldValueDouble (  
    UAHARCHIVE hArchive,  
    LONG lField,  
    double dValue )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

The field number, where lField = 1 is addressing the first configured field. The ID field is addressed with lField = 0.

double dValue

Field value

Return value

TRUE

Successful writing of field value

FALSE

Error

2.1.40.30 uaArchiveSetFieldValueFloat (RT Professional)

Description:

Writes a Float value into a field of the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveSetFieldValueFloat (
    UAHARCHIVE hArchive,
    LONG lField,
    float fValue )
```

Parameters:

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

Field number, with lField = 1 addressing the first configured field. The ID field is addressed with lField = 0.

float fValue

Field value

Return value:

TRUE

Successful writing of field value

FALSE

Error

2.1.40.31 uaArchiveSetFieldValueLong (RT Professional)

Description

Writes a Long Integer value into a field of the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveSetFieldValueLong (  
    UAHARCHIVE hArchive,  
    LONG lField,  
    LONG dValue )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

The field number, where lField = 1 is addressing the first configured field. The ID field is addressed with lField = 0.

LONG dValue

Field value

Return value

TRUE

Successful writing of field value

FALSE

Error

2.1.40.32 uaArchiveSetFieldValueString (RT Professional)

Description

Writes a String into a field of the current data record.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveSetFieldValueString (  
    UAHARCHIVE hArchive,  
    LONG lField,  
    LPCSTR pszString )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG lField

The field number, where lField = 1 is addressing the first configured field. The ID field is addressed with lField = 0 .

LPCSTR pszString

Field value

Return value

TRUE

Successful writing of field value

FALSE

Error

2.1.40.33 uaArchiveSetFilter (RT Professional)

Description

Sets the filter. You can call the system function without having opened the recipe with "uaArchiveOpen".

Note

If you have opened the recipe with "uaArchiveOpen", reload the recipe after filtering by means of "uaArchiveRequery".

Can only be used in C scripting.

Syntax

```
VOID uaArchiveSetFilter (
    UAHARCHIVE hArchive,
    LPSTR pszFilter )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LPSTR pszFilter

Filter to be set.

2.1.40.34 uaArchiveSetSort (RT Professional)

Description

Sets the sorting of the recipe. You can call the system function without having opened the recipe with "uaArchiveOpen".

Note

If you have opened the recipe with "uaArchiveOpen", reload the recipe after sorting by means of "uaArchiveRequery".

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveSetSort (  
    UAHARCHIVE hArchive,  
    LPSTR pszSort )
```

The system function can be used in C scripting only.

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LPCSTR pszSort

Sorting

Return value

TRUE

Successful setting of the sorting

FALSE

Error

2.1.40.35 uaArchiveUpdate (RT Professional)

Description

Updates the open recipe. All data changes of a recipe are transferred to the database. The configuration of the recipe remains unchanged.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveUpdate (  
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful update of the recipe

FALSE

"Update_failed" error = 106

This error occurs when there is a consistency violation. Example: The "Field requires a value" flag is set in a field, but no value has been entered there.

2.1.40.36 uaArchiveWriteTagValues (RT Professional)

Description

Writes the values of the current data record into the tags.

Can only be used in C scripting.

Syntax

```
BOOL uaArchiveWriteTagValues (  
    UAHARCHIVE hArchive,  
    LONG* pnFields,
```



```

LONG cFields,
LONG lOptions )

```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

LONG* pnFields

Reserved for future applications (NULL)

LONG cFields

Reserved for future applications (0)

LONG lOptions

Reserved for future applications (0)

Return value

TRUE

Successful reading in the recipe

FALSE

Error

2.1.40.37 uaArchiveWriteTagValuesByName (RT Professional)

Description

Writes the values of the current data record into the tags. The access is based on the names of the recipe and field.

Can only be used in C scripting.

Syntax

```

BOOL uaArchiveWriteTagValuesByName (
    UAHARCHIVE hArchive,
    LPCSTR pszFields,
    LONG lOptions )

```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using `uaQueryArchive` or `uaQueryArchiveByName`.

LPCSTR pszFields

Reserved for future applications (NULL)

LONG lOptions

Reserved for future applications (0)

Return value

TRUE

Successful reading in the recipe

FALSE

Error

2.1.40.38 uaConnect (RT Professional)

Description

Establishes connection to recipes (Runtime).

Can only be used in C scripting.

Syntax

```
BOOL uaConnect (  
    UAHCONNECT* phConnect )
```

Parameters

UAHCONNECT* phConnect

Pointer to handle for newly connected recipe.

Return value

TRUE

Successful connection of a recipe

FALSE

Error

2.1.40.39 uaDisconnect (RT Professional)

Description

If a connection to recipes (Runtime) exists, it will be disconnected.
Can only be used in C scripting.

Syntax

```
BOOL uaDisconnect (  
    UAHCONNECT hConnect )
```

Parameters

UAHCONNECT hConnect

Handle for the connected recipe (Runtime). The handle is generated using uaConnect.

Return value

TRUE

Successful disconnection of a recipe

FALSE

Error

2.1.40.40 uaGetArchive (RT Professional)

Description

Reads the recipe configuration.
Can only be used in C scripting.

Syntax

```
BOOL uaGetArchive (  
    UAHCONFIG hConfig,  
    long lArchive,  
    UACONFIGARCHIVE* pArchive )
```

Parameters

UAHCONFIG hConfig,

Configuration handle of the recipe. This handle is generated using uaQueryConfiguration.

long lArchive,

Archive index (0 to (uaGetNumArchives()-1))

UACONFIGARCHIVE* pArchive

Pointer to buffer for receiving the recipe configuration

Return value

TRUE

Successful access to the recipe

FALSE

Error

2.1.40.41 uaGetField (RT Professional)

Description:

Reads the field configuration.

Can only be used in C scripting.

Syntax

```
BOOL uaGetField (  
    UAHCONFIG hConfig,  
    long lArchive,  
    long lField,  
    UACONFIGFIELD* pField )
```

Parameters:

UAHCONFIG hConfig,

Configuration handle of the recipe. This handle is generated using uaQueryConfiguration.

long lArchive,

Archive index (0 to (uaGetNumArchives()-1))

long lField,

The field number; if lField = 0 the first field is addressed.

UACONFIGFIELD* pArchive

Pointer to buffer for receiving the field configuration.

Return value**TRUE**

Successful access to the recipe

FALSE

Error

2.1.40.42 uaGetLastError (RT Professional)**Description**

The system functions of the WinCC script language return a BOOL value; TRUE corresponds to error-free processing. If FALSE is returned, you can use "uaGetLastError()" and "uaGetLastHResult()" to read the error of the last system function.

Can only be used in C scripting.

If you do not call uaGetLastError() until several system functions have been processed, uaGetLastError() will return the most recent error. You should call a "uaGetLastError()" and "uaGetLastHResult()" system function whenever FALSE is returned to identify the system function that triggered the error.

Example:

```
if ( uaArchiveGetFieldValueLong ( hArchive, Index, &IntValue ) ==
TRUE )

printf( "Field Value = %u\n", IntValue );

else

printf("Error calling uaArchiveGetFieldValueLong: %d / %08lx\n",
uaGetLastError(), uaGetLastHResult());
```

You should always query system functions that do not return a value (VOID) by calling uaGetLastError() .

Example:

```
uaArchiveGetFilter(hArchive, pszFilter, cMaxLen);

INT nUAError = uaGetLastError ( );

if ( UA_ERROR_SUCCESS != nUAError)

{

    printf( "Filter = [%s]\n", pszFilter );

}

else

{
```

```
        printf("Error calling uaArchiveGetFilter: %d, hr=0x%08lX\n",
nUAEError, uaGetLastHResult());
    }

    INT uaGetLastError()
```

Return value

Error status of the last system function executed. uaGetLastError() can return the following errors:

- UA_ERROR_SUCCESS
- UA_ERROR_GENERIC
- UA_ERROR_CONNECT_FAILED
- UA_ERROR_OPEN_FAILED
- UA_ERROR_CLOSE_FAILED
- UA_ERROR_REQUERY_FAILED
- UA_ERROR_MOVE_FAILED
- UA_ERROR_INSERT_FAILED
- UA_ERROR_UPDATE_FAILED
- UA_ERROR_DELETE_FAILED
- UA_ERROR_IMPORT_FAILED
- UA_ERROR_EXPORT_FAILED
- UA_ERROR_READ_FAILED
- UA_ERROR_WRITE_FAILED
- UA_ERROR_GET_FAILED
- UA_ERROR_SET_FAILED
- UA_ERROR_INVALID_NAME
- UA_ERROR_INVALID_TYPE
- UA_ERROR_INVALID_NUMRECS
- UA_ERROR_INVALID_COMMTYPE
- UA_ERROR_INVALID_LENGTH
- UA_ERROR_INVALID_PRECISION
- UA_ERROR_NULL_POINTER
- UA_ERROR_INVALID_POINTER
- UA_ERROR_INVALID_HANDLE
- UA_ERROR_INVALID_INDEX
- UA_ERROR_SERVER_UNKNOWN

These error constants as well as the predefines of the user archive routines are located in CCUACAPI.H.

2.1.40.43 uaGetLastHResult (RT Professional)

Description

Reads the last occurred COM error. This system function is used primarily to analyze incompatibility in the COM implementation or to identify registry errors and communication errors.

This system function must always be called in addition to UAGetLastError if one of the user archive system functions (e.g uaConnect) returns the value "FALSE" to signal an error.

Can only be used in C scripting.

Syntax

```
LONG uaGetLastHResult()
```

Return value

Last occurred COM error

2.1.40.44 uaGetNumArchives (RT Professional)

Description

Reads the number of recipes currently configured.

Can only be used in C scripting.

Syntax

```
LONG uaGetNumArchives (  
    UAHCONFIG hConfig )
```

Parameters

UAHCONFIG hConfig

Configuration handle of the recipe. This handle is generated using uaQueryConfiguration.

Return value

Number of recipes currently configured. In case of an error, -1 will be returned.

2.1.40.45 uaGetNumFields (RT Professional)

Description

Supplies the number of the configured fields. The "ID", "Last User" and "Last Access" fields are not included. Indexes are specified from 0 to uaGetNumFields() -1 in the configuration calls.

Can only be used in C scripting.

Syntax

```
LONG uaGetNumFields (  
    UAHCONFIG hConfig,  
    long lArchive )
```

Parameters

UAHCONFIG hConfig,

Configuration handle of the recipe. This handle is generated using uaQueryConfiguration.

long lArchive,

Archive index (0 to (uaGetNumArchives()-1))

Return value

Number of configured fields. In case of an error, -1 will be returned.

2.1.40.46 uaQueryArchive (RT Professional)

Description

Establishes a connection to the recipe for Runtime operation. UaQueryArchive creates the handle UAHARCHIVE.

Can only be used in C scripting.

Syntax

```
BOOL uaQueryArchive (  
    UAHCONNECT hConnect,  
    LONG lArchive,  
    UAHARCHIVE* phArchive )
```


Parameters

UAHCONNECT hConnect

Handle of the connected recipe (Runtime). The handle is generated using uaConnect.

LONG lArchive

ID of the archive to be connected

UAHARCHIVE* phArchive

Pointer to handle of the recipe

Return value

TRUE

Successful generation of the handle to the recipe.

FALSE

Error

Comment

If you use User Archives functions in a client project that works with redundant server pairs, the user archives connection cannot be switched automatically to the new master when switching masters. In this case, all user archives calls return the LastError `UA_ERROR_SERVER_UNKNOWN = 1004`, which means that the user programs must execute a new `uaQueryArchive()` or `uaQueryArchiveByName()` and `uaArchiveOpen()`.

2.1.40.47 uaQueryArchiveByName (RT Professional)

Description

Establishes a connection to the recipe for Runtime operation using the recipe name. `UaQueryArchiveByName` creates the `UAHARCHIVE` handle for the recipe.

Can only be used in C scripting.

Syntax

```
BOOL uaQueryArchiveByName (  
    UAHCONNECT hConnect,  
    LPCSTR pszName,  
    UAHARCHIVE* phArchive )
```

Parameters

UAHCONNECT hConnect

Handle of the connected recipe (Runtime). The handle is generated using uaConnect.

LPCSTR pszName

Name of the recipe. With a client project, you can add a server prefix with '::' to the recipe name as a separator, if a server other than the default server is used.

UAHARCHIVE* phArchive

Pointer to handle of the recipe

Return value

TRUE

Successful generation of the handle to the recipe.

FALSE

Error

Comment

If you use User Archives functions in a client project that works with redundant server pairs, the user archives connection cannot be switched automatically to the new master when switching masters. In this case, all user archives calls return the `LastError` `UA_ERROR_SERVER_UNKNOWN = 1004`, which means that the user programs must execute a new `uaQueryArchive()` or `uaQueryArchiveByName()` and `uaArchiveOpen()`.

2.1.40.48 UaQueryConfiguration (RT Professional)

Description

Establishes a connection to the recipe for the configuration.

Can only be used in C scripting.

Syntax

```
BOOL uaQueryConfiguration (  
    UAHCONFIG* phConfig )
```

Parameters

UAHCONFIG* phConfig,

Pointer to handle of the recipe.

Return value

TRUE

Successful access to recipe

FALSE

Error

2.1.40.49 **uaReleaseArchive (RT Professional)**

Description

Releases the connection to the current recipe.

Can only be used in C scripting.

Syntax

```
BOOL uaReleaseArchive (  
    UAHARCHIVE hArchive )
```

Parameters

UAHARCHIVE hArchive

Handle of the recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful release of connection to the recipe.

FALSE

Error

Comment

The "hArchive" handle must be set to "NULL" on successful release to make absolutely sure that the "UA_ERROR_INVALID_HANDLE" error is triggered on further use of the invalid handle and without the respective function dwelling at the COM interface for a longer period of time.

2.1.40.50 **uaReleaseConfiguration (RT Professional)**

Description

Releases connection to recipes (configuration).

Can only be used in C scripting.

Syntax

```
BOOL uaReleaseConfiguration (  
    UAHCONFIG hConfig,  
    BOOL bSave )
```

Parameters

UAHCONFIG hConfig

Configuration handle of the recipe. This handle is generated using uaQueryConfiguration.

BOOL bSave

Saves changes made to the configuration before releasing the connection to recipes for the configuration.

TRUE = Save Changes, FALSE = Discard Changes

Warning: Save changes (bSave = TRUE) may only be used when Runtime is not active! You can check whether Runtime is active by requesting ualsActive().

Return value

TRUE

Successful connection release

FALSE

Error

2.2 C-bib (RT Professional)

2.2.1 ctype functions (RT Professional)

Function overview

The following ctype functions are available:

- long int isalnum (long int x);
- long int isalpha (long int x);
- long int isdigit (long int x);
- long int isgraph (long int x);
- long int islower (long int x);

- long int isprint (long int x);
- long int ispunct (long int x);
- long int isspace (long int x);
- long int isupper (long int x);
- long int isxdigit (long int x);
- long int tolower (long int x);
- long int toupper (long int x);

For a description of the ctype functions see the specialist literature on programming language C.

The functions can be used in C scripts only.

2.2.2 Function group c_bib (RT Professional)

Introduction

The c_bib function group contains C functions from the C library and is divided into:

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio itself is again divided into:

- char_io
- directio
- error
- file
- file_pos
- output

For a description of these functions see the relevant specialist literature.

Specific features of WinCC functions

The localtime function behaves as follows when the date is issued:

- The counting of the months begins with 0.
- The years are counted as of 1990, starting with 0.

Only 360 characters of the functions of the C library printf(), sprintf(), fprintf() can be edited in WinCC.

2.2.3 math functions (RT Professional)

Function overview

The following math functions are available:

- double acos (double x);
- double asin (double x);
- double atan (double x);
- double atan2 (double x, double y);
- double ceil (double x);
- double cos (double x);
- double cosh (double x);
- double exp (double x);
- double fabs (double x);
- double floor (double x);
- double fmod (double x, double y);
- double frexp (double x, long int* y);
- double ldexp (double x, long int y);
- double log (double x);
- double log10 (double x);
- double modf (double x, double* y);
- double pow (double x, double y);
- double sin (double x);
- double sinh (double x);
- double sqrt (double x);
- double tan (double x);
- double tanh (double x);

For a description of the math functions see the specialist literature on programming language C.

The functions can be used in C scripts only.

2.2.4 memory functions (RT Professional)

Function overview

The following memory functions are available:

- long int memcmp (const void* cs, const void* ct, size_t n);
- void* memchr (const void* cs, long int c, size_t n);
- void* memcpy (void* s, const void* ct, size_t n);
- void* memmove (void* s, const void* ct, size_t n);
- void* memset (void* s, long int c, size_t n);

For a description of the memory functions see the specialist literature on programming language C.

The functions can be used in C scripts only.

2.2.5 multibyte functions (RT Professional)

Function overview

The following multibyte functions are available:

- int _ismbcalnum(unsigned int c)
- int _ismbcalpha(unsigned int c)
- int _ismbcdigit(unsigned int c)
- int _ismbcgraph(unsigned int c)
- int _ismbcclower(unsigned int c)
- int _ismbcprint(unsigned int c)
- int _ismbcpunct(unsigned int c)
- int _ismbcspc(unsigned int c)
- int _ismbcupper(unsigned int c)
- int _mbscmp(const unsigned char *string1, const unsigned char *string2)
- int _mbsncmp(const unsigned char *string1, const unsigned char *string2, size_t count)
- int _mbsrchr(const unsigned char *string, unsigned int c)
- size_t _mbscspn(const unsigned char *string, const unsigned char *strCharSet)
- size_t _mbsspn(const unsigned char *string, const unsigned char *strCharSet)

- `size_t _mbstrlen( const char *string )`
- `size_t _mbslen( const unsigned char *string )`
- `unsigned char *_mbscat( unsigned char *strDestination, const unsigned char *strSource)`
- `unsigned char *_mbschr( const unsigned char *string, unsigned int c )`
- `unsigned char *_mbscpy( unsigned char *strDestination, const unsigned char *strSource )`
- `unsigned char *_mbsdec( const unsigned char *start, const unsigned char *current )`
- `unsigned char *_mbsinc( const unsigned char *current )` `size_t _mbclen( const unsigned char *c );`
- `unsigned char *_mbsncat( unsigned char *strDest, const unsigned char *strSource, size_t count)`
- `unsigned char *_mbsncpy( unsigned char *strDest, const unsigned char *strSource, size_t count)`
- `unsigned char *_mbspbrk( const unsigned char*string, const unsigned char *strCharSet )`
- `unsigned char *_mbsstr( const unsigned char *string, const unsigned char *strCharSet )`
- `unsigned char *_mbstok( unsigned char*strToken, const unsigned char *strDelimit )`

A description of the multibyte functions can be found in the specialist literature on programming language C.

The functions can be used in C scripts only.

2.2.6 stdio functions (RT Professional)

Function overview

The following stdio functions are available:

- `char* fgets (char* s, long int n, FILE* stream);`
- `char* tmpnam (char* s);`
- `FILE* fopen (const char* name, const char* mode);`
- `FILE* freopen (const char* filename, const char* mode, FILE* stream);`
- `FILE* tmpfile ();`
- `fprintf();`
- `long int fclose (FILE* stream);`
- `long int feof (FILE* stream);`
- `long int ferror (FILE* stream);`
- `long int fflush (FILE* stream);`
- `long int fgetc (FILE* stream);`
- `long int fgetpos (FILE* stream, fpos_t* ptr);`

- long int fputc (long int c, FILE* stream);
- long int fputs (const char* s, FILE* stream);
- long int fseek (FILE* stream, long int offset, long int origin);
- long int fsetpos (FILE* stream, const fpos_t* ptr);
- long int ftell (FILE* stream);
- long int getc (FILE* stream);
- long int putc (long int c, FILE* stream);
- long int remove (const char* filename);
- long int rename (const char* oldname, const char* newname);
- long int setvbuf (FILE* stream, char* buf, long int mode, size_t size);
- long int ungetc (long int c, FILE* stream);
- long int vfprintf (FILE* stream, const char* format, va_list arg);
- long int vsprintf (char* s, const char* format, va_list arg);
- printf();
- size_t fread (void* ptr, size_t size, size_t nobj, FILE* stream);
- size_t fwrite (void* ptr, size_t size, size_t nobj, FILE* stream);
- void clearerr (FILE* stream);
- void rewind (FILE* stream);
- void setbuf (FILE* stream, char* buf);

For a description of the studio functions see the specialist literature on programming language C.

The functions can be used in C scripts only.

2.2.7 stdlib functions (RT Professional)

Function overview

The following stdlib functions are available:

- char* getenv (const char* name);
- div_t div (long int num, long int denom);
- double atof (const char* s);
- double strtod (const char* s, char** endp);
- ldiv_t ldiv (long int num, long int denom);
- long int abs (long int n);
- long int atoi (const char* s);

- long int atol (const char* s);
- long int labs (long int n);
- long int rand ();
- long int srand (unsigned long int seed);
- long int strtol (const char* s, char** endp, long int base);
- long int system (const char* s);
- unsigned long int strtoul (const char* s, char** endp, long int base);
- void abort ();
- void* bsearch (const void* key, const void* base, size_t n, size_t size, long int(* cmp) (const
- void* calloc (size_t nobj, size_t size);
- void exit (long int status);
- void free (void* p);
- void* keyval, const void* datum));
- void* malloc (size_t size);
- void qsort (void* base, size_t n, size_t size, long int* cmp, const void* , const void*);
- void* realloc (void* p, size_t size);

For a description of the stdlib functions see the specialist literature on programming language C.

The functions can be used in C scripts only.

2.2.8 string functions (RT Professional)

Function overview

The following string functions are available:

- char* strcat (char* s, const char* ct);
- char* strchr (const char* cs, long int c);
- char* strcpy (char* s, const char* ct);
- char* strerror (size_t n);
- char* strncat (char* s, const char* ct, size_t n);
- char* strncpy (char* s, const char* ct, size_t n);
- char* strpbrk (const char* cs, const char* ct);
- char* strrchr (const char* cs, long int c);
- char* strstr (const char* cs, const char* ct);
- char* strtok (char* s, const char* ct);

- long int strcmp (const char* cs, const char* ct);
- long int strncmp (const char* cs, const char* ct, size_t n);
- size_t strcspn (const char* cs, const char* ct);
- size_t strlen (const char* cs);
- size_t strspn (const char* cs, const char* ct);

For a description of the string functions see the specialist literature on programming language C.

The functions can be used in C scripts only.

2.2.9 time functions (RT Professional)

Function overview

The following time functions are available:

- char* asctime (const struct tm* tp);
- char* ctime (const time_t* tp);
- clock_t clock ();
- double difftime (time_t time2, time_t time1);
- size_t strftime (char* s, size_t smax, const char* fmt, const struct tm* tp);
- struct tm* gmtime (const time_t* tp);
- struct tm* localtime (const time_t* tp);
- time_t mktime (struct tm* tp);
- time_t time (time_t* tp);

For a description of the time functions see the specialist literature on programming language C.

The functions can be used in C scripts only.

2.3 Structure definition (RT Professional)

2.3.1 Structure definition CCAPErrorsExecute (RT Professional)

```
typedef struct {
  DWORD dwCurrentThreadID; Thread ID of the current thread
  DWORD dwErrorCode1;      Error code 1
  DWORD dwErrorCode2;      Error code 2
  BOOL bCycle;             cycle/acycle
  char* szApplicationName; Name of the application
  char* szFunctionName;    Name of the function
  char* szTagName;         Name of the tag
  LPVOID lpParam;          Pointer to the action stack
  DWORD dwParamSize;       Size of the action stack
  DWORD dwCycle;           Cycle of the tag
  CMN_ERROR* pError;       Pointer to CMN_ERROR
} CCAPErrorsExecute;
```

dwErrorCode1	dwErrorCode2	bCycle	szApplicationName	szFunctionName	szTagName	lpParamSize	dwParamSize	dwCycle	pError	description
1007001	0	x	x	x		x	x			Action requires exception
1007001	1	x	x	x		x	x			Exception when accessing the return result
1007001	4097	x	x	x		x	x			Stack overflow while executing the action
1007001	4098	x	x	x		x	x			The action contains a division by 0
1007001	4099	x	x	x		x	x			The action contains an access to a non-existing symbol
1007001	4100	x	x	x		x	x			The action contains an access violation
1007004	0	x	x	x						Function is not known
1007005	1	x	x							Action does not include a P code.
1007005	2	x	x							Incorrect function name
1007005	4	x	x	x		x	x			Return value type is invalid
1007005	32768f	x	x	x		x	x			Ciss Compiler error when loading the action

100700 6	0	x	x	x	x	x	x	x		Tag is not defined
100700 6	1	x	x	x	x	x	x	x		Tag timeout
100700 6	2	x	x	x	x	x	x	x	x	Tag cannot be returned in the desired format
100700 6	3	x	x	x	x	x	x	x	x	Tag returns status violation, status present in CMN_ERROR.dwError1
100700 7	1	x	x	x		x	x		x	Error with PDLRTGetProp
100700 7	2	x	x	x		x	x		x	Error with PDLRTSetProp
100700 7	3	x	x	x		x	x		x	Error with DM call

Error structure

The OnErrorExecute function uses the error structure to evaluate or to output error messages, if marked by an "x" in the pError column.

2.3.2 Structure definition CCAPTime (RT Professional)

```
typedef struct {
  DWORD dwCurrentThreadID; ThreadID of the current Thread
  DWORD dwCode;             Code
  BOOL bCycle;              cycle/acycle
  char* szApplicationName;  Name of the Application
  char* szFunctionName;     Name of the Function
  LPVOID lpParam;           Pointer to the Action-Stack
  DWORD dwParamSize;        size of the Action-Stack
  double dblTime;
  DWORD dwFlags;            flags
} CCAPTime;
```

Members

dwCode

The structure element dwCode provides information on calling of OnTime:

dwCode = 113	Call with time definition for each action
dwCode = 114	Call with time monitoring for each action

dwFlags

The structure element dwFlags provides information on the output type:

dwFlags = TRUE	The results are output to a file
dwFlags = FALSE	The results are output to the diagnostics window

2.3.3 Structure definition CMN_ERROR (RT Professional)

```
struct CMNERRORSTRUCT {
  DWORD      dwError1,
  DWORD      dwError2,
  DWORD      dwError3,
  DWORD      dwError4,
  DWORD      dwError5;
  TCHAR      szErrorText[MAX_ERROR_LEN];
}
CMN_ERROR
```

Description

The extended error structure contains the error code and an error text for the error that has occurred. Each application can use the error structure to evaluate or to output error messages.

Members

dwError1 .. dwError5

These entries can be used in any way by the API functions.

The API descriptions inform about the values the respective entries contain in case of an error. If not specified otherwise, the error codes are present in dwError1.

szErrorText

Buffer for the text description of the error cause

The content is determined from the resources and therefore language-dependent.

2.3.4 Structure definition DM_TYPEREF (RT Professional)

```
typedef struct {
    DWORD dwType;
    DWORD dwSize;
    char szTypeName[MAX_DM_TYPE_NAME + 1];
}
DM_TYPEREF;
```

Members

dwType

Specifies the tag type

dwType	PLC data type	HMI data type
DM_VARTYPE_BIT	Binary tag	Bool
DM_VARTYPE_SBYTE	Signed 8-bit value	Byte
DM_VARTYPE_BYTE	Unsigned 8-bit value	UByte
DM_VARTYPE_SWORD	Signed 16-bit value	Short
DM_VARTYPE_WORD	Unsigned 16-bit value	UShort
DM_VARTYPE_SDWORD	Signed 32-bit value	Integer
DM_VARTYPE_DWORD	Unsigned 32-bit value	UInteger
DM_VARTYPE_FLOAT	Floating-point number 32-bit IEEE 754	Float
DM_VARTYPE_DOUBLE	Floating-point number 64-bit IEEE 754	Double
DM_VARTYPE_TEXT_8	Text tag, 8-bit character set	Char
DM_VARTYPE_TEXT_16	Text tag, 16-bit character set	String
DM_VARTYPE_RAW	Raw data type	Raw
DM_VARTYPE_STRUCT	Structure tag	Struct
DM_VARTYPE_TEXTREF	Text reference tag	String

dwSize

Specifies the length of the data type in bytes.

szTypeName

In the case of structure tags, contains the name of the structure type

2.3.5 Structure definition DM_VAR_UPDATE_STRUCT (RT Professional)

```
typedef struct {  
    DM_TYPEREF dmTypeRef;  
    DM_VARKEY dmVarKey;  
    VARIANT dmValue;  
    DWORD dwState;  
}  
DM_VAR_UPDATE_STRUCT;
```

Members

dmTypeRef

Contains information on the data type. For performance reasons, nothing is entered into this structure in case of cyclic requests.

dmVarKey

Specifies the tags to be edited.

dmValue

Tag value

Upon access to the value of the VARIANT a ".u." has to be inserted between the name of the VARIANT and the name of the member.

Example

```
// Supply variant  
myVariant.vt = VT_I4;  
myVariant.u.lVal = 233;
```

A description of the data type VARIANT can be found in the associated documentation. The VARIANT dmValue must be initialized with VariantInit() before first use and enabled again with VariantClear(&dmValue) after use. For this reason, the structure DM_VAR_UPDATE_STRUCT must not be deleted with ZeroMemory() or memset().

dwState

Identifies the tag status.

2.3.6 Structure definition DM_VAR_UPDATE_STRUCTEX (RT Professional)

```
typedef struct {  
    DM_TYPEREF dmTypeRef;  
    DM_VARKEY dmVarKey;  
    VARIANT dmValue;  
    DWORD dwState;  
    DWORD dwQualityCode;  
}  
DM_VAR_UPDATE_STRUCTEX;
```

Members

dmTypeRef

Contains information on the data type. For performance reasons, nothing is entered into this structure in case of cyclic requests.

dmVarKey

Specifies the tags to be edited.

dmValue

Tag value

Upon access to the value of the VARIANT a ".u." has to be inserted between the name of the VARIANT and the name of the member.

Example

```
// Supply variant  
myVariant.vt = VT_I4;  
myVariant.u.lVal = 233;
```

A description of the data type VARIANT can be found in the associated documentation. The VARIANT dmValue must be initialized with VariantInit() before first use and enabled again with VariantClear(&dmValue) after use. For this reason, the structure DM_VAR_UPDATE_STRUCTEX must not be deleted with ZeroMemory() or memset().

dwState

Identifies the tag status.

dwQualityCode

Identifies the QualityCode tag.

2.3.7 Structure definition DM_VARKEY (RT Professional)

```
typedef struct {  
    DWORD dwKeyType;  
    DWORD dwID;  
    char szName[ MAX_DM_VAR_NAME + 1 ];  
    LPVOID lpvUserData;  
}  
DM_VARKEY;
```

Members

dwKeyType

Defines whether the tag is to be addressed by a key ID or by its name.

DM_VARKEY_ID Specification via key ID

DM_VARKEY_NAME Specification via tag name

dwID

Contains the key ID of the tags if dwKey type is set accordingly.

szName

Contains the name of the tag if dwKey type is set accordingly.

lpvUserData

Pointer to application-specific data

2.3.8 Structure definition LINKINFO (RT Professional)

```
typedef struct {  
    LINKTYPE LinkType;  
    DWORD dwCycle;  
    TCHAR szLinkName[256];  
}  
LINKINFO;
```

Members

LinkType

LinkType are enumeration constants defined in the "Trigger.h" file. They are to be integrated into your script with the "#include "Trigger.h" command and the corresponding enumeration constants.

BUBRT_LT_NOLINK	0	no shortcut
BUBRT_LT_VARIABLE_DIRECT	1	direct tag
BUBRT_LT_VARIABLE_INDIRECT	2	indirect tag
BUBRT_LT_ACTION	3	C action
BUBRT_LT_ACTION_WIZARD	4	Dynamic Dialog
BUB_LT_DIRECT_CONNECTION	5	Direct connection
BUBRT_LT_ACTION_WIZARD_INPROC	6	Dynamic Dialog

For the SetLink function, only the enumeration constants BUBRT_LT_VARIABLE_DIREKT and BUBRT_LT_VARIABLE_INDIRECT may be used. The GetLink function allows to return all listed enumeration constants.

dwCycle

Update cycle time

255	Picture cycle
235	Window Cycle
0	Upon change
1	250ms
2	500 ms
3	1 s
4	2 s
5	5s
6	10s
7	1min
8	5min
9	10min
10	1h
11-15	User cycle 1-5

szLinkName

Tag name

2.3.9 Structure definition MSG_FILTER_STRUCT (RT Professional)

```
typedef struct {
    CHAR          szFilterName[MSG_MAX_TEXTLEN+1];
    WORD          dwFilter;
    SYSTEMTIME    st[2];
    DWORD         dwMsgNr[2];
    DWORD         dwMsgClass;
    DWORD         dwMsgType[MSG_MAX_CLASS];
    DWORD         dwMsgState;
    WORD          wAGNr[2];
    WORD          wAGSubNr[2];
    DWORD         dwArchivMode;
    char          szTB[MSG_MAX_TB] [
MSG_MAX_TB_CONTENT+1]
    DWORD         dwTB;
    Double        dPValue[MSG_MAX_PVALUE] [2];
    DWORD         dwPValue[2];
    DWORD         dwMsgCounter[2];
    DWORD         dwQuickSelect;
}
MSG_FILTER_STRUCT;
```

Description

In this structure the criteria are specified.

Members

dwFilter

The filter conditions are defined by means of the following constants from the file "m_global.h":

MSG_FILTER_DATE_FROM	Date from
MSG_FILTER_DATE_TO	Date to
MSG_FILTER_TIME_FROM	Time from
MSG_FILTER_TIME_TO	Time to
MSG_FILTER_NR_FROM	Message number from
MSG_FILTER_NR_TO	Message number to
MSG_FILTER_CLASS	Message classes
MSG_FILTER_STATE	Message status
MSG_FILTER_AG_FROM	AS number from
MSG_FILTER_AG_TO	AS number to
MSG_FILTER_AGSUB_FROM	AG subnumber from
MSG_FILTER_AGSUB_TO	AG subnumber to
MSG_FILTER_TEXT	Message texts
MSG_FILTER_PVALUE	Process values

MSG_FILTER_COUNTER_FROM	Internal message counter from
MSG_FILTER_COUNTER_TO	Internal message counter to
MSG_FILTER_PROCESSMSG	Process messages
MSG_FILTER_SYSMSG	System messages
MSG_FILTER_BEDMSG	Operator messages
MSG_FILTER_DATE	Date from to
MSG_FILTER_TIME	Time from to
MSG_FILTER_NR	Message number from to
MSG_FILTER_VISIBLEONLY	Display visible messages
MSG_FILTER_HIDDENONLY	Display hidden messages

st

Date/time from - to

Where st[0] is the start time (from), st[1] the end time (to)

Assign these fields for the criteria: MSG_FILTER_DATE, MSG_FILTER_DATE_FROM, MSG_FILTER_DATE_TO, MSG_FILTER_TIME, MSG_FILTER_TIME_FROM, bzw. MSG_FILTER_TIME_TO

If the current time is needed to pass a SYSTEMTIME- parameter, use the GetLocalTime function and not GetSystemTime.. There is usually a considerable difference between these two functions.

dwMsgNr

Message number from - to

Where dwMsgNr[0] is start no. (from), dwMsgNr[1] the end no. (to)

Assign these fields for the criteria: MSG_FILTER_NR, MSG_FILTER_NR_FROM bzw. MSG_FILTER_NR_TO

dwMsgClass

Message classes bit-coded.

Assign this field for the criterion: MSG_FILTER_CLASS

dwMsgType

Message type per alarm class, bit-coded

Assign this field for the criterion: MSG_FILTER_CLASS

dwMsgState

Message status bit-coded.

Assign this field for the criterion: MSG_FILTER_STATE

wAGNr

AGNr from - to

Assign these fields for the criteria: MSG_FILTER_AG_FROM or MSG_FILTER_AG_TO

wAGSubNr

AGSubNr from - to

Assign this field for the criteria: MSG_FILTER_AGSUB_FROM or MSG_FILTER_AGSUB_TO

dwArchivMode

Logging / reporting

Must be assigned 0.

szTB

Texts of the text blocks

Assign these fields for the criterion: MSG_FILTER_TEXT

dwTB

Active text blocks (from - to, bit-coded)

Assign this field for the criterion: MSG_FILTER_TEXT

dPValue

Process values from - to

Assign these fields for the criterion: MSG_FILTER_PVALUE

dwPValue

Active process values (from - to, bit-coded)

Assign this field for the criterion: MSG_FILTER_PVALUE

dwMsgCounter

Internal message counter from - to

Assign these fields for the criteria: MSG_FILTER_COUNTER_FROM,
MSG_FILTER_COUNTER_TO

dwQuickSelect

Quick selection for hour, day, month

The parameter is reserved for future upgrades and must be preset to 0.

Assign this field for the criterion: MSG_FILTER_QUICKSELECT

LOWORD Type:

MSG_FILTER_QUICK_MONTH	Quick selection last n months
MSG_FILTER_QUICK_DAYS	Quick selection last n days
MSG_FILTER_QUICK_HOUR	Quick selection last n hours

HIWORD Number: 1...n

The end time of the quick selection refers to the current system time of the local computer.
The start time is calculated back n * (months, days, hours).

2.3.10 Structure definition MSG_RTDATA_STRUCT (RT Professional)

```
typedef struct {
    DWORD          dwMsgState;
    DWORD          dwMsgNr;
    SYSTEMTIME     stMsgTime;
    DWORD          dwTimeDiff;
    DWORD          dwCounter;
    DWORD          dwFlags;
    WORD           wPValueUsed;
    WORD           wTextValueUsed;
    double         dPValue[MSG_MAX_PVALUE];
    MSG_TEXTVAL_STRUCT mtTextValue[MSG_MAX_PVALUE];
}
MSG_RTDATA_STRUCT;
```

Members

dwMsgState

Message status

MSG_STATE_COME	0x00000001	Message came in
MSG_STATE_GO	0x00000002	Message went out
MSG_STATE_QUIT	0x00000003	Message acknowledged
MSG_STATE_LOCK	0x00000004	Message locked
MSG_STATE_UNLOCK	0x00000005	Alarm locked
MSG_STATE_QUIT_SYSTEM	0x00000010	Message acknowledged by system
MSG_STATE_QUIT_EMERGENCY	0x00000011	Emergency acknowledgement
MSG_STATE_QUIT_HORN	0x00000012	Alarm alert acknowledgment
MSG_STATE_COMEGO	0x00000013	Message came in and went out, only displayed in the "Current messages" display
MSG_STATE_UPDATE	0x00010000	Bit for message update
MSG_STATE_RESET	0x00020000	Bit for message reset
MSG_STATE_SUMTIME	0x00040000	Bit active for daylight savings time
MSG_STATE_INSTANCE	0x00080000	Bit for instance message (n messages of a no.)

dwMsgNr

Message number

stMsgTime

Date/Time: Telegram time depending on the calling function

dwTimeDiff

Duration coming/Telegram time in seconds

dwCounter

Internal message counter

dwFlags

Message flags in the database

MSG_FLAG_SUMTIME	0x00000001	Daylight savings time active
MSG_FLAG_COMMENT	0x00000002	Message has comments
MSG_FLAG_ARCHIV	0x00000004	Archiving
MSG_FLAG_PROTOCOL	0x00000008	Logging
MSG_FLAG_TEXTVALUES	0x00000010	Message has values accompanying the text
MSG_FLAG_TIMEINVALID	0x00000020	Bit for invalid date/time stamp
MSG_FLAG_INSTANCE	0x00000040	Instance message identification (185269)

wPValueUsed

Process values used, bit-coded. Every bit may only be set in one of the two structure elements "wPValueUsed" or "wTextValueUsed". An accompanying value may either be a number or a text.

wTextValueUsed

text values used, bit-coded. Every bit may only be set in one of the two structure elements "wPValueUsed" or "wTextValueUsed". An accompanying value may either be a number or a text.

Runtime API (RT Professional)

3.1 Runtime API (RT Professional)

Introduction

Runtime API describes the open programming interface of WinCC. By means of the API functions, you make use of the internal functions of WinCC in separate applications and access data of HMI tags or archive data.

Note

Siemens assumes no liability and makes no warranty that data and information transported via API interfaces are compatible with software of third parties.

We expressly point out that improper use of the API interface can result in data loss or production outage.

Examples:

- MSRTCreateMsgPlus(): Create alarm
- DMGetValue(): Get tag value

Requirements

- Programming environment is installed, e.g., MS Visual Studio
- WinCC Runtime Professional is installed.

Use

You use the API functions in the following places:

- Within WinCC: In user-defined C-functions and local C-scripts.
- Outside of WinCC: In Windows applications that are created in the C/C++ programming language. To use Runtime API in the programming languages C# or VB.net, you must program a corresponding conversion.

Include files and Lib files

In order to generate executable code, you require the Include files and Lib files. You can find these files in the directory:

- %ProgramFiles%\SIEMENS\Automation\SCADA-RT_V11\WinCC\include
- %ProgramFiles%\SIEMENS\Automation\SCADA-RT_V11\WinCC\lib

3.2 Data management functions (RT Professional)

3.2.1 Basics (RT Professional)

3.2.1.1 Overview of structures (RT Professional)

Overview

DM_CONNECTION_DATA (Page 1565)	Connection data
DM_CONNKEY (Page 1566)	List connection data (filter structure)
DM_CYCLE_INFO (Page 1567)	Update cycles
DM_DATA_SERVICE (Page 1568)	Data transport channel
DM_DIRECTORY_INFO (Page 1569)	Path and file names of the configuration data
DM_DLGOPTIONS (Page 1570)	Dialog specifications
DM_MACHINE_TABLE (Page 1572)	Format conversion
DM_PROJECT_INFO (Page 1573)	Project information
DM_SD_TARGET_APP (Page 1577)	Specification of applications
DM_SD_TARGET_MACHINE (Page 1576)	Computer information
DM_SEND_DATA_STRUCT (Page 1574)	Specification of partners for data exchange
DM_TYPEREF (Page 1578)	Tag type preference
DM_VAR_UPDATE_STRUCTEX (Page 1581)	Get tag values
DM_VAR_UPDATE_STRUCT (Page 1579)	Get tag values
DM_VARFILTER (Page 1584)	Selection of tags
DM_VARGRP_DATA (Page 1586)	Information about a tag group
DM_VARGRPKEY (Page 1587)	Specification of a tag group
DM_VARIABLE_DATA4 (Page 1590)	Including information about a tag group
DM_VARIABLE_DATA (Page 1588)	Including information about a tag group
DM_FORMAT_INFO (Page 1571)	Format conversion
DM_VARKEY (Page 1592)	Specification of a tag
DM_VARLIMIT (Page 1595)	Tag limits
MCP_NEWVARIABLE_DATA_4 (Page 1598)	Tag definition
MCP_NEWVARIABLE_DATA_5 (Page 1600)	Tag definition
MCP_NEWVARIABLE_DATA_EX4 (Page 1604)	Tag definition
MCP_NEWVARIABLE_DATA_EX (Page 1602)	Tag definition
MCP_NEWVARIABLE_DATA (Page 1596)	Tag definition
MCP_VARIABLE_COMMON_EX (Page 1608)	Tag description
MCP_VARIABLE_COMMON (Page 1606)	Tag description
MCP_VARIABLE_LIMITS_EX (Page 1614)	Tag limits
MCP_VARIABLE_LIMITS5 (Page 1612)	Tag limits
MCP_VARIABLE_LIMITS (Page 1610)	Tag limits
MCP_VARIABLE_PROTOCOL_EX (Page 1617)	Tag limit handling

MCP_VARIABLE_PROTOCOL (Page 1615)	Tag limit handling
MCP_VARIABLE_SCALES (Page 1618)	Tag scaling

3.2.1.2 Overview of the functions (RT Professional)

Overview

DM_ENUM_CYCLES_PROC (Page 1640)	List update cycles (callback)
DM_ENUM_FORMATS_PROC (Page 1637)	List format conversions (callback)
DM_NOTIFY_PROC (Page 1631)	Notification function
DMActivateRTProject (Page 1619)	Activate project
DMChangeDataLocale (Page 1627)	Language switch (notify applications)
DMConnect (Page 1628)	Make connection to the Data Manager
DMDeactivateRTProject (Page 1633)	Deactivate runtime project
DMDisconnect (Page 1634)	Terminate connection to the Data Manager
DMEnumNumberFormats (Page 1636)	List format conversions
DMEnumUpdateCycles (Page 1639)	List update cycles
DMExitWinCCEx (Page 1643)	Exit WinCC with defaults
DMExitWinCC (Page 1642)	Exit WinCC
DMGetConnectionState (Page 1646)	Query connection to the Data Manager
DMGetHotkey (Page 1648)	Get hotkey ID
DMGetMachineInfo (Page 1649)	Query startup parameters of an application
DMGetMachineTable (Page 1650)	Query computer list

See also

DMGetDataLocale (Page 1647)

3.2.1.3 Quality codes of HMI tags (RT Professional)

Introduction

The "Quality Code" is required to evaluate the status and quality of a tag. The quality of the entire value transfer and value processing of the respective HMI tag is summarized in the indicated Quality Code. For example, it is possible to determine from the Quality Code whether the current value is a start value or substitute value.

The quality codes are prioritized. If several codes occur at the same time, the Quality Code reflecting the lowest quality is displayed.

Evaluation of Quality Codes

You can evaluate the Quality Code in a number of different ways:

- Evaluation with VB functions
- Evaluation with C functions
- Evaluation using the "Quality Code changed" event of an I/O field.

Note

In order to include the entire value transfer and value processing in the Quality Code for a process tag, the connected automation system must support the Quality Code. When configuring the PLC tags in the automation system, make sure there is enough memory space for the Quality Code. In an S7 automation system, for example, the Quality Code requires an additional byte, which is appended to the process value. To prevent errors, this byte must be taken into account when configuring a tag, for instance at the end of a data block.

Structure

The Quality Code has the following binary structure:

QQSSSSL

Q: Quality

S: Substatus of the quality

L: Limits. This value is optional.

Note

The Quality Code shown in the "Quality" table are base values for the quality stages. Making use of the substatus and limit elements gives rise to intermediate values over and above the quality stage concerned.

Quality

The first two digits specify the quality of the tag.

	Q	Q	S	S	S	S	L	L
	2	2	2	2	2	2	2	2
	7	6	5	4	3	2	1	0
Bad - The value is not useful	0	0	-	-	-	-	-	-
Uncertain - The quality of the value is less than normal, but the value may still be useful.	0	1	-	-	-	-	-	-
Good (Non-Cascade) - The quality of the value is good. Possible alarm conditions may be indicated by the sub-status.	1	0	-	-	-	-	-	-
Good (Cascade) - The value may be used in control.	1	1	-	-	-	-	-	-

Substatus

The quality alone is not enough. Individual qualities are divided into substatuses. The Quality Code is binary-coded. The value must be converted to hexadecimal format for analysis of the Quality Code.

Quality Codes of Tags

Possible Quality Code are listed in the following table. At the top of the table, you find the Quality Code reflecting the lowest quality, while the Quality Code reflecting the highest quality is shown at the bottom of the table. The Quality Code representing the highest quality is assigned the lowest priority, while the Quality Code representing the lowest quality is assigned the highest priority. If several statuses occur for one tag in the process, the poorest code is passed on.

Code (Hex)	Quality		Q	Q	S	S	S	S	L	L
0x23	Bad	Device passivated - Diagnostic alerts inhibited	0	0	1	0	0	0	1	1
0x3F	Bad	Function check - Local override	0	0	1	1	1	1	1	1
0x1C	Bad	Out of Service - The value is not reliable because the block is not being evaluated, and may be under construction by a configurer. Set if the block mode is O/S.	0	0	0	1	1	1	-	-
0x73	Uncertain	Simulated value - Start	0	1	1	1	0	0	1	1
0x74	Uncertain	Simulated value - End	0	1	1	1	0	1	-	-
0x84	Good (Non-Cascade)	Active Update event - Set if the value is good and the block has an active Update event.	1	0	0	0	0	1	-	-
0x24	Bad	Maintenance alarm - More diagnostics available.	0	0	1	0	0	1	-	-
0x18	Bad	No Communication, with no usable value - Set if there has never been any communication with this value since it was last "Out of Service".	0	0	0	1	1	0	-	-
0x14	Bad	No Communication, with last usable value - Set if this value had been set by communication, which has now failed.	0	0	0	1	0	1	-	-
0x0C	Bad	Device Failure - Set if the source of the value is affected by a device failure.	0	0	0	0	1	1	-	-
0x10	Bad	Sensor failure	0	0	0	1	0	0	-	-
0x08	Bad	Not Connected - Set if this input is required to be connected and is not connected.	0	0	0	0	1	0	-	-
0x04	Bad	Configuration Error - Set if the value is not useful because there is some inconsistency regarding the parameterization or configuration, depending on what a specific manufacturer can detect.	0	0	0	0	0	1	-	-
0x00	Bad	non-specific - There is no specific reason why the value is bad. Used for propagation.	0	0	0	0	0	0	-	-
0x28	Bad	Process related - Substitute value	0	0	1	0	1	0	-	-
0x2B	Bad	Process related - No maintenance	0	0	1	0	1	0	1	1
0x68	Uncertain	Maintenance demanded	0	1	1	0	1	0	-	-
0x60	Uncertain	Simulated value - Set when the process value is written by the operator while the block is in manual mode.	0	1	1	0	0	0	-	-
0x64	Uncertain	Sensor calibration	0	1	1	0	0	1	-	-

3.2 Data management functions (RT Professional)

Code (Hex)	Quality		Q	Q	S	S	S	S	L	L
0x5C	Uncertain	Configuration error	0	1	0	1	1	1	-	-
0x58	Uncertain	Sub-normal	0	1	0	1	1	0	-	-
0x54	Uncertain	Engineering Unit Range Violation - Set if the value lies outside of the set of values defined for this parameter. The Limits define which direction has been exceeded.	0	1	0	1	0	1	-	-
0x50	Uncertain	Sensor conversion not accurate	0	1	0	1	0	0	-	-
0x4B	Uncertain	Substitute (constant)	0	1	0	0	1	0	1	1
0x78	Uncertain	Process related - No maintenance	0	1	1	1	1	0	-	-
0x4C	Uncertain	Initial Value - Value of volatile parameters during and after reset of the device or of a parameter.	0	1	0	0	1	1	-	-
0x48	Uncertain	Substitute value - Predefined value is used instead of the calculated one. This is used for fail safe handling.	0	1	0	0	1	0	-	-
0x44	Uncertain	Last Usable Value - Whatever was writing this value has stopped doing so. This is used for fail safe handling.	0	1	0	0	0	1	-	-
0x40	Uncertain	Non-specific - There is no specific reason why the value is uncertain. Used for propagation.	0	1	0	0	0	0	-	-
0xE0	Good (Cascade)	Initiate Fail Safe (IFS) - The value is from a block that wants its downstream output block (e.g. AO) to go to Fail Safe.	1	1	1	0	0	0	-	-
0xD8	Good (Cascade)	Local Override (LO) - The value is from a block that has been locked out by a local key switch or is a Complex AO/DO with interlock logic active. The failure of normal control must be propagated to a function running in a host system for alarm and display purposes. This also implies "Not Invited".	1	1	0	1	1	0	-	-
0xD4	Good (Cascade)	Do Not Select (DNS) - The value is from a block which should not be selected, due to conditions in or above the block.	1	1	0	1	0	1	-	-
0xCC	Good (Cascade)	Not Invited (NI) - The value is from a block which does not have a target mode that would use this input.	1	1	0	0	1	1	-	-
0xC8	Good (Cascade)	Initialization Request (IR) - The value is an initialization value for a source (back calculation input parameter), because the lower loop is broken or the mode is wrong.	1	1	0	0	1	0	-	-
0xC4	Good (Cascade)	Initialization Acknowledge (IA) - The value is an initialized value from a source (cascade input, remote-cascade in, and remote-output in parameters).	1	1	0	0	0	1	-	-
0xC0	Good (Cascade)	OK - No error or special condition is associated with this value.	1	1	0	0	0	0	-	-
0xA0	Good (Non-Cascade)	Initiate fail safe	1	0	1	0	0	0	-	-
0x98	Good (Non-Cascade)	Unacknowledged Critical Alarm - Set if the value is good and the block has an unacknowledged Alarm with a priority greater than or equal to 8.	1	0	0	1	1	0	-	-
0x94	Good (Non-Cascade)	Unacknowledged Advisory Alarm - Set if the value is good and the block has an unacknowledged Alarm with a priority less than 8.	1	0	0	1	0	1	-	-

Code (Hex)	Quality		Q	Q	S	S	S	S	L	L
0x90	Good (Non-Cascade)	Unacknowledged Update event - Set if the value is good and the block has an unacknowledged Update event.	1	0	0	1	0	0	-	-
0x8C	Good (Non-Cascade)	Active Critical Alarm - Set if the value is good and the block has an active Alarm with a priority greater than or equal to 8.	1	0	0	0	1	1	-	-
0x88	Good (Non-Cascade)	Active Advisory Alarm - Set if the value is good and the block has an active Alarm with a priority less than 8.	1	0	0	0	1	0	-	-
0xA8	Good (Non-Cascade)	Maintenance demanded	1	0	1	0	1	0	-	-
0xA4	Good (Non-Cascade)	Maintenance required	1	0	1	0	0	1	-	-
0xBC	Good (Non-Cascade)	Function check - Local override	1	0	1	1	1	1	-	-
0x80	Good (Non-Cascade)	OK - No error or special condition is associated with this value.	1	0	0	0	0	0	-	-

Limit

The Quality Codes can be further subdivided by limits. Limits are optional.

	Q	Q	S	S	S	S	L	L
O.K. - The value is free to move.	-	-	-	-	-	-	0	0
Low limited - The value has acceded its low limits.	-	-	-	-	-	-	0	1
High limited - The value has acceded its high limits.	-	-	-	-	-	-	1	0
Constant (high and low limited) - The value cannot move, no matter what the process does.	-	-	-	-	-	-	1	1

Quality Codes in Communication with OPC

For communication via "OPC", the Quality Codes that the OPC does not support are converted.

Quality Code in WinCC	Quality Code in OPC
0x48	0x40
0x4C	0x40
0x5C	0x40
0x60	0x40
0x80...0xD4	0xC0
0xD8	0xC0

3.2.1.4 Constants (RT Professional)

General definitions

MAX_DM_OHIO_MACHINES	64	Maximum number of PCs in a project
MAX_DM_OHIO_APPLICATIONS	32	Maximum number of local client applications
MAX_DM_UPDATE_CYCLES	15	Maximum number of update cycles
MAX_DM_SYSTEM_CYCLES	10	Of total update cycles, 10 are system cycles
MAX_DM_USER_CYCLES	5	and 5 are user cycles
MAX_DM_VAR_NAME	128	Maximum length of a tag name
MAX_DM_TYPE_NAME	128	Maximum length of a data type name
MAX_DM_GROUP_NAME	64	Maximum length of a tag group name
MAX_DM_CYCLE_NAME	64	Maximum length of a cycle time text
MAX_DM_FORMAT_NAME	64	Maximum length of a format specification
MAX_DM_SCALE_NAME	64	Maximum length of a scaling specification
MAX_DM_SCALE_PARAM_NAME	64	Maximum length of a scaling parameter description
MAX_DM_MEMBER_NAME	32	Maximum length for members of combined types
MAX_DM_INFOTEXT_LEN	255	Maximum length of an info text
MAX_DM_SHIFT_NAME_LEN	32	Maximum length of a shift name
MAX_DM_SHIFTS	16	Maximum number of shifts per day
MAX_DM_SHIFT_HOLIDAYS	30	Maximum number of holidays in shift schedule
MAX_DM_SHIFT_HOLYNAME	64	Maximum length of the name of a holiday
MAX_DM_SERVICE_NAME	32	Maximum length of a service name
MAX_DM_APP_NAME	32	Maximum length of a logical application name
MAX_DM_DSN_NAME	32	Maximum length of data source name of the database
MAX_DM_UNIT_NAME	65	Maximum length of a unit
MAX_DM_CONNECTION_NAME	32	Maximum length of a connection
MAX_DM_VAR_SPECIFIC	25	Maximum length of the specific part of a tag in GAPI
MAX_DM_CON_SPECIFIC	128	Maximum length of the specific part of a connection in GAPI
MAX_DM_CON_COMMON	128	Maximum length of the specific part of a connection in GAPI
MAX_DM_VARTYPE_TEXT_LEN	255	Maximum length of a text tag

Tag types

DM_VARTYPE_BIT	1	Binary tag
DM_VARTYPE_SBYTE	2	Signed 8-bit value
DM_VARTYPE_BYTE	3	Unsigned 8-bit value
DM_VARTYPE_SWORD	4	Signed 16-bit value
DM_VARTYPE_WORD	5	Unsigned 16-bit value
DM_VARTYPE_SDWORD	6	Signed 32-bit value
DM_VARTYPE_DWORD	7	Unsigned 32-bit value
DM_VARTYPE_FLOAT	8	Floating-point number 32-bit IEEE 754

DM_VARTYPE_DOUBLE	9	Floating-point number 64-bit IEEE 754
DM_VARTYPE_TEXT_8	10	Text tag, 8-bit character set
DM_VARTYPE_TEXT_16	11	Text tag, 16-bit character set
DM_VARTYPE_RAW	12	Raw data type
DM_VARTYPE_STRUCT	14	Structure tag
DM_VARTYPE_TEXTREF	18	Text reference from text lib

Property flags

DM_SIMULATION_MODE	0x00000001	Tag is being simulated (no write access to the process)
DM_INTERNAL_VAR	0x00000002	Internal tag
DM_EXTERNAL_VAR	0x00000004	External tag
DM_HAS_MIN_LIMIT	0x00000008	Tag has a fixed low limit
DM_HAS_MAX_LIMIT	0x00000010	Tag has a fixed high limit
DM_HAS_DEFAULT_VALUE	0x00000020	Tag has a substitute value
DM_HAS_STARTUP_VALUE	0x00000040	Tag has an initial value
DM_USE_DEFAULT_ON_STARTUP	0x00000080	Enter substitute value at system start
DM_USE_DEFAULT_ON_MAX	0x00000100	Enter substitute value when high limit is violated
DM_USE_DEFAULT_ON_MIN	0x00000200	Enter substitute value when low limit is violated
DM_USE_DEFAULT_ON_COMM_ERROR	0x00000400	Substitute value on connection error
DM_WRITE_ACCESS_APPLICATION	0x00000800	Applications may write the tag
DM_WRITE_ACCESS_PROCESS	0x00001000	Tag may be written by the process
DM_INTERNAL_VAR_LOKAL	0x00002000	Internal tag, update 1 = Local computer / 0 = Project-wide
DM_INVISIBLE	0x00004000	Do not display tag on the interface in Control Center
DM_EXTERNAL_LOCK	0x00008000	Tag generated externally and can only be deleted externally

Logging flags

DM_NOTIFY_MAX_LIMIT	0x00000001	Log entry on reaching the high limit
DM_NOTIFY_MIN_LIMIT	0x00000002	Log entry on reaching the low limit
DM_NOTIFY_FORMAT_ERROR	0x00000004	Log entry on conversion error
DM_NOTIFY_ACCESS_FAULT	0x00000008	Log entry on prohibited write access
DM_NOTIFY_APPLICATION_WRITE	0x00000010	Log entry on write access by application
DM_NOTIFY_PROCESS_WRITE	0x00000020	Log entry on write access by process

Status flags

You can monitor the tag status of individual WinCC tags in Runtime. The tag status also includes violations of the configured measuring range limits as well as the status of the connection between WinCC and the automation system.

The quality code includes information about the quality of a tag, regardless of where this code was formed. It takes into consideration the status of the entire value transfer and value processing.

If a violation occurs at the low limit of the measuring range in the system, for example, the quality code "0x55" is always reported. This measuring range violation can occur in the WinCC data manager as well as in the field device. You can use the tag status to determine whether this measuring range violation occurred in WinCC or before the value was transferred to WinCC.

If the tag status reports a limit violation with the code 0x0010, for example, the low range limit configured in WinCC has been violated. If the tag status does not report a limit violation, the quality code transferred to WinCC already contained the limit violation.

	0x0000	No error
DM_VARSTATE_NOT_ESTABLISHED	0x0001	Connection to partner not established
DM_VARSTATE_HANDSHAKE_ERROR	0x0002	Protocol error
DM_VARSTATE_HARDWARE_ERROR	0x0004	Network module defective
DM_VARSTATE_MAX_LIMIT	0x0008	Configured high limit violated
DM_VARSTATE_MIN_LIMIT	0x0010	Configured low limit violated
DM_VARSTATE_MAX_RANGE	0x0020	Format limit exceeded
DM_VARSTATE_MIN_RANGE	0x0040	Format limit fallen below
DM_VARSTATE_CONVERSION_ERROR	0x0080	Display conversion error (in conjunction with DM_VARSTATE_..._RANGE))
DM_VARSTATE_STARTUP_VALUE	0x0100	Initial value of tag
DM_VARSTATE_DEFAULT_VALUE	0x0200	Substitute value of tag
DM_VARSTATE_ADDRESS_ERROR	0x0400	Addressing error in channel
DM_VARSTATE_INVALID_KEY	0x0800	Tag not found/not available
DM_VARSTATE_ACCESS_FAULT	0x1000	Access to tag denied
DM_VARSTATE_TIMEOUT	0x2000	Timeout/no feedback from channel
DM_VARSTATE_SERVERDOWN	0x4000	Server is down

Notification classes

DM_NOTIFY_CLASS_ERROR	0x00000001	Notification code contains error ID
DM_NOTIFY_CLASS_WARNING	0x00000002	Notification code contains warning
DM_NOTIFY_CLASS_DATA	0x00000003	Notification code contains data type

Notification codes

DM_NOTIFY_SHUTDOWN	0x00000001	Class Error : Data Manager is shutting down
DM_NOTIFY_PROCESSNET_ERROR	0x00000002	Class Error : Error on process bus

DM_NOTIFY_SYSNET_ERROR	0x00000003	Class Error : Error on system bus
DM_NOTIFY_QUEUE_50_PERCENT	0x00000001	Class Warning : Fill level of application queue 50%
DM_NOTIFY_QUEUE_60_PERCENT	0x00000002	Class Warning : Fill level of application queue 60%
DM_NOTIFY_QUEUE_70_PERCENT	0x00000003	Class Warning : Fill level of application queue 70%
DM_NOTIFY_QUEUE_80_PERCENT	0x00000004	Class Warning : Fill level of application queue 80%
DM_NOTIFY_QUEUE_90_PERCENT	0x00000005	Class Warning : Fill level of application queue 90%
DM_NOTIFY_QUEUE_OVERFLOW	0x00000006	Class Warning : Application queue over-flow
DM_NOTIFY_CYCLES_CHANGED	0x00000010	Class Warning => Read in cycles again
DM_NOTIFY_MACHINES_CHANGED	0x00000011	Class Warning : Computer list
DM_NOTIFY_PROJECT_OPENED	0x00000012	Class Warning: Project is being loaded
DM_NOTIFY_PROJECT_CLOSED	0x00000013	Class Warning: Project is being closed
DM_NOTIFY_SYSTEM_LOCALE	0x00000014	Switch language (resources)
DM_NOTIFY_DATA_LOCALE	0x00000015	Switch language (configuration data)
DM_NOTIFY_PROJECT_RUNTIME	0x00000016	Project activated in runtime mode
DM_NOTIFY_PROJECT_EDIT	0x00000017	Runtime mode for project deactivated
DM_NOTIFY_HOTKEY_CHANGE	0x00000018	The hotkeys have changed
DM_NOTIFY_APPLICATION_DATA	0x00000001	Class Data : Data sent by application
DM_NOTIFY_VARIABLE_DATA	0x00000002	Class Data : Tag data
DM_NOTIFY_SERVERDOWN	0x00000022	
DM_NOTIFY_SERVERUP	0x00000023	

Computer type

Computer type	0	Server
	1	Client

Registered messages

DM_OHIOLANGUAGE "WM_OHIOLANGUAGE"

Internal system tag

DM_VAR_CURRENT_LANGUAGE "OHIOCurrentLanguage"

3.2.1.5 Error messages (RT Professional)

Overview

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

DM_E_SYS_ERROR	0x10000000	If this bit is set in dwError1 of error structure CMN_ERROR, then dwError2 contains the system error code.
DM_E_OK	0x00000000	No error
DM_E_CANCEL	0x00000001	User has selected "Cancel" in a dialog
DM_E_FILE	0x00000002	Error during file operation
DM_E_UPDATE	0x00000003	Updating project
DM_E_NO_RT_PRJ	0x00000004	No project in runtime mode
DM_E_NOT_SUPPORTED	0x00000005	Requested service unavailable
DM_E_ALREADY_CONNECTED	0x00000006	Connection to the Data Manager already established
DM_E_NOT_CONNECTED	0x00000007	No connection to the Data Manager
DM_E_INVALID_TAID	0x00000008	Invalid transaction ID
DM_E_INVALID_KEY	0x00000009	Tag not found
DM_E_INVALID_TYPE	0x0000000A	Invalid tag type
DM_E_MAX_LIMIT	0x0000000B	High limit of the tag violated
DM_E_MIN_LIMIT	0x0000000C	Low limit of the tag violated
DM_E_MAX_RANGE	0x0000000D	High limit of the format conversion violated
DM_E_MIN_RANGE	0x0000000E	Low limit of the format conversion violated
DM_E_ACCESS_FAULT	0x0000000F	Prohibited write access to a tag
DM_E_TIMEOUT	0x00000010	Timeout error
DM_E_ALREADY_EXIST	0x00000011	Object to be created already exists
DM_E_PARAM	0x00000012	Invalid parameter
DM_E_INV_PRJ	0x00000013	Cannot find/load specified project
DM_E_UNKNOWN	0x00000014	Unknown error
DM_E_OOM	0x00000015	Out of memory
DM_E_NOT_CREATED	0x00000016	Project could not be created
DM_E_MACHINE_NOT_FOUND	0x00000017	Computer not found
DM_E_NO_INFO_FOUND	0x00000018	Cannot find startup information
DM_E_INTERNAL	0x00000019	Internal processing error
DM_E_INVALID_LOCALE	0x0000001A	Incorrect local ID
DM_E_COMMUNICATION	0x0000001B	Incorrect local ID
DM_E_DONT_EXIST	0x0000001C	Object does not exist e.g.: attempt was made to create a tag, and specified connection does not exist

DM_E_ALREADY_ACTIVATED	0x0000001D	Project is in runtime mode e.g.: attempt is made to activate a project that is already in runtime mode.
DM_E_NO_OPEN_PROJECT	0x0000001E	No project is open.
DM_E_ALREADY_DEACTIVATED	0x0000001F	Project is not in runtime mode
DM_E_NO_RIGHTS	0x00000020	Wrong CreatorID transferred
DM_E_NOT_DELETED	0x00000021	Object could not be deleted
DM_E_LICENSE	0x00000022	Software protection: No license
DM_E_LICENSE_LIMIT	0x00000023	Software protection: Limit reached/ exceeded
DM_E_INVALID_OBJECTTYPE	0x00000024	Invalid object type
DM_E_OP_REQUIERES_PRJEDITMODE	0x00000025	Project is in runtime mode e.g.: operation requires that the project is not activated.
DM_E_INTERFACE	0x00000026	Internal error while accessing interfaces
DM_E_UNIT_NOT_FOUND	0x00000027	Unit not found
DM_E_CONNECTION_NOT_FOUND	0x00000028	Connection not found

See also

CMN_ERROR (Page 2488)

3.2.1.6 Conversion routines (Control Center) (RT Professional)

Overview

No.	Conversion routines for the "Signed 8-bit value" data type
0	CharToSignedByte
1	CharToUnsignedByte
2	CharToUnsignedWord
3	CharToUnsignedDword
4	CharToSignedWord
5	CharToSignedDword
6	CharToMSBByte
7	CharToMSBWord
8	CharToMSBDword
9	CharToBCDByte
10	CharToBCDWord
11	CharToBCDDword
12	CharToSignedBCDByte
13	CharToSignedBCDWord
14	CharToSignedBCDDword
15	CharToExtSignedBCDByte

3.2 Data management functions (RT Professional)

16	CharToExtSignedBCDWord
17	CharToExtSignedBCDDword
18	CharToAikenByte
19	CharToAikenWord
20	CharToAikenDword
21	CharToSignedAikenByte
22	CharToSignedAikenWord
23	CharToSignedAikenDword
24	CharToExcessByte
25	CharToExcessWord
26	CharToExcessDword
27	CharToSignedExcessByte
28	CharToSignedExcessWord
29	CharToSignedExcessDword
> 29	(CharToSignedByte)

No.	Conversion routines for the "Unsigned 8-bit value" data type
0	ByteToUnsignedByte
1	ByteToUnsignedWord
2	ByteToUnsignedDword
3	ByteToSignedByte
4	ByteToSignedWord
5	ByteToSignedDword
6	ByteToBCDByte
7	ByteToBCDWord
8	ByteToBCDDword
9	ByteToAikenByte
10	ByteToAikenWord
11	ByteToAikenDword
12	ByteToExcessByte
13	ByteToExcessWord
14	ByteToExcessDword
> 14	(ByteToUnsignedByte)

No.	Conversion routines for the "Signed 16-bit value" data type
0	ShortToSignedWord
1	ShortToUnsignedByte
2	ShortToUnsignedWord
3	ShortToUnsignedDword
4	ShortToSignedByte
5	ShortToSignedDword

3.2 Data management functions (RT Professional)

6	ShortToMSBByte
7	ShortToMSBWord
8	ShortToMSBDword
9	ShortToBCDByte
10	ShortToBCDWord
11	ShortToBCDDword
12	ShortToSignedBCDByte
13	ShortToSignedBCDWord
14	ShortToSignedBCDDword
15	ShortToExtSignedBCDByte
16	ShortToExtSignedBCDWord
17	ShortToExtSignedBCDDword
18	ShortToAikenByte
19	ShortToAikenWord
20	ShortToAikenDword
21	ShortToSignedAikenByte
22	ShortToSignedAikenWord
23	ShortToSignedAikenDword
24	ShortToExcessByte
25	ShortToExcessWord
26	ShortToExcessDword
27	ShortToSignedExcessByte
28	ShortToSignedExcessWord
29	ShortToSignedExcessDword
> 29	(ShortToSignedWord)

No.	Conversion routines for the "Unsigned 16-bit value" data type
0	WordToUnsignedWord
1	WordToUnsignedByte
2	WordToUnsignedDword
3	WordToSignedByte
4	WordToSignedWord
5	WordToSignedDword
6	WordToBCDByte
7	WordToBCDWord
8	WordToBCDDword
9	WordToAikenByte
10	WordToAikenWord
11	WordToAikenDword
12	WordToExcessByte
13	WordToExcessWord
14	WordToExcessDword

3.2 Data management functions (RT Professional)

15	WordToSimaticBCDCounter
16	WordToSimaticCounter
> 16	(WordToUnsignedWord)

No.	Conversion routines for the "Signed 32-bit value" data type
0	LongToSignedDword
1	LongToUnsignedByte
2	LongToUnsignedWord
3	LongToUnsignedDword
4	LongToSignedByte
5	LongToSignedWord
6	LongToMSBByte
7	LongToMSBWord
8	LongToMSBDword
9	LongToBCDByte
10	LongToBCDWord
11	LongToBCDDword
12	LongToSignedBCDByte
13	LongToSignedBCDWord
14	LongToSignedBCDDword
15	LongToExtSignedBCDByte
16	LongToExtSignedBCDWord
17	LongToExtSignedBCDDword
18	LongToAikenByte
19	LongToAikenWord
20	LongToAikenDword
21	LongToSignedAikenByte
22	LongToSignedAikenWord
23	LongToSignedAikenDword
24	LongToExcessByte
25	LongToExcessWord
26	LongToExcessDword
27	LongToSignedExcessByte
28	LongToSignedExcessWord
29	LongToSignedExcessDword
30	LongToSimaticBCDTimer
31	(LongToSignedDword)
32	(LongToSignedDword)
33	(LongToSignedDword)
34	LongToSimaticTimer
> 34	(LongToSignedDword)

No.	Conversion routines for the "Unsigned 32-bit value" data type
0	DwordToUnsignedDword
1	DwordToUnsignedByte
2	DwordToUnsignedWord
3	DwordToSignedByte
4	DwordToSignedWord
5	DwordToSignedDword
6	DwordToBCDByte
7	DwordToBCDWord
8	DwordToBCDDword
9	DwordToAikenByte
10	DwordToAikenWord
11	DwordToAikenDword
12	DwordToExcessByte
13	DwordToExcessWord
14	DwordToExcessDword
15	DwordToSimaticBCDTimer
16	(DwordToUnsignedDword)
17	(DwordToUnsignedDword)
18	(DwordToUnsignedDword)
19	DwordToSimaticTimer
> 19	(DwordToUnsignedDword)

No.	Conversion routines for the "Floating-point number 32-bit IEEE 754" data type
0	FloatToFloat
1	FloatToUnsignedByte
2	FloatToUnsignedWord
3	FloatToUnsignedDword
4	FloatToSignedByte
5	FloatToSignedWord
6	FloatToSignedDword
7	FloatToDouble
8	FloatToMSBByte
9	FloatToMSBWord
10	FloatToMSBDword
11	FloatToBCDByte
12	FloatToBCDWord
13	FloatToBCDDword
14	FloatToSignedBCDByte

3.2 Data management functions (RT Professional)

15	FloatToSignedBCDWord
16	FloatToSignedBCDDword
17	FloatToExtSignedBCDByte
18	FloatToExtSignedBCDWord
19	FloatToExtSignedBCDDword
20	FloatToAikenByte
21	FloatToAikenWord
22	FloatToAikenDword
23	FloatToSignedAikenByte
24	FloatToSignedAikenWord
25	FloatToSignedAikenDword
26	FloatToExcessByte
27	FloatToExcessWord
28	FloatToExcessDword
29	FloatToSignedExcessByte
30	FloatToSignedExcessWord
31	FloatToSignedExcessDword
32	FloatToSimaticBCDTimer
33	(FloatToFloat)
34	(FloatToFloat)
35	(FloatToFloat)
36	FloatToS5Float
37	FloatToSimaticTimer
> 37	(FloatToFloat)

No.	Conversion routines for the "Floating-point number 64-bit IEEE 754" data type
0	DoubleToDouble
1	DoubleToUnsignedByte
2	DoubleToUnsignedWord
3	DoubleToUnsignedDword
4	DoubleToSignedByte
5	DoubleToSignedWord
6	DoubleToSignedDword
7	DoubleToFloat
8	DoubleToMSBByte
9	DoubleToMSBWord
10	DoubleToMSBDword
11	DoubleToBCDByte
12	DoubleToBCDWord
13	DoubleToBCDDword
14	DoubleToSignedBCDByte
15	DoubleToSignedBCDWord

16	DoubleToSignedBCDDword
17	DoubleToExtSignedBCDByte
18	DoubleToExtSignedBCDWord
19	DoubleToExtSignedBCDDword
20	DoubleToAikenByte
21	DoubleToAikenWord
22	DoubleToAikenDword
23	DoubleToSignedAikenByte
24	DoubleToSignedAikenWord
25	DoubleToSignedAikenDword
26	DoubleToExcessByte
27	DoubleToExcessWord
28	DoubleToExcessDword
29	DoubleToSignedExcessByte
30	DoubleToSignedExcessWord
31	DoubleToSignedExcessDword
32	DoubleToSimaticBCDTimer
33	(DoubleToDouble)
34	(DoubleToDouble)
35	(DoubleToDouble)
36	DoubleToS5Float
37	DoubleToSimaticTimer
> 37	(DoubleToDouble)

3.2.2 Structures (RT Professional)

3.2.2.1 DM_CONNECTION_DATA (RT Professional)

Declaration

```
typedef struct {
    CHAR          szConnection[
        MAX_DM_CONNECTION_NAME +3];
    CHAR          szUnitName[MAX_DM_UNIT_NAME +1];
    CHAR          szCommon[MAX_DM_CON_COMMON +1];
    CHAR          szSpecific[MAX_DM_CON_SPECIFIC +1];
    DWORD         dwVarNum;
}
DM_CONNECTION_DATA;
```

Members

- szConnection**
Name of the logical connection
- szUnitName**
Names of the channel unit
- szCommon**
The parameter is reserved for future development.
- szSpecific**
szSpecific contains the address parameters for the connection, e.g. Ethernet address, slot number.... Refer to the communication manual for PLC-specific details.

This is the same value that is displayed in the Parameter column of the properties of a tag in WinCC.
- dwVarNum**
Number of assigned tags

Required files

dmclient.h

API functions

DM_ENUM_CONNECTION_PROC (Page 1792)	List connection data (callback)

See also

DM_ENUM_CONNECTION_PROC (Page 1792)

3.2.2.2 DM_CONNKEY (RT Professional)

Declaration

```
typedef struct {  
    CHAR      szName[ MAX_DM_CONNECTION_NAME + 1 ];  
    LPVOID    lpvUserData;  
}  
DM_CONNKEY;
```

Members**szName**

Name of the logical connection

lpvUserData

Pointer to application-specific data

Required files

dmclient.h

API functions

DMEnumConnectionData (Page 1789)	List connection data

See also

DMEnumConnectionData (Page 1789)

3.2.2.3 DM_CYCLE_INFO (RT Professional)**Declaration**

```
typedef struct {
    DWORD    dwCycleTime;
    DWORD    dwCycleIndex;
    char     szDescription[ MAX_DM_CYCLE_NAME + 1 ];
}
DM_CYCLE_INFO;
```

Members**dwCycleTime**

Time base of the update cycle

dwCycleIndex

identifies the order within the list of update cycles.

"On change"	Index: 0
"250ms"	Index: 1
:	:
"User cycle 5"	Index: 15

szDescription

Description of the update cycle.

Required files

dmclient.h

API functions

DM_ENUM_CYCLES_PROC (Page 1640)	List update cycles (callback)

See also

DM_ENUM_CYCLES_PROC (Page 1640)

3.2.2.4 DM_DATA_SERVICE (RT Professional)**Declaration**

```
typedef struct {
    DWORD    dwTeleType;
    char      szService[MAX_DM_SERVICE_NAME + 1];
    char      szSendingApp[MAX_DM_APP_NAME + 1];
    DWORD    dwSendingMachine;
    DWORD    dwDataSize;
    BYTE      byData[1];
}
DM_DATA_SERVICE;
```

Members**dwTeleType**

The parameter is reserved for future development and must be preset to 0.

szService

Name of the data transport channel. This name corresponds to the one assigned during the installation of the service (DMInstallDataService).

szSendingApp

logical name of the sender. This name corresponds to the application name specified for DMConnect.

dwSendingMachine

Index of the computer from which the packet was sent (0 - .63). All computers registered in the computer list can be accessed by their index for the API functions. The first entry in the computer list has the index "0".

dwDataSize

Size of the data packet in bytes:Data[0] ... byData[dwDataSize - 1]

byData

Pointer to the data

Required files

dmclient.h

API functions

DM_DATA_SERVICE_PROC (Page 1668)	List data transport channels (callback), install data transport channel (callback)

See also

DM_DATA_SERVICE_PROC (Page 1668)

3.2.2.5 DM_DIRECTORY_INFO (RT Professional)**Declaration**

```
typedef struct {
    char    szProjectDir[_MAX_PATH + 1 ];
    char    szProjectAppDir[_MAX_PATH + 1 ];
    char    szGlobalLibDir[_MAX_PATH + 1 ];
    char    szProjectLibDir[_MAX_PATH + 1 ];
    char    szLokalProjectAppDir[_MAX_PATH + 1 ];
}
DM_DIRECTORY_INFO;
```

Members**szProjectDir**

Full path of the project directory, e.g. D:\WinCC\Projekt1

szProjectAppDir

Full path of the subdirectory of the application in the project directory, e.g.:D:\WinCC\Projekt1\GraCS

szGlobalLibDir

Full path of the cross-project library directory, e.g. D:\WinCC\aplib

szProjectLibDir

Full path of the project-based library directory, e.g. D:\WinCC\Projekt1\Library

szLokalProjectAppDir

Full path of the subdirectory of the application in the project directory on the local computer.

Required files

dmclient.h

API functions

DMGetProjectDirectory (Page 1657)	Get path and file names of the configuration data

See also

DMGetProjectDirectory (Page 1657)

3.2.2.6 DM_DLGOPTIONS (RT Professional)**Declaration**

```
typedef struct {
    DWORD                dwFlags;
    LPRECT               lprcPreference;
    DM_TEST_DROP_TARGET_PROC lpfnTestDropTarget;
    DM_DROP_TARGET_PROC   lpfnDropTarget;
    LPVOID               lpvUser;
}
DM_DLGOPTIONS;
```

Members**dwFlags**

dwFlags is used to specify the reaction of the dialog or dialog box:

DM_DLG_NOT_MODAL	The tag dialog is not modal.
DM_DLG_NOT_MOVEABLE	The dialog box is not movable.
DM_DLG_DRAGDROP	Objects in the dialog box can be edited with drag-and-drop.

lprcPreference

Pointer to a RECT type structure with information about the size of the dialog box. If lprcPreference == NULL, the dialog appears centered and with a predefined size.

lpfnTestDropTarget

The parameter is reserved for future development and must be preset to NULL.

lpfnDropTarget

The parameter is reserved for future development and must be preset to NULL.

lpvUser

Pointer to application-specific data.

Required files

dmclient.h

API functions

DMSHOWVARDATABASE (Page 1756)	Open tag selection dialog
DMSHOWVARDATABASEMULTI (Page 1761)	Open tag selection dialog

See also

DMSHOWVARDATABASE (Page 1756)

DMSHOWVARDATABASEMULTI (Page 1761)

3.2.2.7 DM_FORMAT_INFO (RT Professional)**Declaration**

```
typedef struct {
    DWORD    dwID;
    char     szName[MAX_DM_FORMAT_NAME + 1];
}
DM_FORMAT_INFO;
```

Members**dwID**

Number of the conversion routine to be used. You can find more detailed information in the "Conversion routines" section.

szName

Name of the conversion routine to be used. You can find more detailed information in the "Conversion routines" section.

Required files

dmclient.h

API functions

DM_ENUM_FORMATS_PROC (Page 1637)	List format conversions (callback)

See also

DM_ENUM_FORMATS_PROC (Page 1637)

3.2.2.8 DM_MACHINE_TABLE (RT Professional)**Declaration**

```
typedef struct {
    LONG                nNumMachines;
    LONG                nLocalMachine;
    DM_SD_TARGET_MACHINE tm[MAX_DM_OHIO_MACHINES];
}
DM_MACHINE_TABLE;
```

Members**nNumMachines**

Number of computers in the project (the max. number MAX_DM_OHIO_MACHINES may not be exceeded).

nLocalMachine

Index of the entry of the local computer in the computer list.

All computers registered in the computer list can be accessed by their index for the API functions. The first entry in the computer list has the index "0".

tm

Full DM_SD_TARGET_MACHINE (Page 1576) type array with information about the computers involved in the project. Only nNumMachines of these structures are passed here, however (therefore max. usable index = (nMumMachines - 1)).

Required files

dmclient.h

API functions

DMGetMachineTable (Page 1650)	Query computer list

See also

DM_SD_TARGET_MACHINE (Page 1576)

DMGetMachineTable (Page 1650)

3.2.2.9 DM_PROJECT_INFO (RT Professional)**Declaration**

```
typedef struct {
    char        szProjectFile[ _MAX_PATH + 1 ];
    char        szDSNName[ MAX_DM_DSN_NAME + 1 ];
    DWORD       dwDataLocale;
}
DM_PROJECT_INFO;
```

Members**szProjectFile**

File name of the project, including path and extension.

szDSNName

Data source name of the database

dwDataLocale

Code of the language that is used during configuration.

Required files

dmclient.h

API functions

DM_ENUM_OPENED_PROJECTS_PROC (Page 1655)	List projects (open) (callback)
DMEEnumOpenedProjects (Page 1654)	Get project information

See also

DM_ENUM_OPENED_PROJECTS_PROC (Page 1655)

DMEEnumOpenedProjects (Page 1654)

DMGetProjectInformation (Page 1658)

3.2.2.10 DM_SEND_DATA_STRUCT (RT Professional)**Declaration**

```
typedef struct {
    BOOL                fHighPriority;
    char                szService[MAX_DM_SERVICE_NAME + 1];
    DWORD              dwTargetMachineFlags;
    DWORD              dwTargetMachines;
    DM_SD_TARGET_MACHINE dmTargetMachine[
        MAX_DM_OHIO_MACHINES];
    DWORD              dwTargetApps;
    DM_SD_TARGET_APP   dmTargetApp[
        MAX_DM_OHIO_APPLICATIONS];
    DWORD              dwDataSize;
    BYTE               byData[1];
}
DM_SEND_DATA_STRUCT;
```

Members**fHighPriority**

fHighPriority indicates the priority for the data communication:

1	High priority
0	Normal priority

szService

Name of the service to be used. In order for an application to receive data, not only do the conditions of dwTargetMachineFlags have to be met, but also the corresponding service has to be installed through DMInstallDataService.

dwTargetMachineFlags

dwTargetMachineFlags indicate the applications to which data should be set with the DMSendApplicationData function.

DM_SD_LOCAL	Only to local applications (excludes all other flags!) All other flags are currently not allowed and are reserved for future development!
DM_SD_ALL_MACHINES	To all computers in the project. (The parameter is reserved for future development)

DM_SD_ALL_SERVERS	To all servers in the project. (The parameter is reserved for future development)
DM_SD_ALL_CLIENTS	To all clients in the project. (The parameter is reserved for future development)
DM_SD_RELATED_MACHINES	To all clients that belong to the same server as the local host (The parameter is reserved for future development)
DM_SD_FIRST_SERVER	(The parameter is reserved for future development)
DM_SD_PRIMARY_SERVER	(The parameter is reserved for future development)
DM_SD_EXCEPT_LOCAL	To all computers, except local ones, that meet the address description. (The parameter is reserved for future development)

dwTargetMachines

Number of completed structures dmTargetMachine.

Reserved for future development, must be filled with 0.

dmTargetMachine

Pointer to the DM_SD_TARGET_MACHINE (Page 1576) structure for specifying the computer to which the data is to be sent. Only the szMachineName parameter is relevant here.

Reserved for future development, should be fully initialized with 0L.

dwTargetApps

Number of completed structures dmTargetApp.

dmTargetApp

DM_SD_TARGET_APP (Page 1577) type structures contain the names of the applications to which the data is to be sent.

dwDataSize

Amount of data to be sent in bytes

byData

Array containing the data to be sent

Comments

The DMSendApplicationData function is only implemented locally.

The members present in this structure for remote access are reserved for future development. dwTargetMachineFlags can only be filled with DM_SD_LOCAL, any other specification results in errors, and dwTargetMachines must be 0L.

Required files

dmclient.h

API functions

DMSendApplicationData (Page 1672)	Data communication with applications

See also

DMSendApplicationData (Page 1672)

DM_SD_TARGET_MACHINE (Page 1576)

DM_SD_TARGET_APP (Page 1577)

3.2.2.11 DM_SD_TARGET_MACHINE (RT Professional)

Declaration

```
typedef struct {
    BOOL    fServer;
    BOOL    fLocal;
    char    szMachineName[
                MAX_COMPUTERNAME_LENGTH + 1];
}
DM_SD_TARGET_MACHINE;
```

Members

fServer

Indicates whether the computer at hand is a server or a client.

The parameter is not relevant for the purposes of data communication (DM_SEND_DATA_STRUCT).

fLocal

Indicates whether the querying application is running on the local computer or another computer configured in the network.

The parameter is not relevant for the purposes of data communication (DM_SEND_DATA_STRUCT).

szMachineName

Computer name

Comments

DM_SD_TARGET_MACHINE is part of the DM_MACHINE_TABLE (Page 1570) and DM_SEND_DATA_STRUCT (Page 1572) structures.

Required files

dmclient.h

See also

DM_MACHINE_TABLE (Page 1570)

DM_SEND_DATA_STRUCT (Page 1572)

3.2.2.12 DM_SD_TARGET_APP (RT Professional)**Declaration**

```
typedef struct {  
    char    szAppName[MAX_DM_APP_NAME + 1];  
}  
DM_SD_TARGET_APP;
```

Members**szAppName**

For szAppName, use the name of the application used for calling DMConnect.

Comments

DM_SD_TARGET_APP is part of the DM_SEND_DATA_STRUCT (Page 1572) structure.

Required files

dmclient.h

See also

DM_SEND_DATA_STRUCT (Page 1572)

3.2.2.13 DM_TYPEREF (RT Professional)

Declaration

```
typedef struct {
    DWORD    dwType;
    DWORD    dwSize;
    char     szTypeName[MAX_DM_TYPE_NAME + 1];
}
DM_TYPEREF;
```

Members

dwType

The type of tag is specified in dwType:

DM_VARTYPE_BIT	Binary tag
DM_VARTYPE_SBYTE	Signed 8-bit value
DM_VARTYPE_BYTE	Unsigned 8-bit value
DM_VARTYPE_SWORD	Signed 16-bit value
DM_VARTYPE_WORD	Unsigned 16-bit value
DM_VARTYPE_SDWORD	Signed 32-bit value
DM_VARTYPE_DWORD	Unsigned 32-bit value
DM_VARTYPE_FLOAT	Floating-point number 32-bit IEEE 754
DM_VARTYPE_DOUBLE	Floating-point number 64-bit IEEE 754
DM_VARTYPE_TEXT_8	Text tag, 8-bit character set
DM_VARTYPE_TEXT_16	Text tag, 16-bit character set
DM_VARTYPE_RAW	Raw data type
DM_VARTYPE_STRUCT	Structure tag
DM_VARTYPE_TEXTREF	Text reference from text lib
Values >=1024	User-defined structure types This type ID is needed in some calls, for example, for filters to identify a specific structure type.

dwSize

dwSize specifies the length of the data type (in bytes) on the OS.

szTypeName

The name of the structure type for structure tags is in szTypeName.

Comments

DM_TYPEREF is used in the DM_VAR_UPDATE_STRUCT (Page 1579) and DM_VARIABLE_DATA (Page 1588) structures.

Required files

dmclient.h

API functions

DMGetVarType (Page 1726)	Get data type of a tag

See also

DM_VAR_UPDATE_STRUCT (Page 1579)

DMGetVarType (Page 1726)

DM_VARIABLE_DATA (Page 1588)

DM_VAR_UPDATE_STRUCTEX (Page 1581)

DM_VARIABLE_DATA4 (Page 1590)

3.2.2.14 DM_VAR_UPDATE_STRUCT (RT Professional)**Declaration**

```
typedef struct {
    DM_TYPEREF    dmTypeRef;
    DM_VARKEY     dmVarKey;
    VARIANT       dmValue;
    DWORD         dwState;
}
DM_VAR_UPDATE_STRUCT;
```

Members**dmTypeRef**

The DM_TYPEREF (Page 1576) structure contains information on the tag type.

Exception: For performance reasons, nothing is entered into this structure in case of cyclic requests.

dmVarKey

The tag to be processed is specified through the DM_VARKEY (Page 1592) structure.

dmValue

Tag value.

Note

The dmValue VARIANT must be initialized with VariantInit(&dmValue) before its first use and then released with VariantClear(&dmValue) after its use.

Only an array of several DM_VAR_UPDATE_STRUCT can be pre-initialized before its first use with ZeroMemory() or memset(). This affects the VARIANT dmValue contained here like a VariantInit(&dmValue), since 0 corresponds to a VT_EMPTY.

The DM_VAR_UPDATE_STRUCT structure may not be cleared with ZeroMemory() or memset() once it has been used. VARIANT then contains data that may be of the VT_BSTR type.

If the tag type is VT_BSTR, problems can occur in memory management (memory leak), if no VariantClear(&dmValue) is made before allocated DM_VAR_UPDATE_STRUCT structures are deleted with delete[].

	(0x0000)	No error
DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	Connection to partner not established
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	Protocol error
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	Network module faulty
DM_VARSTATE_MAX_LIMIT	(0x0008)	Configured high limit violated
DM_VARSTATE_MIN_LIMIT	(0x0010)	Configured low limit violated
DM_VARSTATE_MAX_RANGE	(0x0020)	High format limit violated
DM_VARSTATE_MIN_RANGE	(0x0040)	Low format limit violated
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	Display conversion error (in conjunction with DM_VARSTATE_..._RANGE)
DM_VARSTATE_STARTUP_VALUE	(0x0100)	Initialization value of tag
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	Substitute value of tag
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	Addressing error in the channel
DM_VARSTATE_INVALID_KEY	(0x0800)	Tag not found / not available
DM_VARSTATE_ACCESS_FAULT	(0x1000)	Access to tag not permitted
DM_VARSTATE_TIMEOUT	(0x2000)	Timeout / no feedback from channel
DM_VARSTATE_SERVERDOWN	(0x4000)	Server is down

Required files

dmclient.h

API functions

DM_NOTIFY_VARIABLE_PROC (Page 1806)	Get tag values (callback)
DMGetValue (Page 1689)	Get tag values

See also

DM_NOTIFY_VARIABLE_PROC (Page 1806)

DM_VARKEY (Page 1592)

DMGetValue (Page 1689)

DM_TYPEREF (Page 1576)

3.2.2.15 DM_VAR_UPDATE_STRUCTEX (RT Professional)**Declaration**

```
typedef struct {  
    DM_TYPEREF    dmTypeRef;  
    DM_VARKEY     dmVarKey;  
    VARIANT       dmValue;  
    DWORD         dwState;  
    DWORD         dwQualityCode;  
}  
DM_VAR_UPDATE_STRUCTEX;
```

Members**dmTypeRef**

The DM_TYPEREF (Page 1576) structure contains information on the tag type.

Exception: For performance reasons, nothing is entered into this structure in case of cyclic requests.

dmVarKey

The tag to be processed is specified through the DM_VARKEY (Page 1592) structure.

dmValue

Tag value.

Note

The VARIANT dmValue must be initialized with VariantInit(&dmValue) before its first use and then released with VariantClear(&dmValue) after its use.

Only an array of several DM_VAR_UPDATE_STRUCT can be pre-initialized before its first use with ZeroMemory() or memset(). This affects the VARIANT dmValue contained here like a VariantInit(&dmValue), since 0 corresponds to a VT_EMPTY.

The DM_VAR_UPDATE_STRUCTEX structure may not be cleared with ZeroMemory() or memset() once it has been used. VARIANT then contains data that may be of the VT_BSTR type.

If the tag type is VT_BSTR, problems can occur in memory management (memory leak), if no VariantClear(&dmValue) is made before allocated DM_VAR_UPDATE_STRUCTEX structures are deleted with delete[].

dwState

Indicates whether the value of the tag was successfully changed or whether errors have occurred:

DM_VARSTATE_NOT_ESTABLISHED	0x0001	Connection to partner not established
DM_VARSTATE_HANDSHAKE_ERROR	0x0002	Protocol error
DM_VARSTATE_HARDWARE_ERROR	0x0004	Network module faulty
DM_VARSTATE_MAX_LIMIT	0x0008	Configured high limit violated
DM_VARSTATE_MIN_LIMIT	0x0010	Configured low limit violated
DM_VARSTATE_MAX_RANGE	0x0020	High format limit violated
DM_VARSTATE_MIN_RANGE	0x0040	Low format limit violated
DM_VARSTATE_CONVERSION_ERROR	0x0080	Display conversion error (in conjunction with DM_VARSTATE_..._RANGE)
DM_VARSTATE_STARTUP_VALUE	0x0100	Initialization value of tag
DM_VARSTATE_DEFAULT_VALUE	0x0200	Substitute value of tag
DM_VARSTATE_ADDRESS_ERROR	0x0400	Addressing error in the channel
DM_VARSTATE_INVALID_KEY	0x0800	Tag not found / not available
DM_VARSTATE_ACCESS_FAULT	0x1000	Access to tag not permitted
DM_VARSTATE_TIMEOUT	0x2000	Timeout / no feedback from channel
DM_VARSTATE_SERVERDOWN	0x4000	Server is down

dwQualityCode

Quality code of the tag value.

The quality code is formed as follows:

	WinCC DM tag status	Quality code according to Profibus PA/ OPC
DM_VARSTATE_SERVERDOWN (0x4000)	Server is down	Bad, out of service, 0x1C

DM_VARSTATE_HARDWARE_ERROR (0x0004)	Network module faulty	Bad, device failure, 0x0C
DM_VARSTATE_NOT_ESTABLISHED (0x0001)	Connection to partner not established	Bad, not connected, 0x08
DM_VARSTATE_TIMEOUT (0x2000)	Timeout / no feedback from channel	Uncertain, last usable value, 0x44
DM_VARSTATE_HANDSHAKE_ERROR (0x0002)	Protocol error	Bad, no communication (no usable value), 0x18
DM_VARSTATE_ADDRESS_ERROR (0x0400)	Addressing error in the channel	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_INVALID_KEY (0x0800)	Tag not found / not available	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_ACCESS_FAULT (0x1000)	Access to tag not permitted	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_MAX_RANGE (0x0020)	High format limit violated	Uncertain, engineering unit range violation, high limit gesetzt, 0x56
DM_VARSTATE_MIN_RANGE (0x0040)	Low format limit violated	Uncertain, engineering unit range violation, low limit gesetzt, 0x55
DM_VARSTATE_CONVERSION_ERROR (0x0080)	Display conversion error	Uncertain, engineering unit range violation, ohne limits gesetzt, 0x54
DM_VARSTATE_MAX_LIMIT (0x0008)	Configured high limit violated	Indicate with Limit-Bit HIGH, 0x01, OR-ed together with the other tag status, e.g. good (cascade), ok, results in 0xC2
DM_VARSTATE_MIN_LIMIT (0x0010)	Configured low limit violated	Indicate with Limit-Bit LOW, 0x02, OR-ed together with the other tag status, e.g. good (cascade), ok, results in 0xC1
DM_VARSTATE_STARTUP_VALUE (0x0100)	Initialization value of tag	Uncertain, initial value, 0x4C
DM_VARSTATE_DEFAULT_VALUE (0x0200)	Substitute value of tag	Uncertain, substitute-set, 0x48

Required files

dmclient.h

API functions

DM_NOTIFY_VARIABLEEX_PROC (Page 1817)	Get tag values (callback)
DMGetValueEx (Page 1691)	Get tag values

See also

DM_NOTIFY_VARIABLEEX_PROC (Page 1817)

DM_TYPEREF (Page 1576)

DM_VARKEY (Page 1592)

DMGetValueEx (Page 1691)

3.2.2.16 DM_VARFILTER (RT Professional)

Declaration

```
typedef struct {
    DWORD      dwFlags;
    DWORD      dwNumTypes;
    LPDWORD    pdwTypes;
    LPSTR      lpszGroup;
    LPSTR      lpszName;
    LPSTR      lpszConn;
}
DM_VARFILTER;
```

Members

dwFlags

The dwFlags parameter can be used to set a selection criterion for the tags:

DM_VARFILTER_TYPE	0x00000001	The tag type (pdwTypes) is used as the selection criterion.
DM_VARFILTER_GROUP	0x00000002	The group name (lpszGroup) is used as the selection criterion.
DM_VARFILTER_NAME	0x00000004	The tag name (lpszName) is used as the selection criterion.
DM_VARFILTER_CONNECTION	0x00000008	The name of the logical connection (lpszConn) is used as the selection criterion.
DM_VARFILTER_FAST_CALLBACK	0x00010000	(Flag for DMEnumVariables): If this flag is set, the callback data is no longer buffered. The callback therefore returns the data directly from the database. The advantage of the flag is a faster response before the first callback is sent. At the same time, it avoids heavy memory load for a large amounts of data. Caution You may not call any other DMClient function in this callback. A blockade may otherwise occur.
DM_VARFILTER_LOCAL_ONLY	0x00020000	(Flag for DMEnumVariables): If this flag is set, only local tags are enumerated. Tags used in the project from other servers are not required.

dwNumTypes

Number of tag types specified in pdwTypes.

pdwTypes

pdwTypes identifies the tag types that are to be used as a selection criterion.

DM_VARTYPE_BIT	Binary tag
DM_VARTYPE_SBYTE	Signed 8-bit value

DM_VARTYPE_BYTE	Unsigned 8-bit value
DM_VARTYPE_SWORD	Signed 16-bit value
DM_VARTYPE_WORD	Unsigned 16-bit value
DM_VARTYPE_SDWORD	Signed 32-bit value
DM_VARTYPE_DWORD	Unsigned 32-bit value
DM_VARTYPE_FLOAT	Floating-point number 32-bit IEEE 754
DM_VARTYPE_DOUBLE	Floating-point number 64-bit IEEE 754
DM_VARTYPE_TEXT_8	Text tag, 8-bit character set
DM_VARTYPE_TEXT_16	Text tag, 16-bit character set

The DM_VARTYPE_RAW, DM_VARTYPE_STRUCT and DM_VARTYPE_TEXTREF types are not supported.

Specify the appropriate TypID in order to select structured tags with self-defined structures. Use the GAPIEnumTypes function to determine the TypID.

IpszGroup

Pointer to the name of the tag group. This name should be used as a selection criterion. Wildcards are prohibited.

IpszName

Pointer to the name of a tag. This name should be used as a selection criterion. Wildcards are prohibited.

IpszConn

Pointer to the name of a logical connection. This name should be used as a selection criterion. Wildcards are prohibited.

Required files

dmclient.h

API functions

DMLenumVariables (Page 1686)	List tag names
DMSHOWVarDatabase (Page 1756)	Open tag selection dialog
DMSHOWVarDatabaseMulti (Page 1761)	Open tag selection dialog

See also

DMLenumVariables (Page 1686)

DMSHOWVarDatabase (Page 1756)

DMSHOWVarDatabaseMulti (Page 1761)

3.2.2.17 DM_VARGRP_DATA (RT Professional)

Declaration

```
typedef struct {
    CHAR      szName[ MAX_DM_VAR_NAME + 1 ];
    DWORD     dwCreatorID;
    WORD      dwVarNum;
    LPVOID    lpvUserData;
}
DM_VARGRP_DATA;
```

Members

- szName**
Name of the tag group
- dwCreatorID**
In the creator identification, you can determine who created an object.
The values 0 – 10100 and 11000 – 11100 are reserved for internal or specific systems.
- dwVarNum**
Number of tags within the tag group
- lpvUserData**
Pointer to application-specific data.

Required files

dmclient.h

API functions

DM_ENUM_VARGRP_PROC (Page 1685)	List information about a tag group (callback)

See also

DM_ENUM_VARGRP_PROC (Page 1685)

3.2.2.18 DM_VARGRPKEY (RT Professional)

Declaration

```
typedef struct {  
    CHAR      szName[ MAX_DM_VAR_NAME + 1 ];  
    LPVOID     lpvUserData;  
}  
DM_VARGRPKEY;
```

Members

szName

Name of the tag group

lpvUserData

Pointer to application-specific data.

Required files

dmclient.h

API functions

DMEnumVarGrpData (Page 1681)	List information about a tag group

See also

DMEnumVarGrpData (Page 1681)

3.2.2.19 DM_VARIABLE_DATA (RT Professional)

Declaration

```
typedef struct {
    DM_TYPEREF          dmTypeRef;
    DM_VARLIMIT          dmVarLimit;
    VARIANT              dmStart;
    VARIANT              dmDefault;
    DWORD               dwNotify;
    DWORD               dwFlags;
    CHAR                 szSpecific[MAX_DM_VAR_SPECIFIC +1];
    CHAR                 szGroup[MAX_DM_GROUP_NAME +1];
    CHAR                 szConnection[
        MAX_DM_CONNECTION_NAME +1];
    CHAR                 szChannel[_MAX_PATH +1];
    CHAR                 szUnit[MAX_DM_UNIT_NAME +1];
}
DM_VARIABLE_DATA;
```

Members

dmTypeRef

The DM_TYPEREF (Page 1576) structure contains information on the tag type.

dmVarLimit

The DM_VARLIMIT (Page 1595) structure contains information on the limits of a tag.

dmStart

Start value of the tag. Must be release with VariantClear(&dmStart) after use.

dmDefault

Substitute value of tag. Must be release with VariantClear(&dmDefault) after use.

dwNotify

dwNotify specifies the events that generate a report entry:

DM_NOTIFY_MAX_LIMIT	0x00000001	On reaching the high limit
DM_NOTIFY_MIN_LIMIT	0x00000002	On reaching the low limit
DM_NOTIFY_FORMAT_ERROR	0x00000004	On conversion error
DM_NOTIFY_ACCESS_FAULT	0x00000008	On prohibited write access
DM_NOTIFY_APPLICATION_WRITE	0x00000010	On write access by application
DM_NOTIFY_PROCESS_WRITE	0x00000020	On write access by process

dwFlags

indicates how the substitute value should be used:

DM_HAS_MIN_LIMIT	0x00000008	Tag has a fixed low limit
DM_HAS_MAX_LIMIT	0x00000010	Tag has a fixed high limit
DM_HAS_DEFAULT_VALUE	0x00000020	Tag has a substitute value
DM_HAS_STARTUP_VALUE	0x00000040	Tag has an initial value
DM_USE_DEFAULT_ON_STARTUP	0x00000080	Enter substitute value at system start
DM_USE_DEFAULT_ON_MAX	0x00000100	Enter substitute value when high limit is violated
DM_USE_DEFAULT_ON_MIN	0x00000200	Enter substitute value when low limit is violated
DM_USE_DEFAULT_ON_COMM_ERROR	0x00000400	Substitute value on connection error

szSpecific

szSpecific contains the address relationship of the tag, for example, data block and byte in the block etc. Refer to the communication manual for PLC-specific details.

This is the same value that appears in the properties of a tag in WinCC in the Parameter column.

szGroup

Name of the group to which the tag belongs.

This value is not provided for server tags on the multi-client.

szConnection

Name of the logical connection to which the tag is linked.

szChannel

File name of the channel driver.

szUnit

Name of the channel unit to which the tag is linked.

Required files

dmclient.h

API functions

DM_ENUM_VARIABLE_PROC (Page 1676)	List information about a tag (callback)

See also

DM_TYPEREF (Page 1576)

DM_ENUM_VARIABLE_PROC (Page 1676)

DM_VARLIMIT (Page 1595)

DMEnumVarData (Page 1674)

3.2.2.20 DM_VARIABLE_DATA4 (RT Professional)**Declaration**

```

typedef struct {
    DM_TYPEREF          dmTypeRef;
    DM_VARLIMIT          dmVarLimit;
    VARIANT              dmStart;
    VARIANT              dmDefault;
    DWORD                dwNotify;
    DWORD                dwFlags;
    CHAR                  szSpecific[MAX_DM_VAR_SPECIFIC +1];
    CHAR                  szGroup[MAX_DM_GROUP_NAME +1];
    CHAR                  szConnection[
        MAX_DM_CONNECTION_NAME +1];
    CHAR                  szChannel[_MAX_PATH +1];
    CHAR                  szUnit[MAX_DM_UNIT_NAME +1];
    MCP_VARIABLE_SCALES   Scaling;
    DWORD                dwASDataSize;
    DWORD                dwOSDataSize;
    DWORD                dwVarProperty;
    DWORD                dwFormat;
}
DM_VARIABLE_DATA4;

```

Members**dmTypeRef**

The DM_TYPEREF (Page 1576) structure contains information on the tag type.

dmVarLimit

The DM_VARLIMIT (Page 1595) structure contains information on the limits of a tag.

dmStart

Start value of the tag. Must be release with VariantClear(&dmStart) after use.

dmDefault

Substitute value of tag. Must be release with VariantClear(&dmDefault) after use.

dwNotify

dwNotify specifies the events that generate a report entry:

DM_NOTIFY_MAX_LIMIT	0x00000001	On reaching the high limit
DM_NOTIFY_MIN_LIMIT	0x00000002	On reaching the low limit

DM_NOTIFY_FORMAT_ERROR	0x00000004	On conversion error
DM_NOTIFY_ACCESS_FAULT	0x00000008	On prohibited write access
DM_NOTIFY_APPLICATION_WRITE	0x00000010	On write access by application
DM_NOTIFY_PROCESS_WRITE	0x00000020	On write access by process

dwFlags

Indicates how the substitute value should be used:

DM_HAS_MIN_LIMIT	0x00000008	Tag has a fixed low limit
DM_HAS_MAX_LIMIT	0x00000010	Tag has a fixed high limit
DM_HAS_DEFAULT_VALUE	0x00000020	Tag has a substitute value
DM_HAS_STARTUP_VALUE	0x00000040	Tag has an initial value
DM_USE_DEFAULT_ON_STARTUP	0x00000080	Enter substitute value at system start
DM_USE_DEFAULT_ON_MAX	0x00000100	Enter substitute value when high limit is violated
DM_USE_DEFAULT_ON_MIN	0x00000200	Enter substitute value when low limit is violated
DM_USE_DEFAULT_ON_COMM_ERROR	0x00000400	Substitute value on connection error

szSpecific

szSpecific contains the address relationship of the tag, for example, data block and byte in the block etc. Refer to the communication manual for PLC-specific details.

This is the same value that is displayed in the Parameter column of the properties of a tag in WinCC.

szGroup

Name of the group to which the tag belongs.

szConnection

Name of the logical connection to which the tag is linked.

szChannel

File name of the channel driver.

szUnit

Name of the channel unit to which the tag is linked.

Scaling

MCP_VARIABLE_SCALES (Page 1618) Structure with the description of tag scaling.

dwASDataSize

Length of the tags in the PLC (number of bits)

dwOSDataSize

Length of the tags in the OS (number of bits)

dwVarProperty

Indicates whether this is an internal or external tag:

DM_INTERNAL_VAR	0x00000002	Internal tag
DM_EXTERNAL_VAR	0x00000004	External tag

dwFormat

Number of the conversion routine used. You can find more detailed information in the "Conversion routines" section.

Required files

dmclient.h

API functions

DM_ENUM_VARIABLE_PROC4 (Page 1680)	List information about a tag (callback)

See also

DMEEnumVarData4 (Page 1677)
 DM_ENUM_VARIABLE_PROC4 (Page 1680)
 DM_TYPEREF (Page 1576)
 DM_VARLIMIT (Page 1595)
 MCP_VARIABLE_SCALES (Page 1618)

3.2.2.21 DM_VARKEY (RT Professional)**Declaration**

```
typedef struct {
    DWORD    dwKeyType;
    DWORD    dwID;
    char     szName[ MAX_DM_VAR_NAME + 1 ];
    LPVOID   lpvUserData;
}
DM_VARKEY;
```

Members

dwKeyType

dwKeyType defines whether the tag is to be addressed by a key ID or by its name.

DM_VARKEY_ID	Specification via key ID
DM_VARKEY_NAME	Specification via tag name

When DM_VARKEY is used as the return structure, both types can be set if the key ID and tag name are returned.

When DM_VARKEY is used as a source parameter, the tag name should be preferably set, since the server prefix can only be specified there. If the key ID is used, operation is always local.

dwID

Contains the key ID of the tag, if dwKeyType is set accordingly.

szName

Contains the name of the tag, if dwKeyType is set accordingly.

lpvUserData

Pointer to application-specific data.

Comments

The DM_VARKEY structure is part of the DM_VAR_UPDATE_STRUCT (Page 1577) structure.

Required files

dmclient.h

API functions

DM_ENUM_TYPEMEMBERS_PROC_EX (Page 1781)	List tags in a structured tag (callback)
DM_ENUM_TYPEMEMBERS_PROC (Page 1777)	List tags in a structured tag (callback)
DM_ENUM_VAR_PROC (Page 1688)	List tag names (callback)
DM_ENUM_VARIABLE_PROC (Page 1676)	List information about a tag (callback)
DM_NOTIFY_SELECT_VAR_PROC (Page 1767)	Open tag selection dialog (callback)
DMEnumVarData (Page 1674)	List information about a tag
DMGetValue (Page 1689)	Get tag values
DMGetValueWait (Page 1702)	Get tag values with update
DMGetVarLimits (Page 1719)	Get limits of a tag
DMGetVarInfo (Page 1708)	Get tag ID, get tag name
DMGetVarType (Page 1726)	Get data type of a tag
DMSetValue (Page 1732)	Change tag values

DMSetValueMessage (Page 1738)	Change tag values with alarm output
DMSetValueWait (Page 1742)	Change tag values with notification
DMSetValueWaitMessage (Page 1747)	Change tag values with notification and alarm output
DMShowVarDatabase (Page 1756)	Open tag selection dialog
DMStartVarUpdate (Page 1804)	Define update tags

See also

DMStartVarUpdate (Page 1804)
 DM_VAR_UPDATE_STRUCT (Page 1577)
 DMGetValue (Page 1689)
 DMGetValueWait (Page 1702)
 DMGetVarInfo (Page 1708)
 DMGetVarLimits (Page 1719)
 DMGetVarType (Page 1726)
 DMSetValue (Page 1732)
 DMSetValueMessage (Page 1738)
 DMSetValueWait (Page 1742)
 DMSetValueWaitMessage (Page 1747)
 DMShowVarDatabase (Page 1756)
 DMEnumVarData (Page 1674)
 DM_ENUM_VAR_PROC (Page 1688)
 DM_ENUM_VARIABLE_PROC (Page 1676)
 DM_ENUM_TPEMEMBERS_PROC (Page 1777)
 DM_ENUM_TPEMEMBERS_PROC_EX (Page 1781)
 DM_NOTIFY_SELECT_VAR_PROC (Page 1767)
 DMStartVarUpdateEx (Page 1807)
 DM_VAR_UPDATE_STRUCTEX (Page 1579)
 DM_ENUM_VARIABLE_PROC4 (Page 1680)
 DMGetValueEx (Page 1691)
 DMGetValueWaitEx (Page 1704)
 DM_ENUM_TPEMEMBERS_PROC_EX4 (Page 1784)
 MSG_CSDATA_STRUCT_PLUS (Page 2378)

3.2.2.22 DM_VARLIMIT (RT Professional)

Declaration

```
typedef struct {  
    VARIANT    dmMaxRange;  
    VARIANT    dmMinRange;  
    VARIANT    dmMaxLimit;  
    VARIANT    dmMinLimit;  
}  
DM_VARLIMIT;
```

Members

dmMaxRange

High limit of the format conversion

dmMinRange

Low limit of the format conversion

dmMaxLimit

High limit of the tag.

dmMinLimit

Low limit of the tag.

Comments

DM_VARLIMIT is part of the DM_VARIABLE_DATA (Page 1586) structure. All VARIANT must be release with VariantClear(&dmxxx) after use.

Required files

dmclient.h

API functions

DMGetVarLimits (Page 1719)	Get limits of a tag

See also

DM_VARIABLE_DATA (Page 1586)

DMGetVarLimits (Page 1719)

DM_VARIABLE_DATA4 (Page 1588)

3.2.2.23 MCP_NEWVARIABLE_DATA (RT Professional)

Declaration

```
typedef struct {
    DWORD                dwFlags;
    char                 szProjectFile[_MAX_PATH +1];
    char                 szConnection[
        MAX_DM_CONNECTION_NAME +3];
    char                 szVarName[MAX_DM_VAR_NAME +1];
    char                 szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON Common;
    MCP_VARIABLE_PROTOCOL Protocol;
    MCP_VARIABLE_LIMITS Limits;
    char                 szSpecific[MAX_DM_VAR_SPECIFIC +1];
}
MCP_NEWVARIABLE_DATA;
```

Members

dwFlags

dwFlags indicates how the tag should be processed:

MCP_NVAR_FLAG_CREATE	1	Create tag
MCP_NVAR_FLAG_MODIFY	2	Change tag (not to be used in RT)
MCP_NVAR_FLAG_TEST	3	Check for existence of a tag

The flags must not be ORed.

szProjectFile

Name of the project file, including path and extension.

The name of the project file can be determined with DMEnumOpenedProjects or in RT with DMGetRuntimeProject.

If an empty string is entered, an internal DMEnumOpenedProjects is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other entry is rejected with an error (DM_E_NOT_CONNECTED).

szConnection

Name of the logical connection assigned to the tag.

szVarName

Name of the tag to be processed.

szGroupName

Name of the group to which the tag belongs.

If a specified group name is in a connection other than that specified in `szConnection`, the group name is ignored without an error message and the tag is created directly in the connection.

If a group name is specified that does not yet exist, the group is implicitly created.

When a tag is changed with `MCP_NVAR_FLAG_MODIFY`, a change in the group name is ignored without an error message.

Common

`MCP_VARIABLE_COMMON` (Page 1606) Structure with the description of the tag.

Protocol

The `MCP_VARIABLE_PROTOCOL` (Page 1615) structure with the description of how limit violations are handled by the tag.

Limits

`MCP_VARIABLE_LIMITS` (Page 1610) Structure with the limits of the tag.

szSpecific

`szSpecific` contains the address relationship of the tag, for example, data block and byte in the block etc. Refer to the communication manual for PLC-specific details.

This is the same value that is displayed in the Parameter column of the properties of a tag in WinCC.

Comments

The `MCP_NEWVARIABLE_DATA_EX` (Page 1602) structure has a similar use for extended functionality.

Required files

`dmclient.h`

API functions

<code>GAPICreateNewVariable</code> (Page 1769)	Create tag

See also

`GAPICreateNewVariable` (Page 1769)

`MCP_VARIABLE_PROTOCOL` (Page 1615)

`MCP_VARIABLE_COMMON` (Page 1606)

`MCP_VARIABLE_LIMITS` (Page 1610)

`MCP_NEWVARIABLE_DATA_EX` (Page 1602)

3.2.2.24 MCP_NEWVARIABLE_DATA_4 (RT Professional)

Declaration

```
typedef struct {
    DWORD                dwFlags;
    char                 szProjectFile[_MAX_PATH +1];
    char                 szConnection[
        MAX_DM_CONNECTION_NAME +3];
    char                 szVarName[MAX_DM_VAR_NAME +1];
    char                 szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON Common;
    MCP_VARIABLE_PROTOCOL Protocol;
    MCP_VARIABLE_LIMITS Limits;
    char                 szSpecific[MAX_DM_VAR_SPECIFIC +1];
    MCP_VARIABLE_SCALES Scaling;
}
MCP_NEWVARIABLE_DATA_4;
```

Members

dwFlags

dwFlags indicates how the tag should be processed:

MCP_NVAR_FLAG_CREATE	1	Create tag
MCP_NVAR_FLAG_MODIFY	2	Change tag (if not available, then re-create) (not to be used in RT)
MCP_NVAR_FLAG_TEST	3	Check for existence of a tag

The flags must not be ORed.

szProjectFile

Name of the project file, including path and extension.

The name of the project file can be determined with DMEnumOpenedProjects or in runtime with DMGetRuntimeProject.

If an empty string is entered, an internal DMEnumOpenedProjects is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected with the error, DM_E_NOT_CONNECTED.

szConnection

Name of the logical connection assigned to the tag.

szVarName

Name of the tag to be processed.

szGroupName

Name of the group to which the tag belongs.

If a specified group name is in a connection other than that specified in szConnection, the group name is ignored without an error message and the tag is created directly in the connection.

If a group name is specified that does not yet exist, the group is implicitly created.

When a tag is changed with MCP_NVAR_FLAG_MODIFY, a change in the group name is ignored without an error message.

Common

MCP_VARIABLE_COMMON (Page 1606) Structure with the description of the tag.

Protocol

The MCP_VARIABLE_PROTOCOL (Page 1615) structure with the description of how limit violations are handled by the tag.

Limits

MCP_VARIABLE_LIMITS (Page 1610) Structure with the limits of the tag.

szSpecific

szSpecific contains the address relationship of the tag, for example, data block and byte in the block etc. Refer to the communication manual for PLC-specific details.

This is the same value that is displayed in the Parameter column of the properties of a tag in WinCC.

Scaling

MCP_VARIABLE_SCALES (Page 1618) Structure with the description of tag scaling.

Comments

The MCP_NEWVARIABLE_DATA_EX4 (Page 1604) structure has a similar use for extended functionality.

Required files

dmclient.h

API functions

GAPICreateNewVariable4 (Page 1770)	Create tag
GAPICreateNewVariableEx4 (Page 1773)	Create tag with Creator ID

See also

MCP_VARIABLE_COMMON (Page 1606)
 MCP_VARIABLE_LIMITS (Page 1610)
 MCP_VARIABLE_PROTOCOL (Page 1615)
 GAPICreateNewVariable4 (Page 1770)
 MCP_VARIABLE_SCALES (Page 1618)
 MCP_NEWVARIABLE_DATA_EX4 (Page 1604)
 GAPICreateNewVariableEx4 (Page 1773)

3.2.2.25 MCP_NEWVARIABLE_DATA_5 (RT Professional)**Declaration**

```

typedef struct {
    DWORD                dwFlags;
    char                 szProjectFile[_MAX_PATH +1];
    char                 szConnection[
        MAX_DM_CONNECTION_NAME +3];
    char                 szVarName[MAX_DM_VAR_NAME +1];
    char                 szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON Common;
    MCP_VARIABLE_PROTOCOL Protocol;
    MCP_VARIABLE_LIMITS5 Limits;
    char                 szSpecific[MAX_DM_VAR_SPECIFIC +1];
    MCP_VARIABLE_SCALES Scaling;
}
MCP_NEWVARIABLE_DATA_5;
  
```

Members**dwFlags**

dwFlags indicates how the tag should be processed:

MCP_NVAR_FLAG_CREATE	1	Create tag
MCP_NVAR_FLAG_MODIFY	2	Change tag (if not available, then re-create) (not to be used in RT)
MCP_NVAR_FLAG_TEST	3	Check for existence of a tag

The flags must not be ORed.

szProjectFile

Name of the project file, including path and extension.

The name of the project file can be determined with DMEnumOpenedProjects or in runtime with DMGetRuntimeProject.

If an empty string is entered, an internal DMEnumOpenedProjects is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected with the error, DM_E_NOT_CONNECTED.

szConnection

Name of the logical connection assigned to the tag.

szVarName

Name of the tag to be processed.

szGroupName

Name of the group to which the tag belongs.

If a specified group name is in a connection other than that specified in szConnection, the group name is ignored without an error message and the tag is created directly in the connection.

If a group name is specified that does not yet exist, the group is implicitly created.

When a tag is changed with MCP_NVAR_FLAG_MODIFY, a change in the group name is ignored without an error message.

Common

MCP_VARIABLE_COMMON (Page 1606) Structure with the description of the tag.

Protocol

The MCP_VARIABLE_PROTOCOL (Page 1615) structure with the description of how limit violations are handled by the tag.

Limits

MCP_VARIABLE_LIMITS5 (Page 1612) Structure with the limits of the tag.

szSpecific

szSpecific contains the address relationship of the tag, for example, data block and byte in the block etc. Refer to the communication manual for PLC-specific details.

This is the same value that is displayed in the Parameter column of the properties of a tag in WinCC.

Scaling

MCP_VARIABLE_SCALES (Page 1618) Structure with the description of tag scaling.

Comments

The MCP_NEWVARIABLE_DATA_EX4 (Page 1604) structure has a similar use for extended functionality.

Required files

dmclient.h

API functions

GAPICreateNewVariable5 (Page 1772)	Create tag with Creator ID

See also

GAPICreateNewVariable5 (Page 1772)

MCP_VARIABLE_COMMON (Page 1606)

MCP_VARIABLE_LIMITS5 (Page 1612)

MCP_VARIABLE_PROTOCOL (Page 1615)

MCP_NEWVARIABLE_DATA_EX4 (Page 1604)

MCP_VARIABLE_SCALES (Page 1618)

3.2.2.26 MCP_NEWVARIABLE_DATA_EX (RT Professional)

Declaration

```
typedef struct {
    DWORD                dwFlags;
    char                 szProjectFile[_MAX_PATH +1];
    char                 szConnection[
        MAX_DM_CONNECTION_NAME +3];
    char                 szVarName[MAX_DM_VAR_NAME +1];
    char                 szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON_EX Common;
    MCP_VARIABLE_PROTOCOL_EX Protocol;
    MCP_VARIABLE_LIMITS_EX Limits;
    char                 szSpecific[MAX_DM_VAR_SPECIFIC +1]
}
MCP_NEWVARIABLE_DATA_EX;
```

Members

dwFlags

dwFlags indicates how the tag should be processed:

MCP_NVAR_FLAG_CREATE	1	Create tag
MCP_NVAR_FLAG_MODIFY	2	Change tag (not to be used in runtime)
MCP_NVAR_FLAG_TEST	3	Checks for existence of a tag.

The flags must not be ORed.

szProjectFile

Name of the project file, including path and extension.

The name of the project file can be determined with `DMEnumOpenedProjects` or in runtime with `DMGetRuntimeProject`.

If an empty string is entered, an internal `DMEnumOpenedProjects` is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected with the error, `DM_E_NOT_CONNECTED`.

szConnection

Name of the logical connection assigned to the tag.

szVarName

Name of the tag to be processed.

szGroupName

Name of the group to which the tag belongs.

If a specified group name is in a connection other than that specified in `szConnection`, the group name is ignored without an error message and the tag is created directly in the connection.

If a group name is specified that does not yet exist, the group is implicitly created.

When a tag is changed with `MCP_NVAR_FLAG_MODIFY`, a change in the group name is ignored without an error message.

Common

`MCP_VARIABLE_COMMON_EX` (Page 1608) Structure with the description of the tag.

Protocol

The `MCP_VARIABLE_PROTOCOL_EX` (Page 1617) structure with the description of how limit violations are handled by the tag.

Limits

`MCP_VARIABLE_LIMITS_EX` (Page 1614) Structure with the limits of the tag.

szSpecific

`szSpecific` contains the address relationship of the tag, for example, data block and byte in the block etc. Refer to the communication manual for PLC-specific details.

This is the same value that is displayed in the Parameter column of the properties of a tag in WinCC.

Comments

The `MCP_NEWVARIABLE_DATA` (Page 1594) structure has a similar use.

Required files

`dmclient.h`

API functions

DM_ENUM_TYMEMBERS_PROC_EX (Page 1781)	List tags in a structured tag (callback)

See also

MCP_NEWVARIABLE_DATA (Page 1594)

DM_ENUM_TYMEMBERS_PROC_EX (Page 1781)

MCP_VARIABLE_COMMON_EX (Page 1608)

MCP_VARIABLE_LIMITS_EX (Page 1614)

MCP_VARIABLE_PROTOCOL_EX (Page 1617)

3.2.2.27 MCP_NEWVARIABLE_DATA_EX4 (RT Professional)**Declaration**

```
typedef struct {
    DWORD                dwFlags;
    char                 szProjectFile[_MAX_PATH +1];
    char                 szConnection[
        MAX_DM_CONNECTION_NAME +3];
    char                 szVarName[MAX_DM_VAR_NAME +1];
    char                 szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON_EX Common;
    MCP_VARIABLE_PROTOCOL_EX Protocol;
    MCP_VARIABLE_LIMITS_EX Limits;
    char                 szSpecific[MAX_DM_VAR_SPECIFIC +1]
    MCP_VARIABLE_SCALES  Scaling;
}
MCP_NEWVARIABLE_DATA_EX4;
```

Members**dwFlags**

dwFlags indicates how the tag should be processed:

MCP_NVAR_FLAG_CREATE	1	Create tag
MCP_NVAR_FLAG_MODIFY	2	Change tag (not to be used in RT)
MCP_NVAR_FLAG_TEST	3	Checks for existence of a tag.

The flags must not be ORed.

szProjectFile

Name of the project file, including path and extension.

The name of the project file can be determined with `DMEnumOpenedProjects` or in runtime with `DMGetRuntimeProject`.

If an empty string is entered, an internal `DMEnumOpenedProjects` is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected with the error, `DM_E_NOT_CONNECTED`.

szConnection

Name of the logical connection assigned to the tag.

szVarName

Name of the tag to be processed.

szGroupName

Name of the group to which the tag belongs.

If a specified group name is in a connection other than that specified in `szConnection`, the group name is ignored without an error message and the tag is created directly in the connection.

If a group name is specified that does not yet exist, the group is implicitly created.

When a tag is changed with `MCP_NVAR_FLAG_MODIFY`, a change in the group name is ignored without an error message.

Common

`MCP_VARIABLE_COMMON_EX` (Page 1608) Structure with the description of the tag.

Protocol

The `MCP_VARIABLE_PROTOCOL_EX` (Page 1617) structure with the description of how limit violations are handled by the tag.

Limits

`MCP_VARIABLE_LIMITS_EX` (Page 1614) Structure with the limits of the tag.

szSpecific

`szSpecific` contains the address relationship of the tag, for example, data block and byte in the block etc. Refer to the communication manual for PLC-specific details.

This is the same value that is displayed in the Parameter column of the properties of a tag in WinCC.

Scaling

`MCP_VARIABLE_SCALES` (Page 1618) Structure with the description of tag scaling.

Comments

The `MCP_NEWVARIABLE_DATA_4` (Page 1596) structure has a similar use.

Required files

dmclient.h

API functions

DM_ENUM_TYMEMBERS_PROC_EX4 (Page 1784)	List tags in a structured tag (callback)

See also

MCP_NEWVARIABLE_DATA_4 (Page 1596)

MCP_NEWVARIABLE_DATA_5 (Page 1598)

DM_ENUM_TYMEMBERS_PROC_EX4 (Page 1784)

MCP_VARIABLE_COMMON_EX (Page 1608)

MCP_VARIABLE_LIMITS_EX (Page 1614)

MCP_VARIABLE_PROTOCOL_EX (Page 1617)

MCP_VARIABLE_SCALES (Page 1618)

3.2.2.28 MCP_VARIABLE_COMMON (RT Professional)**Declaration**

```
typedef struct {
    DWORD    dwVarType;
    DWORD    dwVarLength;
    DWORD    dwVarProperty;
    DWORD    dwFormat;
}
MCP_VARIABLE_COMMON;
```

Members**dwVarType**

The type of tag is specified in dwVarType:

DM_VARTYPE_BIT	Binary tag
DM_VARTYPE_SBYTE	Signed 8-bit value
DM_VARTYPE_BYTE	Unsigned 8-bit value
DM_VARTYPE_SWORD	Signed 16-bit value
DM_VARTYPE_WORD	Unsigned 16-bit value
DM_VARTYPE_SDWORD	Signed 32-bit value
DM_VARTYPE_DWORD	Unsigned 32-bit value

DM_VARTYPE_FLOAT	Floating-point number 32-bit IEEE 754
DM_VARTYPE_DOUBLE	Floating-point number 64-bit IEEE 754
DM_VARTYPE_TEXT_8	Text tag, 8-bit character set
DM_VARTYPE_TEXT_16	Text tag, 16-bit character set
DM_VARTYPE_RAW	Raw data type
DM_VARTYPE_STRUCT	Structure tag
DM_VARTYPE_TEXTREF	Text reference from text lib

dwVarLength

The specification of the tag length is only relevant for text tags DM_VARTYPE_TEXT_8 and DM_VARTYPE_TEXT_16.

The text length is specified in characters here (1 to 255).

dwVarProperty

Indicates whether this is an internal or external tag:

DM_INTERNAL_VAR	Internal tag
DM_EXTERNAL_VAR	External tag

dwFormat

Number of the conversion routine to be used. You can find more detailed information in the "Conversion routines" section.

Comments

MCP_VARIABLE_COMMON is part of the MCP_NEWVARIABLE_DATA (Page 1594) and MCP_NEWVARIABLE_DATA_4 (Page 1596) structures.

The MCP_VARIABLE_COMMON_EX (Page 1608) structure has a similar use for extended functionality.

Required files

dmclient.h

See also

MCP_NEWVARIABLE_DATA (Page 1594)

MCP_NEWVARIABLE_DATA_4 (Page 1596)

MCP_VARIABLE_COMMON_EX (Page 1608)

MCP_NEWVARIABLE_DATA_5 (Page 1598)

3.2.2.29 MCP_VARIABLE_COMMON_EX (RT Professional)

Declaration

```
typedef struct {
    DWORD    dwVarType;
    DWORD    dwCreatorID
    DWORD    dwVarLength;
    DWORD    dwVarProperty;
    DWORD    dwFormat;
    DWORD    dwOSOffset;
    DWORD    dwASOffset;
    char     szStructTypeName
}
MCP_VARIABLE_COMMON_EX;
```

Members

dwVarType

The type of tag is specified in dwVarType:

DM_VARTYPE_BIT	Binary tag
DM_VARTYPE_SBYTE	Signed 8-bit value
DM_VARTYPE_BYTE	Unsigned 8-bit value
DM_VARTYPE_SWORD	Signed 16-bit value
DM_VARTYPE_WORD	Unsigned 16-bit value
DM_VARTYPE_SDWORD	Signed 32-bit value
DM_VARTYPE_DWORD	Unsigned 32-bit value
DM_VARTYPE_FLOAT	Floating-point number 32-bit IEEE 754
DM_VARTYPE_DOUBLE	Floating-point number 64-bit IEEE 754
DM_VARTYPE_TEXT_8	Text tag, 8-bit character set
DM_VARTYPE_TEXT_16	Text tag, 16-bit character set
DM_VARTYPE_RAW	Raw data type
DM_VARTYPE_STRUCT	Structure tag
DM_VARTYPE_TEXTREF	Text reference from text lib

dwCreatorID

In the creator identification, you can determine who created an object.

The values 0 – 10100 and 11000 – 11100 are reserved for internal or specific systems.

dwVarLength

The specification of the tag length is only relevant for text tags DM_VARTYPE_TEXT_8 and DM_VARTYPE_TEXT_16. The text length is specified in characters here (1 to 255).

dwVarProperty

Indicates whether this is an internal or external tag:

DM_INTERNAL_VAR	Internal tag
DM_EXTERNAL_VAR	External tag

dwFormat

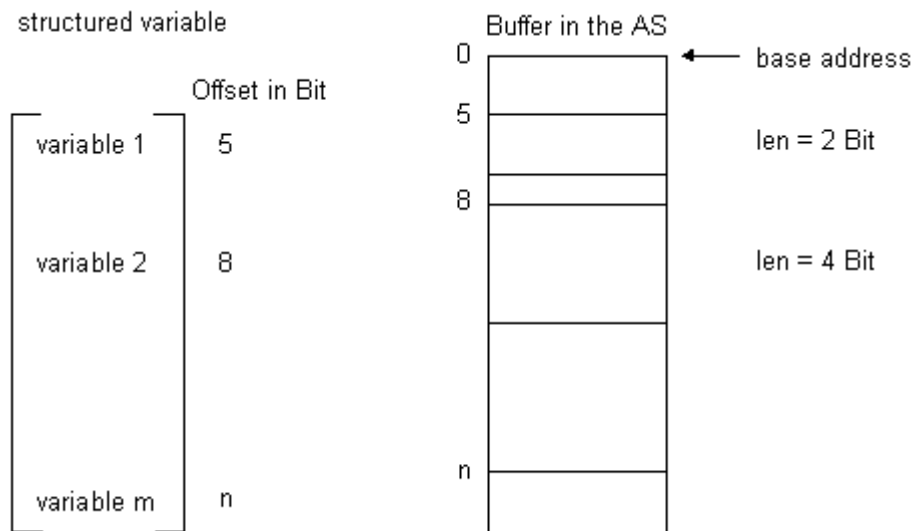
Number of the conversion routine to be used. You can find more detailed information in the "Conversion routines" section.

dwOSOffset

The parameter is reserved for future development and must be preset to 0.

dwASOffset

Offset in the buffer of the PLC



The bit-oriented ASOffset is handled internally in BYTE.

szStructTypeName

Name of the structure type

Comments

MCP_VARIABLE_COMMON_EX is part of the MCP_NEWVARIABLE_DATA_EX (Page 1600) structure.

The MCP_VARIABLE_COMMON (Page 1604) structure has a similar use.

Required files

dmclient.h

See also

MCP_NEWVARIABLE_DATA (Page 1594)

MCP_VARIABLE_COMMON (Page 1604)

MCP_NEWVARIABLE_DATA_EX (Page 1600)

MCP_NEWVARIABLE_DATA_EX4 (Page 1602)

3.2.2.30 MCP_VARIABLE_LIMITS (RT Professional)

Declaration

```
typedef struct {  
    double    dTopLimit;  
    double    dBottomLimit;  
    double    dStartValue;  
    double    dSubstituteValue;  
    BOOL      bTopLimit;  
    BOOL      bBottomLimit;  
    BOOL      bStartValue;  
    BOOL      bConnectionErr;  
    BOOL      bTopLimitValid;  
    BOOL      bBottomLimitValid;  
    BOOL      bStartValueValid;  
    BOOL      bSubstValueValid;  
}  
MCP_VARIABLE_LIMITS;
```

Members

dTopLimit

Value of high limit of the tag

dBottomLimit

Value of low limit of the tag

dStartValue

Start value of the tag

dSubstituteValue

Substitute value of tag

bTopLimit

If this parameter is set, the substitute value should be used if the value of the tag exceeds the value specified in dTopLimit.

bBottomLimit

If this parameter is set, the substitute value should be used if the value of the tag falls below the value specified in dBottomLimit.

bStartValue

If this parameter is set, the substitute value should be used as a start value.

bConnectionErr

If this parameter is set, the substitute value should be used when a connection error occurs.

bTopLimitValid

If this parameter is set, the value specified in dTopLimit applies the high limit.

bBottomLimitValid

If this parameter is set, the value specified in dBottomLimit applies the low limit.

bStartValueValid

If this parameter is set, the value specified in dStartValue applies the start value.

bSubstValueValid

If this parameter is set, the value specified in dSubstituteValue applies the substitute value.

Comments

MCP_VARIABLE_Limits is part of the MCP_NEWVARIABLE_DATA (Page 1594) and MCP_NEWVARIABLE_DATA_4 (Page 1596) structures.

The MCP_VARIABLE_LIMITS_EX (Page 1614) structure has a similar use for extended functionality.

Required files

dmclient.h

See also

MCP_NEWVARIABLE_DATA (Page 1594)

MCP_NEWVARIABLE_DATA_4 (Page 1596)

MCP_VARIABLE_LIMITS_EX (Page 1614)

3.2.2.31 MCP_VARIABLE_LIMITS5 (RT Professional)

Declaration

```
typedef struct {  
    VARIANT    varTopLimit;  
    VARIANT    varBottomLimit;  
    VARIANT    varStartValue;  
    VARIANT    varSubstituteValue;  
    BOOL       bTopLimit;  
    BOOL       bBottomLimit;  
    BOOL       bStartValue;  
    BOOL       bConnectionErr;  
    BOOL       bTopLimitValid;  
    BOOL       bBottomLimitValid;  
    BOOL       bStartValueValid;  
    BOOL       bSubstValueValid;  
}  
MCP_VARIABLE_LIMITS5;
```

Members

varTopLimit

Value of high limit of the tag

varBottomLimit

Value of low limit of the tag

varStartValue

Start value of the tag

varSubstituteValue

Substitute value of tag

bTopLimit

If this parameter is set, the substitute value should be used if the value of the tag exceeds the value specified in dTopLimit.

bBottomLimit

If this parameter is set, the substitute value should be used if the value of the tag falls below the value specified in dBottomLimit.

bStartValue

If this parameter is set, the substitute value should be used as a start value.

bConnectionErr

If this parameter is set, the substitute value should be used when a connection error occurs.

bTopLimitValid

If this parameter is set, the value specified in dTopLimit applies the high limit.

bBottomLimitValid

If this parameter is set, the value specified in dBottomLimit applies the low limit.

bStartValueValid

If this parameter is set, the value specified in dStartValue applies the start value.

bSubstValueValid

If this parameter is set, the value specified in dSubstituteValue applies the substitute value.

Comments

MCP_VARIABLE_LIMITS5 is part of the MCP_NEWVARIABLE_DATA_5 (Page 1598) structure and is needed specify text tags with start and substitute values using the GAPICreateNewVariable5 (Page 1772) function.

Required files

dmclient.h

See also

MCP_NEWVARIABLE_DATA_5 (Page 1598)

GAPICreateNewVariable5 (Page 1772)

3.2.2.32 MCP_VARIABLE_LIMITS_EX (RT Professional)

Declaration

```
typedef struct {
    double    dTopLimit
    double    dBottomLimit;
    double    dStartValue;
    double    dSubstituteValue;
    DWORD     dwLimitFlags
    DWORD     dwTextBibStartText;
    char      szTextStartText[255];
    DWORD     dwTextBibSubstitute;
    char      szTextSubstitute[255]
}
MCP_VARIABLE_LIMITS_EX;
```

Members

dTopLimit

Value of high limit of the tag

dBottomLimit

Value of low limit of the tag

dStartValue

Start value of the tag

dSubstituteValue

Substitute value of tag

LimitFlags

The LimitFlags parameter specifies the validity of the default values and limits of a tag:

MCP_VARLIM_HAS_MIN_LIMIT	Tag has a fixed low limit
MCP_VARLIM_HAS_MAX_LIMIT	Tag has a fixed high limit
MCP_VARLIM_HAS_DEFAULT_VALUE	Tag has a substitute value
MCP_VARLIM_HAS_STARTUP_VALUE	Tag has start value
MCP_VARLIM_USE_DEFAULT_ON_STARTUP	The substitute value should be used at system startup.
MCP_VARLIM_USE_DEFAULT_ON_MAX	The substitute value should be used when the high limit is violated.
MCP_VARLIM_USE_DEFAULT_ON_MIN	The substitute value should be used when the low limit is violated.
MCP_VARLIM_USE_DEFAULT_ON_COMM_ERROR	The substitute value should be used when a connection error occurs.

dwTextBibStartText

The parameter is only relevant for text tags. If the text to be used as the start value should be read from the project text, enter the ID of the corresponding text here.

szTextStartText

The parameter is only relevant for text tags. Specify the text to be used as the start value directly under szTextStartText.

dwTextBibSubstitute

The parameter is only relevant for text tags. If the text to be used as the substitute value should be read from the project text, enter the ID of the corresponding text here.

szTextSubstitute

The parameter is only relevant for text tags. Specify the text to be used as the substitute value directly under szTextSubstitute.

Comments

MCP_VARIABLE_LIMITS_EX is part of the MCP_NEWVARIABLE_DATA_EX (Page 1600) structure.

The MCP_VARIABLE_LIMITS (Page 1608) structure has a similar use.

Required files

dmclient.h

See also

MCP_VARIABLE_LIMITS (Page 1608)

MCP_NEWVARIABLE_DATA_EX (Page 1600)

MCP_NEWVARIABLE_DATA_EX4 (Page 1602)

3.2.2.33 MCP_VARIABLE_PROTOCOL (RT Professional)**Declaration**

```
typedef struct {
    BOOL    bTopLimitErr;
    BOOL    bBottomLimitErr;
    BOOL    bTransformationErr;
    BOOL    bWriteErr;
    BOOL    bWriteErrApplication;
    BOOL    bWriteErrProzess;
}
MCP_VARIABLE_PROTOCOL;
```

Members

bTopLimitErr

A report entry is generated when the value of the tag exceeds the high limit.

bBottomLimitErr

A report entry is generated when the value of the tag falls below the low limit.

bTransformationErr

A report entry is generated when a conversion error occurs.

bWriteErr

A report entry is generated with each unauthorized write access.

bWriteErrApplication

A report entry is generated with each unauthorized write access by the application.

bWriteErrProzess

A report entry is generated with each unauthorized write access by the process.

Comments

MCP_VARIABLE_PROTOCOL is part of the MCP_NEWVARIABLE_DATA (Page 1594) and MCP_NEWVARIABLE_DATA_4 (Page 1596) structures.

The MCP_VARIABLE_PROTOCOL_EX (Page 1617) structure has a similar use for extended functionality.

Required files

dmclient.h

See also

MCP_NEWVARIABLE_DATA (Page 1594)

MCP_NEWVARIABLE_DATA_4 (Page 1596)

MCP_VARIABLE_PROTOCOL_EX (Page 1617)

MCP_NEWVARIABLE_DATA_5 (Page 1598)

3.2.2.34 MCP_VARIABLE_PROTOCOL_EX (RT Professional)

Declaration

```
typedef struct {
    DWORD    dwProtocolFlags
}
MCP_VARIABLE_PROTOCOL_EX;
```

Members

dwProtocolFlags

You can use the dwProtocolFlags parameter to determine which events generate a report entry:

MCP_VARPROT_TOPLIMITERR	A report entry is generated when the value of the tag exceeds the high limit.
MCP_VARPROT_BOTTEMLIMITERR	A report entry is generated when the value of the tag falls below the low limit.
MCP_VARPROT_TRANSFORMATIONERR	A report entry is generated when a conversion error occurs.
MCP_VARPROT_WRITEERR	A report entry is generated with each unauthorized write access.
MCP_VARPROT_WRITEERRAPPLICATION	A report entry is generated with each unauthorized write access by the application.
MCP_VARPROT_WRITEERRPROCESS	A report entry is generated with each unauthorized write access by the process.

A logical combination of these constants is possible.

Comments

MCP_VARIABLE_PROTOCOL_EX is part of the MCP_NEWVARIABLE_DATA_EX (Page 1600) structure.

The MCP_VARIABLE_PROTOCOL (Page 1613) structure has a similar use.

Required files

dmclient.h

See also

MCP_VARIABLE_PROTOCOL (Page 1613)

MCP_NEWVARIABLE_DATA_EX (Page 1600)

MCP_NEWVARIABLE_DATA_EX4 (Page 1602)

3.2.2.35 MCP_VARIABLE_SCALES (RT Professional)

Declaration

```
typedef struct {  
    DWORD      dwVarScaleFlags;  
    double     doMinProc;  
    double     doMaxProc;  
    double     doMinVar;  
    double     doMaxVar;  
}  
MCP_VARIABLE_SCALES;
```

Members

dwVarScaleFlags

DM_VARSCALE_NOSCALE: No scaling

DM_VARSCALE_LINEAR: Linear scaling

doMinProc

Lowest value of the tag in the process

doMaxProc

Highest value of the tag in the process

doMinVar

Lowest value of the tag in WinCC

doMaxVar

Highest value of the tag in WinCC

Comments

MCP_VARIABLE_SCALES is part of the MCP_NEWVARIABLE_DATA_4 (Page 1596) structure.

Required files

dmclient.h

See also

MCP_NEWVARIABLE_DATA_4 (Page 1596)

MCP_NEWVARIABLE_DATA_5 (Page 1598)

DM_VARIABLE_DATA4 (Page 1588)

MCP_NEWVARIABLE_DATA_EX4 (Page 1602)

3.2.3 General functions (RT Professional)

3.2.3.1 DMActivateRTProject (RT Professional)

Use

This function enabled the project open in WinCC.

Declaration

```
BOOL DMActivateRTProject (  
    LPCMN_ERROR    lpdmError);
```

Parameters

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Project enabled.

FALSE

Error.

Comment

This function is processed internally asynchronously. If the instruction is correctly forwarded internally, it always returns TRUE. If, for example, the name of the server computer in the project is specified incorrectly, no error is reported.

A successful start is checked with DMGetRTProject(..) .

If a check for a correct computer name is required, use the DMGetMachineTable(..) and GetComputername(..) functions.

Required files

dmclient.h
dmclient.lib
dmclient.dll

See also

DMGetMachineTable (Page 1650)

3.2.3.2 DMAAddNotify (RT Professional)

Declaration

```
BOOL DMAAddNotify (  
    DM_NOTIFY_PROC    lpfnNotify,  
    LPVOID             lpvUser,  
    LPDWORD            lpdwNotifyCookie,  
    LPCMN_ERROR        lpdmError);
```

Description

Connect a further notify function call to DMClient. The notify functionality corresponds with the functionality you can specify for DMConnect.

Parameters

lpfnNotify

Pointer to further notify functions.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the notify function.

lpdwNotifyCookie

Pointer to a DWORD, in which a cookie to the notify is returned.

The cookie is later required again in the DMRemoveNotify function to remove notify editing again.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Additional notify function connected.

FALSE

Error

Comments

The function is required in case a DMConnect has been performed within the process already but no access is possible to the notify function specified there. This way, an additional notify can be included to also evaluate the notify functions involved.

If, e.g., you use a DLL with DMClient functionality you created yourself within a global script function, it is no longer possible to perform a DMConnect there. A DMConnect has been performed already using the script and notification is no longer possible.

On rare occasions, the notify may be returned even before the function call has returned.

The notify functions included here in addition must be removed again using DMRemoveNotify before DMDisconnect. Also, the notify function must not be a member function of a class object which can be destroyed or deleted prematurely. Eventually, a notify call is pendant following a DMRemoveNotify.

Error messages

DM_E_NOT_CONNECTED	No DMConnect performed yet. The internal administration has not been initialized yet.
DM_E_ALREADY_EXIST	A NotifyProc with the same address exists already in this process.
DM_E_MAX_LIMIT	Maximum number of additional notify functions has been reached (16).
DM_E_PARAM	Parameter error

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMFireNotifyData	Sends data to the additional notify function using the cookie or to all functions
DMRemoveNotify	Removes the additional notify function using the cookie from the notify list

Examples

Button AddNotify:

```
#include "apdefap.h"
void OnClick(char* lpszPictureName,
             char* lpszObjectName,
             char* lpszPropertyName)
{
    AddNotify();
}
```

Project function AddNotify:

```
extern BOOL DM_NotifyProcA(DWORD dwNotifyClass,
                          DWORD dwNotifyCode,
                          LPBYTE lpbyData,
                          DWORD dwItems,
                          LPVOID lpvUser);

void AddNotify()
{
    BOOL bRet = FALSE;
    DWORD dwNotifyCookie = 0L;
    LPVOID lpvUser = NULL;
    CMN_ERRORA err;

    memset(&err, 0, sizeof(CMN_ERRORA));

    dwNotifyCookie = GetTagDWord("DMdwNotifyCookie");
    if (dwNotifyCookie)
    {
        printf("\r\nremove first previous Notify Cookie=%08lx",
              dwNotifyCookie);

        bRet = DMRemoveNotifyA(dwNotifyCookie, &err);

        if (FALSE == bRet)
        {
            printf("\r\nERROR: DMRemoveNotifyA [%s],%ld,%ld,%ld,%ld,%ld",
                  err.szErrorText,
                  err.dwError1,
                  err.dwError2,
                  err.dwError3,
                  err.dwError4,
                  err.dwError5);
        }
        dwNotifyCookie = 0L;
        SetTagDWord("DMdwNotifyCookie", dwNotifyCookie);
    }

    memset(&err, 0, sizeof(CMN_ERRORA));
    lpvUser = (LPVOID)dwNotifyCookie; //set only for show in notify
}
```

```

bRet = DMAddNotifyA(DM_NotifyProcA,
                    lpvUser,
                    &dwNotifyCookie,
                    &err);

if (bRet)
{
    printf("\r\nNotify added, Cookie=%08lx!", dwNotifyCookie);
    SetTagDWord("DMdwNotifyCookie", dwNotifyCookie);
}
else
{
    printf("\r\nERROR: DMAddNotifyA [%s],%ld,%ld,%ld,%ld,%ld",
          err.szErrorText,
          err.dwError1,
          err.dwError2,
          err.dwError3,
          err.dwError4,
          err.dwError5);
}
}

```

Project function DM_NotifyProcA:

```
extern void RemoveNotify();
```

```

BOOL DM_NotifyProcA(DWORD dwNotifyClass,
                   DWORD dwNotifyCode,
                   LPBYTE lpbyData,
                   DWORD dwItems,
                   LPVOID lpvUser)
{
    BOOL bShowParams = FALSE;
    BOOL bSpecialActionRemoveNotify = FALSE;

    printf("\r\n\r\n#### extra added DMNotifyProc [%08lx] entry: ####",
           (DWORD)lpvUser);
    switch (dwNotifyClass)
    {
    case DM_NOTIFY_CLASS_ERROR:
        switch (dwNotifyCode)
        {
        case DM_NOTIFY_SHUTDOWN:
            printf("\r\nDM_NOTIFY_CLASS_ERROR: DM_NOTIFY_SHUTDOWN");
            bShowParams = TRUE;
            bSpecialActionRemoveNotify = TRUE;
            break;
        case DM_NOTIFY_PROCESSNET_ERROR:
            printf("\r\nDM_NOTIFY_CLASS_ERROR:
DM_NOTIFY_PROCESSNET_ERROR");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_SYSNET_ERROR:
            printf("\r\nDM_NOTIFY_CLASS_ERROR:
DM_NOTIFY_SYSNET_ERROR");

```

3.2 Data management functions (RT Professional)

```

        bShowParams = TRUE;
        break;
    default:
        printf("\r\nunknown DM_NOTIFY_CLASS_ERROR:
dwNotifyCode[%08lx]", dwNotifyCode);
        bShowParams = TRUE;
    }
    break;
case DM_NOTIFY_CLASS_WARNING:
    switch (dwNotifyCode)
    {
        case DM_NOTIFY_QUEUE_50_PERCENT:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_QUEUE_50_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_60_PERCENT:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_QUEUE_60_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_70_PERCENT:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_QUEUE_70_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_80_PERCENT:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_QUEUE_80_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_90_PERCENT:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_QUEUE_90_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_OVERFLOW:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_QUEUE_OVERFLOW");
            break;
        case DM_NOTIFY_CYCLES_CHANGED:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_CYCLES_CHANGED");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_MACHINES_CHANGED:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_MACHINES_CHANGED");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_PROJECT_OPENED:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_PROJECT_OPENED [%s]",
                (LPSTR) lpbyData);
            bSpecialActionRemoveNotify = TRUE;
            break;
        case DM_NOTIFY_PROJECT_CLOSED:

```

```

        printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_PROJECT_CLOSE");
        bSpecialActionRemoveNotify = TRUE;
        break;
    case DM_NOTIFY_SYSTEM_LOCALE:
        printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_SYSTEM_LOCALE [%ld]",
            *(DWORD*)lpbyData);
        break;
    case DM_NOTIFY_DATA_LOCALE:
        printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_DATA_LOCALE [%ld]",
            *(DWORD*)lpbyData);
        break;
    case DM_NOTIFY_PROJECT_RUNTIME:
        printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_PROJECT_RUNTIME [%s]",
            (LPSTR)lpbyData);
        break;
    case DM_NOTIFY_PROJECT_EDIT:
        printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_PROJECT_EDIT [%s]",
            (LPSTR)lpbyData);
        bSpecialActionRemoveNotify = TRUE;
        break;
    case DM_NOTIFY_HOTKEY_CHANGE:
        printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_HOTKEY_CHANGE");
        bShowParams = TRUE;
        break;
    case DM_NOTIFY URSEL:
        printf("\r\nDM_NOTIFY_CLASS_WARNING: DM_NOTIFY URSEL");
        bShowParams = TRUE;
        break;
    case DM_NOTIFY_BODO:
        printf("\r\nDM_NOTIFY_CLASS_WARNING: DM_NOTIFY_BODO");
        bShowParams = TRUE;
        break;
    case DM_NOTIFY_BEGIN_PROJECT_EDIT:
        printf("\r\nDM_NOTIFY_CLASS_WARNING:
DM_NOTIFY_BEGIN_PROJECT_EDIT [%s]",
            (LPSTR)lpbyData);
        bShowParams = TRUE;
        bSpecialActionRemoveNotify = TRUE;
        break;
    default:
        printf("\r\nunknown DM_NOTIFY_CLASS_WARNING:
dwNotifyCode[%08lx]", dwNotifyCode);
        bShowParams = TRUE;
    }
    break;
case DM_NOTIFY_CLASS_DATA:

```

3.2 Data management functions (RT Professional)

```

        switch (dwNotifyCode)
        {
            case DM_NOTIFY_APPLICATION_DATA:
                printf("\r\nDM_NOTIFY_CLASS_DATA:
DM_NOTIFY_APPLICATION_DATA:");
                bShowParams = TRUE;
                break;
            case DM_NOTIFY_VARIABLE_DATA:
                printf("\r\nDM_NOTIFY_CLASS_DATA:
DM_NOTIFY_VARIABLE_DATA:");
                bShowParams = TRUE;
                break;
            case DM_NOTIFY_FIRE_DATA:
                // the data sended with DMFireNotifyData is char text
                printf("\r\nDM_NOTIFY_CLASS_DATA: DM_NOTIFY_FIRE_DATA:
(data as text: [%s])",
                    (char*)lpbyData);
                bShowParams = TRUE;
                break;
            default:
                printf("\r\nunknown DM_NOTIFY_CLASS_DATA:
dwNotifyCode[%08lx]", dwNotifyCode);
                bShowParams = TRUE;
        }
        break;
    default:
        printf("\r\nunknown dwNotifyClass[%08lx],
dwNotifyCode[%08lx]",
            dwNotifyClass,
            dwNotifyCode);
        bShowParams = TRUE;
    }

    if (bShowParams)
    {
        printf("\r\ndwNotifyClass=%08lx, dwNotifyCode=%08lx,
dwItems=%ld, lpbyData=%08lx, lpvUser=%08lx",
            dwNotifyClass,
            dwNotifyCode,
            dwItems,
            lpbyData,
            (DWORD)lpvUser);
    }

    if (bSpecialActionRemoveNotify)
    {
        printf("\r\nHave to remove notify if RT exists:");
        RemoveNotify();
    }

    printf("\r\n#### extra added Notify [%08lx] exit ####\r\n",
        (DWORD)lpvUser);

```



```
return TRUE;  
}
```

3.2.3.3 DMChangeDataLocale (RT Professional)

Use

Provides notification of switching of the runtime language to all applications connected with DMConnect.

Declaration

```
BOOL DMChangeDataLocale (  
    LPCTSTR        lpszProjectFile,  
    DWORD          dwLocaleID,  
    LPCMN_ERROR    lpdmError );
```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

dwLocaleID

Pointer to the code of the newly selected language. Possible values are the codes of all languages configured in the text library.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Notify applications when language is switched.

FALSE

Error.

Comment

The application is notified via the callback function DM_NOTIFY_PROC (Page 1631) with the following parameters:

dwNotifyClass	DM_NOTIFY_CLASS_WARNING
dwNotifyCode	DM_NOTIFY_DATA_LOCALE
lpbyData	Pointer to the code of the newly configured language.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_NOT_SUPPORTED	Requested service unavailable

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

AUTOHOTSPOT	Make connection to the data manager

See also

DM_NOTIFY_PROC (Page 1631)

DMEnumOpenedProjects (Page 1654)

DMGetRuntimeProject (Page 1660)

3.2.3.4 DMConnect (RT Professional)

Use

Establishment of a connection by application to the data manager. Only one DMConnect can be performed in an application (process). Additional calls return the error DM_E_ALREADY_CONNECTED.

Declaration

```
BOOL DMConnect (
    LPTSTR          lpszAppName,
    DM_NOTIFY_PROC  lpfnNotify,
    LPVOID          lpvUser,
    LPCMN_ERROR      lpdmError );
```

Parameters

lpszAppName

Pointer to the name of the calling application. Any name can be selected since the parameter is used for internal identification.

The length of application name is limited to MAX_DM_APP_NAME (32 characters). Longer names lead to errors in the embedded OHIOIPC.DLL and to termination with the error DM_E_NOT_CONNECTED.

lpfnNotify

Pointer to the notification function for administrative messages from the data manager to the application.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

The Notify routine should still be available after DMDisconnect, since the late calls may arrive.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function, but is made available again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Set up connection to data manager.

FALSE

Error.

Comment

In the WinCC variants that are implemented on the operating system platforms with multi-threading support, the calls for the notification function are executed asynchronous to the calling process.

You need to configure appropriate synchronization within the application.

If data manager functions are distributed to multiple threads in a program, run DMConnect in each of these threads and a DMDisconnect at the end.

First, use DMGetConnectionState to check if another thread has already established a connection. If a connection already exists, you may not perform DMConnect or DMDisconnect. To avoid this conflict, combine the function calls to the data manager in a single thread and execute the calls only from there.

If data manager functions are swapped out to a DLL that is embedded in an application with the data manager functions, check if the connection can be established and terminated via the DLL. In this case, the application must always secure its own data manager function calls with DMGetConnectionState. However, this offers no protection against disconnection when the thread is encapsulated in the DLL.

Therefore, define your own connection establishment in a flag, so that DMDisconnect is only carried out in the desired situation.

An unchecked DMConnect call with Notify that returns the DM_E_ALREADY_CONNECTED error message may cancel the Notify of a previous call and lead to undesirable results.

Pointers to data ranges passed to the callback function are only valid within the function, in other words, the data manager may release allocated memory areas after the return of the callback. If the application requires access to the data beyond the time of the call, it must copy it accordingly.

In addition, pointers to memory areas on the platforms that only support this only have read access, so that a write access to relevant data cause a protection fault!

The return value of the callback is currently not evaluated and will only be of significance in future versions. The application should return TRUE by default here.

API DLLs must not be directly used in (ISS) services because the internal resources required there are not available.

If you use data manager functions, such as a wrapper, in a managed environment (C#), ensure that multiple ApplicationDomains in an application run in the same process. Therefore, only one DMConnect is permitted in all ApplicationDomains.

Error messages

DM_E_ALREADY_CONNECTED	Connection to the data manager already established.
DM_E_NOT_CONNECTED	Connection to the data manager could not be established (for example, application name too long, etc.).
DM_E_ACCESS_FAULT	No right to establish connection to data manager (Invalid user context, for example ISS services, etc.).

Required files

dmclient.h
 dmclient.lib
 dmclient.dll

Related functions

DMDDisconnect (Page 1634)	Terminate connection to the data manager
DMGetConnectionState (Page 1646)	Check connection to the data manager
DM_NOTIFY_PROC (Page 1631)	Notification function

Examples

Connection to DM (Page 1823) "DM01.cpp"

See also

DM_NOTIFY_PROC (Page 1631)
 DMDDisconnect (Page 1634)
 DMGetConnectionState (Page 1646)
 Connection to DM (Page 1823)
 APConnect (Page 1949)

3.2.3.5 DM_NOTIFY_PROC (RT Professional)**Description**

In order for your application to provide notification of language switching, for example, you need to provide a DM_NOTIFY_PROC type callback function.

Declaration

```

BOOL ( * DM_NOTIFY_PROC) (
    DWORD      dwNotifyClass,
    DWORD      dwNotifyCode,
    LPBYTE     lpbyData,
    DWORD      dwItems,
    LPVOID     lpvUser);

```

Parameters

dwNotifyClass

identifies the notification class:

DM_NOTIFY_CLASS_ERROR	(0x00000001)
DM_NOTIFY_CLASS_WARNING	(0x00000002)
DM_NOTIFY_CLASS_DATA	(0x00000003)

dwNotifyCode

Notification code

for DM_NOTIFY_CLASS_ERROR:		
DM_NOTIFY_SHUTDOWN	(0x00000001)	Data manager is closed
DM_NOTIFY_PROCESSNET_ERROR	(0x00000002)	Error on the process bus
DM_NOTIFY_SYSNET_ERROR	(0x00000003)	Error on the system bus
for DM_NOTIFY_CLASS_WARNING:		
DM_NOTIFY_QUEUE_50_PERCENT	(0x00000001)	Fill level of application queue 50%
DM_NOTIFY_QUEUE_60_PERCENT	(0x00000002)	Fill level of application queue 60%
DM_NOTIFY_QUEUE_70_PERCENT	(0x00000003)	Fill level of application queue 70%
DM_NOTIFY_QUEUE_80_PERCENT	(0x00000004)	Fill level of application queue 80%
DM_NOTIFY_QUEUE_90_PERCENT	(0x00000005)	Fill level of application queue 90%
DM_NOTIFY_QUEUE_OVERFLOW	(0x00000006)	Overflow of the application queue
DM_NOTIFY_CYCLES_CHANGED	(0x00000010)	Rescan update cycles
DM_NOTIFY_MACHINES_CHANGED	(0x00000011)	Rescan computer list
DM_NOTIFY_PROJECT_OPENED	(0x00000012)	Load project
DM_NOTIFY_PROJECT_CLOSE	(0x00000013)	Close project
DM_NOTIFY_SYSTEM_LOCALE	(0x00000014)	Switch configuration language
DM_NOTIFY_DATA_LOCALE	(0x00000015)	Switch GUI language
DM_NOTIFY_PROJECT_RUNTIME	(0x00000016)	Enable project
DM_NOTIFY_PROJECT_EDIT	(0x00000017)	Disable project
DM_NOTIFY_HOTKEY_CHANGE	(0x00000018)	A hot key was changed
for DM_NOTIFY_CLASS_DATA:		
DM_NOTIFY_APPLICATION_DATA	(0x00000001)	Application data
DM_NOTIFY_VARIABLE_DATA	(0x00000002)	Tag data

You can find more information in the notification codes in the description of the constants.

lpbyData

Pointer to the data provided within the DM_NOTIFY_CLASS_DATA class.

dwItems

Number of entries in lpbyData.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

The return values depend on your implementation.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the DMConnect function call has returned.

Required files

dmclient.h

Related functions

DMConnect (Page 1626)	Make connection to the data manager

Examples

AUTOHOTSPOT "DM01.cpp"

See also

DMConnect (Page 1626)

3.2.3.6 DMDeactivateRTProject (RT Professional)**Use**

The project in the runtime mode is disabled.

Declaration

```
BOOL DMDeactivateRTProject (  
    LPCMN_ERROR    lpdmError);
```

Parameters

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Project disabled.

FALSE

Error.

Error messages

DM_E_NO_RT_PRJ	No project in runtime mode

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

OnTestDeactivateRuntimeProject (Page 1835) "TESTCDoc.cpp"

See also

OnTestDeactivateRuntimeProject (Page 1835)

3.2.3.7 DMDDisconnect (RT Professional)

Use

This function is used by an application to terminate an existing connection to the data manager.

Declaration

```
BOOL DMDisConnect (
    LPCMN_ERROR    lpdmError );
```

Parameters

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection terminated.

FALSE

Error.

Comments

If no connection to the data manager has been established with the DMConnect function, the return value is FALSE. The lpdmError->dwError error code contains the DM_E_NOT_CONNECTED, value, no connection to the data manager.

Note

The call must not be used in the destructor of an application (EXE, DLL, OCX, etc.). Due to Microsoft-specific mechanisms, this can cause the call to freeze and thereby crash the program.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DMConnect (Page 1626)	Make connection to the data manager

Examples

Connection to DM (Page 1823) "DM01.cpp"

See also

DMConnect (Page 1626)

Connection to DM (Page 1823)

3.2.3.8 DMEnumNumberFormats (RT Professional)**Use**

The function lists all format conversions available within FORMAT.DLL .

Declaration

```

BOOL DMEnumNumberFormats (
    LPDWORD          lpdwItems,
    DM_ENUM_FORMATS_PROC lpfnFormat,
    LPVOID           lpvUser,
    LPCMN_ERROR      lpdmError);

```

Parameters**lpdwItems**

Pointer to a double word of the application, which contains the number of enumerated format data after the call.

lpfnFormat

Pointer to your callback function that is called for every available number format.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Format conversions listed.

FALSE

Error.

Comments

Pointers passed to the callback are only valid within this, since the system releases all allocated memory areas once the function returns. If an application requires access to the data beyond this, it must copy it accordingly.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DM_ENUM_FORMATS_PROC (Page 1637)	List format conversions (callback)

See also

DM_ENUM_FORMATS_PROC (Page 1637)

3.2.3.9 DM_ENUM_FORMATS_PROC (RT Professional)**Description**

In order to evaluate the format conversions listed by the system, you must provide a DM_ENUM_FORMATS_PROC type callback function.

Declaration

```
BOOL ( * DM_ENUM_FORMATS_PROC) (
    LPDM_FORMAT_INFO    lpdmFormat,
    DWORD               dwItem,
    LPVOID               lpvUser );
```

Parameters

- lpdmFormat**
Pointer to a DM_FORMAT_INFO (Page 1569) type structure with the data of a format conversion.
- dwItem**
Continuous call counter. If the enumeration is not canceled prematurely, dwItem will contain the number of available format conversions.
- lpvUser**
Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

- TRUE**
The enumeration is continued.
- FALSE**
The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

dmclient.h

Related functions

DMEnumNumberFormats (Page 1634)	List format conversions

See also

DMEnumNumberFormats (Page 1634)

DM_FORMAT_INFO (Page 1569)

3.2.3.10 DMEnumUpdateCycles (RT Professional)**Use**

The function lists all update cycles defined in the system. The function passes an information structure for the cycle each time the callback function is called.

Declaration

```

BOOL DMEnumUpdateCycles (
    LPCSTR          lpszProjectFile,
    LPDWORD         lpdwItems,
    DM_ENUM_CYCLES_PROC lpfnCycle,
    LPVOID          lpvUser,
    LPCMN_ERROR     lpdmError);

```

Parameters**lpszProjectFile**

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

lpdwItem

Pointer to a double word of the application, which contains the number of enumerated cycle data after the call.

lpfnCycle

Pointer to your callback function that is called for every available update cycle.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function, but is made available again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Update cycles listed

FALSE

Error

Comments

Pointers passed to the callback function are only valid within the function, since the system releases all allocated memory areas once the function returns.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_CYCLES_PROC (Page 1640)	List update cycles (callback)

Examples

OnTestUpdateCycles (Page 1843) "TESTCDoc.cpp"

See also

DM_ENUM_CYCLES_PROC (Page 1640)

OnTestUpdateCycles (Page 1843)

3.2.3.11 DM_ENUM_CYCLES_PROC (RT Professional)**Description**

In order to evaluate the update cycles listed by the system, you must provide a DM_ENUM_CYCLES_PROC type callback function.

Declaration

```

BOOL ( * DM_ENUM_CYCLES_PROC) (
    LPDM_CYCLE_INFO    lpdmCycle,
    DWORD              dwItem,
    LPVOID              lpvUser);

```

Parameters

lpdmCycle

Pointer to a DM_CYCLE_INFO (Page 1565) type structure with the data of an update cycle.

dwItem

Continuous call counter. If the enumeration is not canceled prematurely, dwItem will contain the number of available update cycles.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

dmclient.h

Related functions

DMEnumUpdateCycles (Page 1637)	List update cycles

See also[DMEEnumUpdateCycles \(Page 1637\)](#)[DM_CYCLE_INFO \(Page 1565\)](#)**3.2.3.12 DMEExitWinCC (RT Professional)****Use**

You can use this function to close WinCC.

Declaration

```
BOOL DMEExitWinCC (  
    VOID);
```

Parameters

None

Comments

You can also use the DMEExitWinCCEx to specify how the operating system should react to the closing of WinCC.

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMEExitWinCCEx (Page 1643)	Close WinCC with defaults

Examples

[OnTestWinCCShutdown \(Page 1860\)](#) "TESTCDoc.cpp"

See also[OnTestWinCCShutdown \(Page 1860\)](#)[DMEExitWinCCEx \(Page 1643\)](#)

3.2.3.13 DMEExitWinCCEx (RT Professional)

Use

You can use this function to close WinCC. You can also specify how the operating system reacts after WinCC is closed.

Declaration

```
BOOL DMEExitWinCCEx (
    DWORD    dwMode );
```

Parameters

dwMode

You can use dwMode to specify various ways the operating system should react after WinCC is closed.

DM_SDMODE_WINCC	WinCC is closed
DM_SDMODE_LOGOFF	When WinCC is closed, the user is also logged off of the OS.
DM_SDMODE_SYSTEM	When WinCC is closed, the operating system is shut down.
DM_SDMODE_REBOOT	When WinCC is closed, the operating system is restarted.

Comments

With the DM_SDMODE_WINCC parameter, the function has the same effect as DMEExitWinCC.

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DMEExitWinCC (Page 1640)	Close WinCC

See also

DMEExitWinCC (Page 1640)

3.2.3.14 DMFireNotifyData (RT Professional)

Declaration

```
BOOL DMFireNotifyData (
    DWORD      dwNotifyCookie,
    DWORD      dwByteCount,
    LPBYTE     lpbyData,
    LPCMN_ERROR lpdmError);
```

Description

Sends an additional notify to a connected notify using the specified cookie identifier.

Parameters

dwNotifyCookie

Cookie for an additional notify function which was connected using DMAddNotify.

If "0" is specified the notify data is sent to all active notifies (broadcast).

dwByteCount

Size of the data buffer to be sent in BYTE.

lpbyData

Pointer to a byte buffer that contains the notify data to be sent.

The size of the buffer is specified by dwByteCount.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Notify data sent.

FALSE

Error

Comments

The sent notify is edited in the said notify routine under
dwNotifyClass=DM_NOTIFY_CLASS_DATA and dwNotifyCode=DM_NOTIFY_FIRE_DATA.

If a DMDisconnect is performed without previously removing the additional notify functions, a final DM_NOTIFY_SHUTDOWN is issued again to all these notify functions which are then removed internally using administration. You can then discard the connection and the cookies as they are no longer valid.

Error messages

DM_E_NOT_CONNECTED	No DMConnect performed yet or DMDisconnect performed already
DM_E_DONT_EXIST	No notify exists for the specified cookie
DM_E_PARAM	Parameter error

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMAAddNotify	Links an additional Notify function to "DMClient"
DMFireNotify	Removes the additional notify function using the cookie from the notify list

Examples

Button DMFireNotifyData:

```
#include "apdefap.h"
void OnClick(char* lpszPictureName,
             char* lpszObjectName,
             char* lpszPropertyName)
{
    DWORD dwNotifyCookie;
    CHAR* pszNotifyText;
    CMN_ERROR err;
    BOOL bRet;

    dwNotifyCookie = GetTagDWord("DMdwNotifyCookie");
    pszNotifyText = GetTagChar("DMNotifySendText");
    memset(&err, 0, sizeof(err));
    bRet = FALSE;

    bRet = DMFireNotifyData(dwNotifyCookie,
                           strlen(pszNotifyText)*sizeof(CHAR)+1,
```

```

                                (LPBYTE)pszNotifyText, &err);
if (FALSE == bRet)
{
printf("\r\nERROR: DMFireNotifyData [%s],%ld,%ld,%ld,%ld,%ld",
    err.szErrorText,
    err.dwError1,
    err.dwError2,
    err.dwError3,
    err.dwError4,
    err.dwError5);
}
}
```

3.2.3.15 DMGetConnectionState (RT Professional)

Use

You can use this function to query for a connection to the data manager, for example, to check whether the DMConnect function call has been correctly executed.

Declaration

```

BOOL DMGetConnectionState (
    LPCMN_ERROR    lpdmError);
```

Parameters

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection established.

FALSE

Error or no connection.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h
 dmclient.lib
 dmclient.dll

Related functions

DMConnect (Page 1626)	Make connection to the data manager

Examples

AUTOHOTSPOT "DM01.cpp"

See also

DMConnect (Page 1626)
 Connection to DM (Page 1823)

3.2.3.16 DMGetDataLocale (RT Professional)**Use**

The function returns the ID of the currently selected runtime language.

Declaration

```

BOOL DMGetDataLocale (
    LPDWORD          lpdwLocaleID,
    LPCMN_ERROR      lpdmError);

```

Parameters**lpdwLocaleID**

Pointer to the code of the currently configured language. Return values are the codes of all languages configured in the text library.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Get language ID.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

3.2.3.17 DMGetHotkey (RT Professional)**Use**

This function determines the action associated with a hotkey.

Declaration

```
BOOL DMGetHotKey (  
    DWORD          dwHotKeyAction,  
    LPDWORD        lpdwHotKey,  
    LPCMN_ERROR     lpdmError);
```

Parameters**dwHotKeyAction**

Action

lpdwHotKey

Pointer to the memory location where the hotkey is stored.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Get hotkey ID.

FALSE

Error.

Required files

dmclient.h

dmclient.lib

dmclient.dll

3.2.3.18 DMGetMachineInfo (RT Professional)**Use**

This function is used by an application to determine the startup parameters of the specific computer. For this to occur, the application must have the relevant entries for the computer properties made with the project management functions beforehand.

Declaration

```
BOOL DMGetMachineInfo (  
    LPCSTR      lpszLogicalName,  
    LPVOID      lpvData,  
    LPDWORD     lpdwSize,  
    LPCMN_ERROR lpdmError);
```

Parameters**lpszLogicalName**

Pointer to the logical name of the Explorer DLL whose startup parameters are queried.

lpvData

Pointer to a data range through which the data of the startup parameters are passed to the application.

lpdwSize

If lpvData = 0, the size of the data range is specified in lpdwSize.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Get information about the local PC.

FALSE

Error.

Error messages

DM_E_NO_RT_PRJ	No project in runtime mode
DM_E_MACHINE_NOT_FOUND	Cannot find PC
DM_E_NO_INFO_FOUND	Cannot find startup information

Required files

dmclient.h

dmclient.lib

dmclient.dll

3.2.3.19 DMGetMachineTable (RT Professional)**Use**

You can use this function to determine information on individual PCs in addition to the number of PCs involved in the project.

Declaration

```

BOOL DMGetMachineTable (
    LPCSTR          lpszProjectFile,
    LPDM_MACHINE_TABLE lpdmMachineTable,
    LPCMN_ERROR      lpdmError);

```

Parameters**lpdwProjectFile**

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with `DMEnumOpenedProjects` or with `DMGetRuntimeProject` in RT.

lpdmMachineTable

Pointer to the `DM_MACHINE_TABLE` (Page 1570) structure in which the data from the computer list is stored.

lpdmError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Data for computer list determined.

FALSE

Error.

Required files

dmclient.h
dmclient.lib
dmclient.dll

Examples

OnTestMachines (Page 1838) "TESTCDoc.cpp"

See also

`DM_MACHINE_TABLE` (Page 1570)
OnTestMachines (Page 1838)
`DMActivateRTProject` (Page 1617)

3.2.3.20 DMRemoveNotify (RT Professional)**Declaration**

```
BOOL DMRemoveNotify (  
    DWORD          dwNotifyCookie,  
    LPCMN_ERROR     lpdmError);
```

Description

Removes an additionally connected notify function using its cookie identifier from the additional notify list.

This function is only available in WinCC Version V7.2 or higher.

Parameters

dwNotifyCookie

Cookie for an additional notify function which was connected using DMAddNotify.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Additional notify function removed.

FALSE

Error

Comments

If a DMDisconnect is performed without previously removing the additional notify functions, a final DM_NOTIFY_SHUTDOWN is issued again to all these notify functions which are then removed internally using administration. You can then discard the connection and the cookies as they are no longer valid.

Error messages

DM_E_NOT_CONNECTED	No DMConnect performed yet or DMDisconnect performed already
DM_E_DONT_EXIST	No notify exists for the specified cookie
DM_E_PARAM	Parameter error

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMFireNotifyData	Sends data to the additional notify function using the cookie or to all functions
DMAddNotify	Links an additional Notify function to "DMClient"

Examples**Button RemoveNotify:**

```
#include "apdefap.h"
void OnClick(char* lpszPictureName,
             char* lpszObjectName,
             char* lpszPropertyName)
{
    RemoveNotify ();
}
```

Project function RemoveNotify:

```
void RemoveNotify()
{
    DWORD dwNotifyCookie = 0L;
    CMN_ERRORA err;
    BOOL bRet = FALSE;

    memset(&err, 0, sizeof(CMN_ERRORA));
    dwNotifyCookie = GetTagDWord("DMdwNotifyCookie");

    if (dwNotifyCookie)
    {
        bRet = DMRemoveNotify(dwNotifyCookie, &err);
        if (bRet)
        {
            printf("\r\nNotify [%08lx] removed", dwNotifyCookie);
            dwNotifyCookie = 0L;
            SetTagDWord("DMdwNotifyCookie", dwNotifyCookie);
        }
        else
        {
            printf("\r\nERROR: DMRemoveNotifyA [%s],%ld,%ld,%ld,%ld,%ld, Cookie=%08lx",
                    err.szErrorText,
                    err.dwError1,
                    err.dwError2,
                    err.dwError3,
                    err.dwError4,
                    err.dwError5,
                    dwNotifyCookie);
        }
    }
    else
    {
        printf("\r\nNo Notify present to remove!");
    }
}
```

```
}  
}
```

3.2.4 Functions for project management (RT Professional)

3.2.4.1 DMLenumOpenedProjects (RT Professional)

Use

Calls the passed callback function for each project open in WinCC.

Declaration

```
BOOL DMLenumOpenedProjects (  
    LPDWORD                                lpdwItems,  
    DM_ENUM_OPENED_PROJECTS_PROC          lpfnEnum,  
    LPVOID                                 lpvUser,  
    LPCMN_ERROR                           lpdmError);
```

Parameters

lpdwItems

Pointer to a double word, which contains the number of open projects when the enumeration is completed.

lpfnEnum

Pointer to your callback function, which receives the project data.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Projects listed.

FALSE

Error.

Comments

Currently, only one project can be open in WinCC. You therefore only get the information for this one project.

Error messages

DM_E_FILE	Error with file operation

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DM_ENUM_OPENED_PROJECTS_PROC (Page 1655)	List open projects (callback)

Examples

Enum open projects (Page 1828) "DM01.cpp"
OnTestOpenProjects (Page 1842) "TESTCDoc.cpp"

See also

DM_ENUM_OPENED_PROJECTS_PROC (Page 1655)
DM_PROJECT_INFO (Page 1571)
Enum open projects (Page 1828)
OnTestOpenProjects (Page 1842)
DMChangeDataLocale (Page 1625)

3.2.4.2 DM_ENUM_OPENED_PROJECTS_PROC (RT Professional)**Description**

In order to be able to evaluate the projects listed by the system, you must provide a callback function of the DM_ENUM_OPENED_PROJECTS_PROC type.

Declaration

```
BOOL ( * DM_ENUM_OPENED_PROJECTS_PROC) (
    LPDM_PROJECT_INFO    lpInfo,
    LPVOID                lpvUser);
```

Parameters

- lpInfo**
Pointer to a DM_PROJECT_INFO (Page 1571) type structure with information about an open project.
- lpvUser**
Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

- TRUE**
The enumeration is continued.
- FALSE**
The enumeration is cancelled.

Comments

Currently, only one project can be open in WinCC. You therefore only get the information for this one project.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

dmclient.h

Related functions

DMEnumOpenedProjects (Page 1652)	List projects (open)

Examples

Enum open projects "DM01.cpp"

See also

DMEnumOpenedProjects (Page 1652)

DM_PROJECT_INFO (Page 1571)

Enum open projects (Page 1828)

3.2.4.3 DMGetProjectDirectory (RT Professional)

Use

Returns the path to the configuration data for the given project that is valid for the application.

Declaration

```
BOOL DMGetProjectDirectory (
    LPCSTR      lpszAppName,
    LPCSTR      lpszProjectFile,
    LPDM_DIRECTORY_INFO lpdmdirInfo,
    LPCMN_ERROR lpdmError );
```

Parameters

lpszAppName

Pointer to the name of the application for which the path is to be determined.

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be determined with DMEnumOpenedProjects.

lpdmDirInfo

Pointer to the DM_DIRECTORY_INFO (Page 1567) structure in which the path of the configuration data is stored.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Get path information.

FALSE

Error.

Error messages

DM_E_FILE	Error with file operation

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

OnTestProjectPaths (Page 1840) "TESTCDoc.cpp"

See also

DM_DIRECTORY_INFO (Page 1567)

OnTestProjectPaths (Page 1840)

3.2.4.4 DMGetProjectInformation (RT Professional)**Use**

Determines relevant information about the given project, such as paths, data source name, computer configuration, etc.

Declaration

```
BOOL DMGetProjectInformation (  
    LPCSTR          lpszProjectFile,  
    LPDPM_PROJECT_INFO lpProjectInfo,  
    LPCMN_ERROR      lpdmError);
```


Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be determined with DMEnumOpenedProjects or in RT with DMGetRuntimeProject.

lpProjectInfo

Pointer to the DM_PROJECT_INFO (Page 1571) structure in which the project information is stored.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Get project information

FALSE

Error

Error messages

DM_E_FILE	Error with file operation

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

Inquire project informations (Page 1834) "DM01.cpp"

OnTestProjectInfo (Page 1839) "TESTCDoc.cpp"

See also

DM_PROJECT_INFO (Page 1571)

Inquire project informations (Page 1834)

OnTestProjectInfo (Page 1839)

3.2.4.5 DMGetRuntimeProject (RT Professional)

Use

Returns the file name of the project that is in online mode. To determine all other project data after successful execution, the calling application remembers the name of the project file entered in `lpszProjectFile`.

Declaration

```
BOOL DMGetRuntimeProject (  
    LPSTR        lpszProjectFile,  
    DWORD        dwBufSize,  
    LPCMN_ERROR  lpdmError);
```

Parameters

lpszProjectFile

Buffer for storing the name of the project file, including path and extension. The buffer must be at least `_MAX_PATH` characters in size.

dwBufSize

Size of the given buffer in characters.

lpdmError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Name of the enabled project determined.

FALSE

Error.

Comments

The call is successfully executed even when runtime has started but not all components are enabled.

Error messages

DM_E_FILE	Error with file operation
DM_E_NO_RT_PRJ	No project in runtime mode

Required files

dmclient.h
dmclient.lib
dmclient.dll

Examples

Read tag (Page 1862) "DM02.cpp"
Write tag (Page 1867) "DM02.cpp"
OnTestRuntimeProject (Page 1842) "TESTCDoc.cpp"

See also

Read tag (Page 1862)
Write tag (Page 1867)
OnTestRuntimeProject (Page 1842)
DMChangeDataLocale (Page 1625)

3.2.4.6 DMOpenProjectDocPlus (RT Professional)**Use**

Opens the project specified in lpszProjectFile.

Declaration

```

BOOL DMOpenProjectDocPlus (
    LPSTR          lpszProjectFile,
    LPCMN_ERROR    lpdmError);

```

Parameters**lpszProjectFile**

Pointer to the name of the project file to be opened, including path and extension.

The release name (e.g. WinCC 50_Project_Odk) implicitly generated by WinCC cannot be used as a path because it exists only as long as this project is open.

The name of the project file is determined with `DMLenumOpenedProjects` or with `DMGetRuntimeProject` in RT.

lpdmError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Project opened.

FALSE

Error.

Required files

dmclient.h

dmclient.lib

dmclient.dll

3.2.4.7 DMOpenProjectPlus (RT Professional)

Use

Selection of a project file via the standard selection dialog of the system. If the project is not yet open in WinCC, the information required for the project is loaded.

When the execution is successful, the calling application should remember the name of the project file entered in `lpszProjectFile`, since it can be used to determine all other information about the project.

The name of the project file can later be determined with `DMLenumOpenedProjects`.

Declaration

```
BOOL DMOpenProjectPlus (
    HWND          hwndParent,
    LPSTR          lpszProjectFile,
    DWORD          dwBufSize,
    LPCMN_ERROR    lpdmError );
```

Parameters

hwndParent

Handle to the window that is used as the parent window for the dialog.

lpszProjectFile

Pointer to the buffer for storing the name of the project file, including path and extension. The buffer should be at least `_MAX_PATH` characters in size.

`lpszProjectFile` must be an empty string or a valid path of an existing project when the function is called.

If `lpszProjectFile` is an empty string, the default selection dialog of the system (see above) is called.

If `lpszProjectFile` is not empty, WinCC interprets the string contained in `lpszProjectFile` as the project path and attempts to open it without opening the selection dialog.

The release name (e.g. WinCC 50_Project_Odk) implicitly generated by WinCC cannot be used as a project path because it exists only as long as this project is open.

dwBufSize

Size of the given buffer in characters.

lpdmError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Close the dialog with "OK".

FALSE

Error or close dialog with "CANCEL"

Comments

The process of opening a project may take several minutes.

Error messages

DM_E_CANCEL	User has selected "Cancel" in a dialog
DM_E_FILE	Error with file operation
DM_E_UPDATE	Updating project

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

Open project by means of dialog (Page 1861) "DM01.cpp"

OnTestOpenProject (Page 1841) "TESTCDoc.cpp"

See also

Open project by means of dialog (Page 1861)

OnTestOpenProject (Page 1841)

3.2.5 Data transport channels (RT Professional)

3.2.5.1 DMClearBlockQueue (RT Professional)

Use

Deletes all entries in the notification queue of the application.

Declaration

```
BOOL DMClearBlockQueue (  
    LPCMN_ERROR    lpdmError );
```

Parameters

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Notification queue is deleted.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h
dmclient.lib
dmclient.dll

3.2.5.2 DMEEnumDataServices (RT Professional)

Use

DMEEnumDataServices calls the passed callback function for each service already installed, as long as the callback function returns TRUE.

Declaration

```

BOOL DMEEnumDataServices (
    LPDWORD                lpdwItems,
    DM_ENUM_DATA_SERVICE_PROC lpfnEnum,
    LPVOID                 lpvUser,
    LPCMN_ERROR             lpdmError);

```

Parameters

lpdwItems

Pointer to a double word of the application, which contains the number of listed services after the call.

lpfnEnum

Pointer to your callback function that is called for every installed service.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Services listed.

FALSE

Error.

Comment

Pointers passed to the callback are only valid within this, since the system releases all allocated memory areas once the function returns. If an application requires access to the data beyond this, it must copy it accordingly.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_DATA_SERVICE_PROC (Page 1666)	List data transport channels (callback)
DMInstallDataService (Page 1670)	Install data transport channel

See also

DM_ENUM_DATA_SERVICE_PROC (Page 1666)

DMInstallDataService (Page 1670)

3.2.5.3 DM_ENUM_DATA_SERVICE_PROC (RT Professional)**Description**

In order to evaluate the installed data transport channels listed by the system, you must provide a DM_ENUM_DATA_SERVICE_PROC type callback function.

Declaration

```

BOOL ( * DM_ENUM_DATA_SERVICE_PROC) (
    LPCSTR                pszService,
    DM_DATA_SERVICE_PROC  pfnService,
    LPVOID                 lpvUser );

```

Parameters

pszService

Pointer to the logical name of the data transport channel.

pfnService

Pointer to your DM_DATA_SERVICE_PROC (Page 1668) type callback function, through which the additional data of a service are made available.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

dmclient.h

Related functions

DMEnumDataServices (Page 1663)	List data transport channels

See also

DMEEnumDataServices (Page 1663)

DM_DATA_SERVICE_PROC (Page 1668)

3.2.5.4 DM_DATA_SERVICE_PROC (RT Professional)

Description

To enable your application to receive data, you must have a data transport channel installed via DMInstallDataService and provide a DM_DATA_SERVICE_PROC type callback function.

You also need a callback function of this type if you want to evaluate the transport channels (services) listed by the system.

Declaration

```
BOOL ( * DM_DATA_SERVICE_PROC) (
    LPDM_DATA_SERVICE    lpds,
    LPVOID                lpvUser);
```

Parameters

lpds

Pointer to a DM_DATA_SERVICE (Page 1566) type structure with the information on an installed data transport channels (service).

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

Depending on the use of the callback function.

FALSE

Error

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

Required files

dmclient.h

Related functions

DMInstallDataService (Page 1670)	Install data transport channel
DM_ENUM_DATA_SERVICE_PROC (Page 1664)	List data transport channels (callback)

See also

DM_ENUM_DATA_SERVICE_PROC (Page 1664)

DM_DATA_SERVICE (Page 1566)

DMInstallDataService (Page 1670)

3.2.5.5 DMGetNumPendingBlocks (RT Professional)**Use**

This function is used by an application to determine the entries in its queue.

Declaration

```

BOOL DMGetNumPendingBlocks (
    LONG          *plEntries,
    LPCMN_ERROR   lpdmError);

```

Parameters

plEntries

After successful execution, the function contains the number of data packets in the queue of the application.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Number of entries in queue determined.

FALSE

Error.

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

3.2.5.6 DMInstallDataService (RT Professional)

Use

Installs a transport channel (service) for data transfer between applications. The function signals the readiness of the application to receive data sent by any other applications under the specified service name. If a service is already installed under the given name, the callback function installed for this is replaced by whatever is passed. If lpfnService == NULL, the service installed lpszService is removed.

Declaration

```
BOOL DMInstallDataService (
    LPCSTR          lpszService,
    DM_DATA_SERVICE_PROC lpfnService,
    LPVOID          lpvUser,
    LPCMN_ERROR      lpdmError);
```

Parameters

IpszService

Pointer to the name of the service. The transport channel is identified by this name, which have have freely assigned.

lpfnService

Pointer to a callback function that is called when data is sent to the application through this service.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function, but is made available again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Data transport channel installed.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DM_DATA_SERVICE_PROC (Page 1666)	List data transport channels (callback), install data transport channel (callback)

See also

DMLenumDataServices (Page 1663)

DM_DATA_SERVICE_PROC (Page 1666)

3.2.5.7 DMSendApplicationData (RT Professional)

Use

The data passed in `lpdmSendData` are sent to all applications that meet the conditions specified in the `DM_SEND_DATA_STRUCT` structure and have installed the service described under `szService` using `DMInstallDataService`.

Declaration

```
BOOL DMSendApplicationData (  
    LPDM_SEND_DATA_STRUCT    lpdmSendData,  
    LPCMN_ERROR               lpdmError);
```

Parameters

lpdmSendData

Pointer to the `DM_SEND_DATA_STRUCT` (Page 1572) structure.

lpdmError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Data sent to application.

FALSE

Error.

Comment

The function is only implemented locally.
The members in the `DM_SEND_DATA_STRUCT` structure for remote access to the "TargetMachine" are reserved for future development.
The `dwTargetMachineFlags` can only be specified with `DM_SD_LOCAL`, any other specification leads to errors and `dwTargetMachines` must be 0L.

Required files

dmclient.h
dmclient.lib
dmclient.dll

See also

DM_SEND_DATA_STRUCT (Page 1572)

3.2.5.8 DMSetBlockQueueSize (RT Professional)**Use**

This function specifies the maximum number of messages that can be held in the notification queue of the application.

Declaration

```
BOOL DMSetBlockQueueSize (  
    LONG          nCount,  
    LPCMN_ERROR   lpdmError);
```

Parameters**nCount**

Maximum number of messages.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Size of the queue set.

FALSE

Error.

Comment

The default value for 32-bit platforms is 1000 entries, which is usually sufficient.

Each message occupies address space of the application. Think carefully before increasing the number of messages as this will slow processing by the application.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h
dmclient.lib
dmclient.dll

3.2.6 Functions for processing tags (RT Professional)

3.2.6.1 DMEnumVarData (RT Professional)

Use

The function is used to determine information about objects that are associated with a tag, for example, associated tag group, channel unit, etc.

Declaration

```
BOOL DMEnumVarData (
    LPCSTR          lpszProjectFile,
    LPDM_VARKEY     lpdmVarKey,
    DWORD           dwItems,
    DM_ENUM_VARIABLE_PROC lpfnEnum,
    LPVOID          lpvUser,
    LPCMN_ERROR     lpdmError);
```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.
The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

lpdmVarKey

Pointer to the beginning of a field for DM_VARKEY (Page 1590) type structures, through which the tags to be listed are specified. As of V5.0 SP2, if no array is specified and supplied with NULL, all tags are enumerated.

If an incorrect name or a false ID is given for DM_VARKEY, no errors are displayed because some can be valid and some incorrect in a list. In order for the user to identify an incorrect DM_VARKEY, a DM_VARIABLE_DATA (Page 1586) type structure is passed for this in the callback, which is filled with 0.

dwItems

Number of tags whose data are listed (= number of tag specifications in lpdmVarKey). If the number is set to 0, all tags are enumerated.

lpfnEnum

Pointer to your callback, which receives the information about a tag.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Information on the tag listed.

FALSE

Error

Comment

There is an advanced function, DMEnumVarData4.

Note

Existing implementations of a multi-stage enumeration of DMEnumVariables with the call of DMEnumVarDataX in the callback for each individual element should be replaced by a single call of DMEnumVarData4. Any necessary filtering should be performed here in the callback itself. The change considerably improves performance.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DM_ENUM_VARIABLE_PROC (Page 1676)	List information about a tag (callback)

See also

DM_VARKEY (Page 1590)
DM_VARIABLE_DATA (Page 1586)
DM_ENUM_VARIABLE_PROC (Page 1676)

3.2.6.2 DM_ENUM_VARIABLE_PROC (RT Professional)

Description

In order to evaluate information about a tag listed by the system, you must provide a DM_ENUM_VARIABLE_PROC type callback function.

Declaration

```
BOOL ( * DM_ENUM_VARIABLE_PROC) (  
    LPDM_VARKEY          lpdmVarKey,  
    LPDM_VARIABLE_DATA   lpdmVarData,  
    LPVOID                lpvUser
```

Parameters

lpdmVarKey

Pointer to a DM_VARKEY (Page 1590) type structure for specification of the tag to be listed.

lpdmVarData

Pointer to a DM_VARIABLE_DATA (Page 1586) type structure with information about a tag.

When the structure completely filled with 0, an incorrect name or incorrect ID was given in DM_VARKEY. Can be queried via `lpdmVarData->dmTypeRef->dwType`, for example.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: `GetMessage`
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

dmclient.h

Related functions

DMEnumVarData (Page 1672)	List information about a tag

See also

DM_VARKEY (Page 1590)

DM_VARIABLE_DATA (Page 1586)

DMEnumVarData (Page 1672)

3.2.6.3 DMEnumVarData4 (RT Professional)

Use

The function is used to determine information about objects that are associated with a tag, for example, associated tag group, channel unit, etc.

If differs from `DMEnumVarData` due to the additional specification of scaling information.

Declaration

```
BOOL DMEnumVarData4 (
    LPCSTR                lpszProjectFile,
    LPDM_VARKEY           lpdmVarKey,
    DWORD                 dwItems,
    DM_ENUM_VARIABLE_PROC4 lpfnEnum,
    LPVOID                 lpvUser,
    LPCMN_ERROR            lpdmError);
```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

lpdmVarKey

Pointer to the beginning of a field for DM_VARKEY (Page 1590) type structures, through which the tags to be listed are specified. If the parameter is set to NULL, all tags are enumerated.

If an incorrect name or a false ID is given for DM_VARKEY, no errors are displayed because some can be valid and some incorrect in a list. In order for the user to identify an incorrect DM_VARKEY, a DM_VARIABLE_DATA4 (Page 1588) type structure is passed for this in the callback, which is filled with 0.

dwItems

Number of tags whose data should be listed (= number of tag specifications in lpdmVarKey). If the number is set to 0, all tags are enumerated.

lpfnEnum

Pointer to your callback, which receives the information about a tag.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Information on the tag listed.

FALSE

Error.

Comment**Note**

Existing implementations of a multi-stage enumeration of `DMEnumVariables` with the call of `DMEnumVarDataX` in the callback for each individual element should be replaced by a single call of `DMEnumVarData4`. Any necessary filtering should be performed here in the callback itself. The change considerably improves performance.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DM_ENUM_VARIABLE_PROC4 (Page 1680)	List information about a tag (callback)

Examples

Enum Data of Tags (Page 1826) "DM01.cpp"

See also

DM_VARKEY (Page 1590)
DM_VARIABLE_DATA4 (Page 1588)
DM_ENUM_VARIABLE_PROC4 (Page 1680)
Enum Data of Tags (Page 1826)

3.2.6.4 DM_ENUM_VARIABLE_PROC4 (RT Professional)

Description

In order to evaluate information about a tag listed by the system, you must provide a DM_ENUM_VARIABLE_PROC4 type callback function. This callback function differs from DM_ENUM_VARIABLE_PROC due to the additional specification of scaling information.

Declaration

```
BOOL ( * DM_ENUM_VARIABLE_PROC4) (
    LPDM_VARKEY          lpdmVarKey,
    LPDM_VARIABLE_DATA4  lpdmVarData,
    LPVOID               lpvUser
```

Parameters

lpdmVarKey

Pointer to a DM_VARKEY (Page 1590) type structure for specification of the tag to be listed.

lpdmVarData

Pointer to a DM_VARIABLE_DATA4 (Page 1588) type structure with information about a tag.

When the structure completely filled with 0, an incorrect name or incorrect ID was given in DM_VARKEY. Can be queried via lpdmVarData->dmTypeRef->dwType, for example.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

dmclient.h

Related functions

DMEnumVarData4 (Page 1675)	List information about a tag

Examples

Enum Data of Tags (Page 1826) "DM01.cpp"

See also

DMEnumVarData4 (Page 1675)

DM_VARKEY (Page 1590)

DM_VARIABLE_DATA4 (Page 1588)

Enum Data of Tags (Page 1826)

3.2.6.5 DMEnumVarGrpData (RT Professional)**Use**

The function returns information about the tag groups via the DM_ENUM_VARGRP_PROC callback function, for example, name, creator ID and number of tags in the group.

Declaration

```

BOOL DMEnumVarGrpData (
    LPSTR          lpszProjectFile,
    LPDM_VARGRPKEY lpdmVarGrpKey,
    DWORD          dwItems,
    DM_ENUM_VARGRP_PROC lpfnEnum,
    LPVOID          lpvUser,
    LPCMN_ERROR     lpdmError);

```

Parameters**lpszProjectFile**

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

lpdmVarGrpKey

Pointer to the first of the DM_VARGRPKEY (Page 1585) type structures, through which the tag group is specified.

dwItems

The number of tags whose data are listed.

lpfnEnum

Pointer to your callback, which receives the information about a tag group.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Information on the tag group listed.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_VARGRP_PROC (Page 1685)	List information about a tag group (callback)

Examples

OnTestEnumGroupsAll (Page 1835) "TESTCDoc.cpp"

See also

DM_VARGRPKEY (Page 1585)

DM_ENUM_VARGRP_PROC (Page 1685)

OnTestEnumGroupsAll (Page 1835)

3.2.6.6 DMLenumVarGrpDataExStr (RT Professional)**Declaration**

```

BOOL DMLenumVarGrpDataExStr (
    LPCTSTR                lpszProjectFile,
    DWORD                  dwFlags,
    LPVARIANT               lpvdmVarGrpKey,
    LPDWORD                 lpdwVarGrpCount,
    DM_ENUM_VARGRP_PROC_EXSTR lpfnEnum,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);

```

Description

This function returns tag group information by means of the DM_ENUM_VARGRP_PROC_EXSTR callback function, for example, name, ID and number of tags in the group.

Parameters**lpszProjectFile**

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMLenumOpenedProjects, or with DMGetRuntimeProject in Runtime.

dwFlags

Reserved for future use. Is to be filled with 0L.

lpvdmVarGrpKey

Pointer to the first structure of the "DM_VARGRPKEY" type specifying the tag groups.

Pointer to a VARIANT for this tag group list. You must create the list as VT_ARRAY | VT_VARIANT as different data types such as VT_I4, VT_BSTR and under certain conditions also VT_LPSTR can be different for each tag. Alternatively, you create a single VARIANT if only one key is specified.

VT_EMPTY or NULL pointers are used to enumerate all groups.

lpdwVarGrpCount

Pointer to DWORD returning the number of tag groups.

lpfnCallback = NULL is used to initially determine how much memory is required for the tag groups.

lpfnEnum

Pointer to your callback function which receives the tag group information.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the callback function.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tag group information enumerated

FALSE

Error

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_VARGRP_PROC_EXSTR	Enumerate tag group information (callback)

3.2.6.7 DM_ENUM_VARGRP_PROC (RT Professional)

Description

In order to evaluate information about a tag group listed by the system, you must provide a DM_ENUM_VARGRP_PROC type callback function.

Declaration

```
BOOL ( * DM_ENUM_VARGRP_PROC) (
    LPDM_VARGRP_DATA    lpdmVarGrpData,
    LPVOID               lpvUser );
```

Parameters

lpdmVarGrpData

Pointer to a DM_VARGRP_DATA (Page 1584) type structure with information about the tag group.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

dmclient.h

Related functions

DMEnumVarGrpData (Page 1679)	List information about a tag group

See also

DMEnumVarGrpData (Page 1679)

DM_VARGRP_DATA (Page 1584)

3.2.6.8 DMEnumVariables (RT Professional)**Use**

The function lists the names of all tags that match a certain selection criterion. The selection criterion is determined by the DM_VARFILTER structure.

Declaration

```

BOOL DMEnumVariables (
    LPCSTR          lpszProjectFile,
    LPDM_VARFILTER  lpdmVarFilter,
    DM_ENUM_VAR_PROC lpfnEnum,
    LPVOID          lpvUser,
    LPCMN_ERROR      lpdmError);

```

Parameters**lpszProjectFile**

Pointer to the name of the project file, including path and extension.
The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

lpdmVarFilter

Pointer to the DM_VARFILTER (Page 1582) structure in which the selection criterion is specified.

lpfnEnum

Pointer to your callback function, which receives the tag names.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tag names listed.

FALSE

Error.

Comment**Note**

Existing implementations of a multi-stage enumeration of DMEnumVariables with the call of DMEnumVarDataX in the callback for each individual element should be replaced by a single call of DMEnumVarData4. Any necessary filtering should be performed here in the callback itself. The change considerably improves performance.

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_VAR_PROC (Page 1688)	List tag names (callback)

Examples

OnTestEnumVariables (Page 1836) "TESTCDoc.cpp"

See also

DM_VARFILTER (Page 1582)

DM_ENUM_VAR_PROC (Page 1688)

OnTestEnumVariables (Page 1836)

3.2.6.9 DM_ENUM_VAR_PROC (RT Professional)

Description

In order to evaluate the names of tags listed by the system, you must provide a DM_ENUM_VAR_PROC type callback function.

Declaration

```
BOOL ( * DM_ENUM_VAR_PROC) (  
    LPDM_VARKEY    lpdmVarKey,  
    LPVOID         lpvUser
```

Parameters

lpdmVarKey

Pointer to a DM_VARKEY (Page 1590) type structure with the name of a tag.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

Additional callback calls are desirable.

FALSE

The callback series should be cancelled.

Comment

NOTICE

You may not call any other DMClient function in this callback. A blockade may otherwise occur.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

dmclient.h

Related functions

DMEnumVariables (Page 1684)	List tag names

See also

DM_VARKEY (Page 1590)

DMEnumVariables (Page 1684)

3.2.6.10 DMGetValue (RT Professional)**Use**

Reads the value of one or more tags from the process image of the data manager. The value that is read out is the one existing at the time of the most recent update.

The more advanced DMGetValueEx function, additionally returns the quality code in the DM_VAR_UPDATE_STRUCTEX structure.

Declaration

```

BOOL DMGetValue (
    LPDM_VARKEY          lpdmVarKey,
    DWORD                dwItems,
    LPDM_VAR_UPDATE_STRUCT lpdmvus,
    LPCMN_ERROR          lpdmError);

```

Parameters**lpdmVarKey**

Pointer to the DM_VARKEY (Page 1590) structure that identifies the tag values to be read.

dwItems

Number of passed structures (corresponds to the number of tag values to be read).

lpdmvus

Pointer to the first DM_VAR_UPDATE_STRUCT (Page 1577) structure that contains the tag values once the function returns.

lpdmError

Pointer to the first of dwItems error structure with the CMN_ERROR type. When an error occurs during the writing of a tag, the system writes the error information into the appropriate structure. Therefore, remember to reserve space for these structures.

Return value**TRUE**

Values determined.

FALSE

Error.

Comment

To provide an application the possibility of separate cycle management, a ZERO pointer is specified as the callback function for DMStartVarUpdate. Then, the data manager updates the tags according to the required cycle. However, the application remains responsible for the timing of the readout obtained from the process image.

Since update requests for a tag can be present from several applications in different cycles, the data manager always updates its process image based on the shortest requested cycle.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMGetValueWait (Page 1702)	Get tag values with update
DMSetValue (Page 1732)	Change tag values

Examples

Read tag (Page 1862) "DM02.cpp"

OnTestVariablenGetValue (Page 1846) "TESTCDoc.cpp"

See also

DM_VARKEY (Page 1590)
 DM_VAR_UPDATE_STRUCT (Page 1577)
 DMGetValueWait (Page 1702)
 DMSetValue (Page 1732)
 Read tag (Page 1862)
 OnTestVariablenGetvalue (Page 1846)

3.2.6.11 DMGetValueEx (RT Professional)**Use**

Reads the value of one or more tags from the process image of the data manager. The value that is read out is the one existing at the time of the most recent update.

In contrast to DMGetValue, the quality code is also returned in the DM_VAR_UPDATE_STRUCTEX structure.

Declaration

```

BOOL DMGetValueEx (
    LPDM_VARKEY          lpdmVarKey,
    DWORD                dwItems,
    LPDM_VAR_UPDATE_STRUCTEX lpdmvus,
    LPCMN_ERROR          lpdmError);

```

Parameters**lpdmVarKey**

Pointer to the DM_VARKEY (Page 1590) structure that identifies the tag values to be read.

dwItems

Number of passed structures (corresponds to the number of tag values to be read).

lpdmvus

Pointer to the first DM_VAR_UPDATE_STRUCTEX (Page 1579) structure that contains the tag values once the function returns.

lpdmError

Pointer to the first of dwItems error structure with the CMN_ERROR type. When an error occurs during the writing of a tag, the system writes the error information into the appropriate structure. Therefore, remember to reserve space for these structures.

Return value**TRUE**

Values determined.

FALSE

Error.

Comment

To provide an application the possibility of separate cycle management, a NULL pointer can be specified as the callback function for DMStartVarUpdate. Then, the data manager updates the tags according to the required cycle. However, the application remains responsible for the timing of the readout obtained from the process image.

Since update requests for a tag can be present from several applications in different cycles, the data manager always updates its process image based on the shortest requested cycle.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMGetValueWait (Page 1702)	Get tag values with update
DMSetValue (Page 1732)	Change tag values

See also

DM_NOTIFY_VARIABLEEX_PROC (Page 1817)

DM_VAR_UPDATE_STRUCTEX (Page 1579)

DM_VARKEY (Page 1590)

DMGetValueWait (Page 1702)

DMSetValue (Page 1732)

DMGetValueWaitEx (Page 1704)

3.2.6.12 DMGetValueExStr (RT Professional)

Declaration

```

BOOL DMGetValueExStr (
    DWORD                dwFlags,
    LPVARIANT            lpdmVarKey,
    LPDM_VAR_UPDATE_STRUCT_EXSTR lpdmvus,
    DWORD                dwdmvusCount,
    LPCMN_ERROR          lpdmError);

```

Description

Reads the value of one or more tags from the process image of the data manager. The value that is read out is the one existing at the time of the most recent update.

In contrast to DMGetValueEx, the DM_VARKEY structure is no longer used; as a result no length constraint applies with tag names.

Parameters

dmFlags

If the tag name needs to be returned in the VT_LPSTR format in the VARIANT of the DM_VAR_UPDATE_STRUCT_EXSTR structure of DM_NOTIFY_VARIABLE_PROC_EXSTR, the DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR flag can be specified here.

lpdmVarKey

Pointer to a VARIANT for the tag list. You must create the list as VT_ARRAY | VT_VARIANT, as various data types such as VT_I4, VT_BSTR and, in some cases, also VT_LPSTR can be present. This applies also to a single key.

lpdmvus

Pointer to the first of the DM_VAR_UPDATE_STRUCT_EXSTR structures which contains the values of the tags after the return of the function.

dwdmvusCount

Number of DM_VAR_UPDATE_STRUCT_EXSTR structures passed (identical with the number of tag values to be read). The number must conform with the size of the variant array in lpdmVarKey.

lpdmError

Pointer to the first "dwdmvusCount" error structure of the CMN_ERROR type. If a tag writing error occurs, the system writes the error information into the corresponding structure. Therefore, remember to reserve space for these structures.

Return value**TRUE**

Values have been determined

FALSE

Error

Comment

In order to e.g. enable an application to manage its own cycle administration, a NULL pointer can be specified for DMStartVarUpdate as a callback function. Then, the data manager updates the tags according to the required cycle. However, the application remains responsible for the timing of the readout obtained from the process image.

As update requests for a tag can be made by multiple applications operating with different cycles, the data manager always updates its process image based on the shortest cycle time.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

These errors are only returned in the first error structure.

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMGetValueWaitExStr	Get tag values with update
DMSetValueExStr	Change tag values

Examples**Script sample "Button DMGetValueExStr/DMSetValueExStr|DMGetValueWaitExStr/DMSetValueWaitExStr":**

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("kernel32.dll")
#define CP_ACP 0
```

```

int MultiByteToWideChar(UINT CodePage, DWORD dwFlags, LPCSTR
lpMultiByteStr, int cbMultiByte, LPWSTR lpWideCharStr, int
cchWideChar);
int WideCharToMultiByte(UINT CodePage, DWORD dwFlags, LPCWSTR
lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr, int
cbMultiByte, LPCSTR lpDefaultChar, LPBOOL lpUsedDefaultChar);
#pragma code()

#pragma code ("OleAut32.dll")
//#include "OleAuto.h"
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound, unsigned
int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR* rgIndices,
void HUGE* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayUnlock(SAFEARRAY FAR* psa);
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
typedef WCHAR OLECHAR;
BSTR SysAllocString(OLECHAR FAR* sz);
HRESULT SysFreeString(BSTR bstr);
#pragma code()

extern BOOL DM_NotifyVariableProcExStr_GetValueWait(DWORD dwTAID,
LPDM_VAR_UPDATE_STRUCT_EXSTR lpdmvus, DWORD dwItems, LPVOID lpvUser);
extern BOOL DM_CompletionProc(DWORD dwTAID, LPDWORD lpdwVarState,
DWORD dwItems, LPVOID lpvUser);

VARIANT vVarKey;
VARIANT* pvElem;
VARIANT vdmValue;
SAFEARRAY* parrayKeys;
SAFEARRAY* parrayValues;
DWORD dwVal[4];
DWORD dwState[4];
DWORD dwMerk;
CMN_ERROR err;
CMN_ERROR errArray[4];
DM_VAR_UPDATE_STRUCT_EXSTR dmvus[4];
HRESULT hr;
long lInx;
BOOL bRet;
DWORD dwFlags;
DWORD* pdwVarState;
DWORD dwTAID;

memset(&err, 0, sizeof(err));
memset(dmvus, 0, sizeof(dmvus));
dwVal[0] = 1;
dwVal[1] = 2;
dwVal[2] = 3;
dwVal[3] = 4;

```

3.2 Data management functions (RT Professional)

```

dwMerk = 0L;
memset(dwState, 0, sizeof(dwState));
pvElem = NULL;
parrayKeys = NULL;
parrayValues = NULL;
hr = 0L; //S_OK
lInx = 0L;
bRet = FALSE;
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
pdwVarState = dwState;
dwTAID = 0L;

printf("\r\n\r\n##### enter Test with DMGetValue(Wait)ExStr/
DMSetValue(Wait)ExStr #####");

VariantInit(&vVarKey);
parrayKeys = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vVarKey.vt = VT_ARRAY | VT_VARIANT;
vVarKey.u.parray = parrayKeys;

SafeArrayLock(parrayKeys);
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_1";
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_2";
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_3";
lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_4";
SafeArrayUnlock(parrayKeys);

printf("\r\nncall DMGetValueExStr:");
bRet = DMGetValueExStr(dwFlags, &vVarKey, dmvus, 4L, &err);

if (bRet == FALSE)
{
    printf("\r\n  error DMGetValueExStr, err=%ld,%ld,%ld,%ld,%ld,
[%s]",
           err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    int i = 0;

```

```

    for (i = 0; i < 4; i++)
    {
        hr = VariantChangeType((VARIANTARG*)&(dmvus[i].vdmValue),
(VARIANTARG*)&(dmvus[i].vdmValue), 0, VT_I4);
        if (hr)
        {
            printf("\r\n    error VariantChangeType[%d] hr=%08lx",
i, hr);
        }
        else
        {
            if (dmvus[i].vdmValue.vt == VT_I4)
            {
                dwVal[i] = dmvus[i].vdmValue.u.lVal;
                printf("\r\n    Var[%d]:[%s] = %ld",
                    i, dmvus[i].vdmVarKey.u.pbVal,
dmvus[i].vdmValue.u.lVal);
            }
            else
            {
                printf("\r\n    wrong datatype %ld, VT_I4 expected after
VariantChange",
                    dmvus[i].vdmValue.vt == VT_I4);
            }
        }
    }

    //clear the internal allocated Data, that no memoryleaks
    //the vdmValues no need to clear here, while no PROPVARIANT here
given (double used)
    //only the dmVarkey of dmvus have to free here
    for (i = 0; i < 4; i++)
    {
        free(dmvus[i].vdmVarKey.u.pbVal);
        dmvus[i].vdmVarKey.u.pbVal = NULL;
        dmvus[i].vdmVarKey.vt = VT_EMPTY;
    }

    // change the values before set again
    dwMerk = dwVal[0];
    dwVal[0] = dwVal[3];
    dwVal[3] = dwMerk;
    dwMerk = dwVal[1];
    dwVal[1] = dwVal[2];
    dwVal[2] = dwMerk;

    VariantInit(&vdmValue);
    parrayValues = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
    vdmValue.vt = VT_ARRAY | VT_VARIANT;
    vdmValue.u.parray = parrayValues;

```

3.2 Data management functions (RT Professional)

```

SafeArrayLock(parrayValues);
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[0];
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[1];
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[2];
lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[3];
SafeArrayUnlock(parrayValues);

printf("\r\n call DMSetValueExStr:");
memset(&err, 0, sizeof(err));
bRet = DMSetValueExStr(&vVarKey, &vdmValue, pdwVarState, &err);

if (bRet == FALSE)
{
    printf("\r\n error DMSetValueExStr, err=%ld,%ld,%ld,%ld,%ld,
[%s]",
           err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    int i = 0;
    for (i = 0; i < 4; i++)
    {
        printf("\r\n dwVarState[%d] = %ld", i, dwState[i]);
    }
}

// clear the values before read back again with DMGetValueWait
(timeout 2sec.)
dwVal[0] = 0;
dwVal[1] = 0;
dwVal[2] = 0;
dwVal[3] = 0;

printf("\r\n call DMGetValueWaitExStr:");
memset(&err, 0, sizeof(err));
bRet = DMGetValueWaitExStr(&dwTAID, dwFlags, &vVarKey, TRUE, 2000,
DM_NotifyVariableProcExStr_GetValueWait, dwVal, &err);
if (bRet == FALSE)
{

```



```

        printf("\r\n  error DMGetValueWaitExStr, err=%ld,%ld,%ld,%ld,%ld,
[%s]",
            err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
    }
else
{
    int i = 0;
    for (i = 0; i < 4; i++)
    {
        printf("\r\n    dwValue[%d] = %ld", i, dwVal[i]);
    }
}

//increment values for identify execute of following DMSetValueWait
dwVal[0] = dwVal[0] + 1;
if (dwVal[0] >= 100)
{
    dwVal[0] = dwVal[0] - 100;
}
dwVal[1] = dwVal[1] + 1;
if (dwVal[1] >= 100)
{
    dwVal[1] = dwVal[1] - 100;
}
dwVal[2] = dwVal[2] + 1;
if (dwVal[2] >= 100)
{
    dwVal[2] = dwVal[2] - 100;
}
dwVal[3] = dwVal[3] + 1;
if (dwVal[3] >= 100)
{
    dwVal[3] = dwVal[3] - 100;
}
VariantClear(&vdmValue);
parrayValues = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vdmValue.vt = VT_ARRAY | VT_VARIANT;
vdmValue.u.parray = parrayValues;
SafeArrayLock(parrayValues);
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[0];
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[1];
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[2];

```

3.2 Data management functions (RT Professional)

```

lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[3];
SafeArrayUnlock(parrayValues);

printf("\r\n call DMSetValueWaitExStr:");
memset(errArray, 0, sizeof(errArray));
bRet = DMSetValueWaitExStr(&dwTAID, &vVarKey, 4, &vdmValue,
pdwVarState, 2000, DM_CompletionProc, NULL, errArray);
if (bRet == FALSE)
{
    int i = 0;
    for(i=0; i < 4; i++)
    {
        printf("\r\n      error DMSetValueWaitExStr, errArray[%d]=%ld,
%ld,%ld,%ld,%ld,[%s]",
            i, errArray[i].dwError1, errArray[i].dwError2,
errArray[i].dwError3,
            errArray[i].dwError4, errArray[i].dwError5,
errArray[i].szErrorText);
    }
}
else
{
}

printf("\r\n##### exit Test with DMGetValue(Wait)ExStr/
DMSetValue(Wait)ExStr #####\r\n");

}

```

Script example "Projekt-Funktion DM_NotifyVariableProcExStr_GetValueWait":

```

#pragma code ("OleAut32.dll")
// #include "OleAuto.h"
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
#pragma code()

BOOL DM_NotifyVariableProcExStr_GetValueWait(DWORD dwTAID,
LPDM_VAR_UPDATE_STRUCT_EXSTR lpdmvus, DWORD dwItems, LPVOID lpvUser)
{
    BOOL bRet = FALSE;
    int i = 0;
    DWORD* pdwVal = NULL;

    pdwVal = (DWORD*)lpvUser;
    // the DMGetValueWait used PROPVARIANT and the lpvUser is ptr to
    OutDataArray (DWORDS)
    printf("\r\n*** DM_NotifyVariableProcExStr_GetValueWait entry ***");

    for (i = 0; i < dwItems; i++)

```

```

{
    VariantChangeType((VARIANTARG*)&(lpdmvus[i].vdmValue),
(VARIANTARG*)&(lpdmvus[i].vdmValue), 0, VT_I4);
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal, "dwVal_1"))
    {
        pdwVal[0] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_1=%ld to dwVal[0]", pdwVal[0]);
        bRet = TRUE;
    }
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal, "dwVal_2"))
    {
        pdwVal[1] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_2=%ld to dwVal[1]", pdwVal[1]);
        bRet = TRUE;
    }
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal, "dwVal_3"))
    {
        pdwVal[2] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_3=%ld to dwVal[2]", pdwVal[2]);
        bRet = TRUE;
    }
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal, "dwVal_4"))
    {
        pdwVal[3] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_4=%ld to dwVal[3]", pdwVal[3]);
        bRet = TRUE;
    }
}

printf("\r\n*** DM_NotifyVariableProcExStr_GetValueWait exit ***");
return bRet;
}

Project-Function DM_CompletionProc:

BOOL DM_CompletionProc(DWORD dwTAID, LPDWORD lpdmVarState, DWORD
dwItems, LPVOID lpvUser)
{
    BOOL bRet = FALSE;
    int i = 0;

    printf("\r\n*** DM_CompletionProc entry ***");

    for (i = 0; i < dwItems; i++)
    {
        printf("\r\n  VarState[%d] = %ld", i, lpdmVarState[i]);
    }

    printf("\r\n*** DM_CompletionProc exit ***");

    return bRet;
}

```

3.2.6.13 DMGetValueWait (RT Professional)

Use

Reads the value of one or more tags from the process image of the data manager. The value that is read out is the one existing at the time of the most recent update.

In contrast to DMGetValue, this function forces a value-based update of the tag values.

The more advanced DMGetValueWaitEx function additionally returns the quality code in the DM_NOTIFY_VARIABLE_EX_PROC callback in the DM_VAR_UPDATE_STRUCTEX structure.

Declaration

```

BOOL DMGetValueWait (
    LPDWORD                pdwTAID,
    LPDM_VARKEY            lpdmVarKey,
    DWORD                  dwItems,
    BOOL                    fWaitForCompletion,
    DWORD                  dwTimeout,
    DM_NOTIFY_VARIABLE_PROC lpfnVariable,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);

```

Parameters

pdwTAID

Pointer to a tag that contains the transaction ID assigned by the data manager after a successful call of the function.

lpdmVarKey

Pointer to the DM_VARKEY (Page 1590) structure that identifies the tag values to be read.

dwItems

Number of passed structures (corresponds to the number of tag values to be read).

fWaitForCompletion

If the fWaitForCompletion flag is set, the passed callback function is called only called when either all the requested tags are updated or the specified timeout period is exceeded.

If fWaitForCompletion is not set, the passed callback function is called immediately. The time characteristics correspond to the function DMGetValue.

dwTimeout

Maximum waiting time of the application in ms. If not all tag values have been updated when the waiting time expires, the callback function with corresponding error codes is called.

If the fWaitForCompletion == FALSE flag is set, the passed value is not evaluated.

lpfnVariable

Pointer to your DM_NOTIFY_VARIABLE_PROC callback function, which is called either after the update of all requested tags or after the waiting period.

If fWaitForCompletion == FALSE, the callback function is called immediately with the values currently set in the process image of the data manager.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Values determined.

FALSE

Error.

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_NOTIFY_VARIABLE_PROC (Page 1806)	Get tag values with update (callback)
DMGetValue (Page 1687)	Get tag values
DMGetValueWaitEx (Page 1704)	Get tag values with update
AUTOHOTSPOT	Change tag values with notification

Examples

OnTestVariablenGetvaluawait (Page 1849) "TESTCDoc.cpp"

See also

DM_NOTIFY_VARIABLE_PROC (Page 1806)

DM_VARKEY (Page 1590)

DMGetValue (Page 1687)

DMGetValueEx (Page 1689)

DMSetValueWait (Page 1742)

OnTestVariablenGetvaluawait (Page 1849)

DMGetValueWaitEx (Page 1704)

3.2.6.14 DMGetValueWaitEx (RT Professional)**Use**

Reads the value of one or more tags from the process image of the data manager. The value that is read out is the one existing at the time of the most recent update.

In contrast to DMGetValueEx, this function forces a value-based update of the tag values.

Declaration

```

BOOL DMGetValueWaitEx (
    LPDWORD                pdwTAID,
    LPDM_VARKEY            lpdmVarKey,
    DWORD                  dwItems,
    BOOL                    fWaitForCompletion,
    DWORD                  dwTimeOut,
    DM_NOTIFY_VARIABLEEX_PROC lpfnVariable,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);

```

Parameters**pdwTAID**

Pointer to a tag that contains the transaction ID assigned by the data manager after a successful call of the function.

lpdmVarKey

Pointer to the DM_VARKEY (Page 1590) structure that identifies the tag values to be read.

dwItems

Number of passed structures (corresponds to the number of tag values to be read).

fWaitForCompletion

If the fWaitForCompletion flag is set, the passed callback function is called only called when either all the requested tags are updated or the specified timeout period is exceeded.

If fWaitForCompletion is not set, the passed callback function is called immediately. The time characteristics correspond to the function DMGetValue.

dwTimeout

Maximum waiting time of the application in ms. If not all tag values have been updated when the waiting time expires, the callback function with corresponding error codes is called.

If the fWaitForCompletion == FALSE flag is set, the passed value is not evaluated.

lpfnVariable

Pointer to your DM_NOTIFY_VARIABLEEX_PROC callback function, which is called either after the update of all requested tags or after the waiting period.

If fWaitForCompletion == FALSE, the callback function is called immediately with the values currently set in the process image of the data manager.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Values determined.

FALSE

Error.

Related functions

DM_NOTIFY_VARIABLEEX_PROC (Page 1817)	Get tag values with notification (callback)
DMGetValueEx (Page 1689)	Get tag values
DMSetValueWait (Page 1742)	Change tag values with notification

See also

DM_NOTIFY_VARIABLE_EXSTR (Page 1817)

DMGetValueWait (Page 1700)

DM_VARKEY (Page 1590)

DMGetValueEx (Page 1689)

DMSetValueWait (Page 1742)

3.2.6.15 DMGetValueWaitExStr (RT Professional)**Declaration**

```

BOOL DMGetValueWaitExStr (
    LPDWORD                pdwTAID,
    DWORD                  dwFlags
    LPVARIANT               lpvdmVarKey,
    LPVARIANT               lpvCookie,
    BOOL                    fWaitForCompletion,
    DWORD                   dwTimeOut,
    DM_NOTIFY_VARIABLE_EXSTR lpfnVariable,
    LPVOID                  lpvUser,
    LPCMN_ERROR              lpdmError);

```

Description

Reads the value of one or more tags from the process image of the data manager. The value that is read out is the one existing at the time of the most recent update.

In contrast to DMGetValueExStr, this function can be used to enforce a value-based update of the tag values.

In contrast to DMGetValueEx, the DM_VARKEY structure is no longer used; as a result no length constraint applies with tag names.

Parameters**pdwTAID**

Pointer to a tag containing the transaction ID allocated by the data manager after the function has been called successfully.

dwFlags

If the tag name needs to be returned in the VT_LPSTR format in the VARIANT of the DM_VAR_UPDATE_STRUCT_EXSTR structure of DM_NOTIFY_VARIABLE_EXSTR, the DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR flag can be specified here.

lpvdmVarKey

Pointer to a VARIANT for the tag list. You must create the list as VT_ARRAY | VT_VARIANT, as various data types such as VT_I4, VT_BSTR and, in some cases, also VT_LPSTR can be present.

lpvCookie

Pointer to a VARIANT for an additional list of user-dependent data for each tag.

This is intended as substitute for lpvUserData of the previous DM_VARKEY and is also returned in the DM_VAR_UPDATE_STRUCT_EXSTR structure for each tag.

fWaitForCompletion

If the fWaitForCompletion flag has been set, the callback function passed is not called before all tags requested have been updated or the timeout time specified has been exceeded.

If the fWaitForCompletion flag has not been set, the callback function is called immediately. The timing is identical with that of the DMGetValueExStr function.

dwTimeout

Maximum waiting time of the application in ms. If not all tag values have been updated when the waiting time expires, the callback function with corresponding error codes is called.

If the fWaitForCompletion flag = FALSE, the value passed is not evaluated.

lpfnVariable

Pointer to your DM_NOTIFY_VARIABLEEX_PROC_EXSTR callback function which is called either after all requested tags have been updated or after the waiting time has expired.

If fWaitForCompletion = FALSE, the callback function is called immediately with the values currently stored in the process image of the data manager.

If a program requests a notify routine, it has to clear its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

On rare occasions, the notify may be returned before the function call has returned.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the callback function.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Values have been determined

FALSE

Error

Required files

dmclient_exstr.h
dmclient.lib
dmclient.dll

Related functions

DMGetValueExStr	Get tag values
DMSetValueWaitExStr	Change tag values with notification

Examples

You can find samples on the DMGetValueExStr page.

3.2.6.16 DMGetVarInfo (RT Professional)

Use

DMGetVarInfo determines the full tag key of a tag. This command can be used to obtain a tag name associated with a tag ID and vice versa.

Declaration

```
BOOL DMGetVarInfo (  
    LPCSTR      lpszProjectFile,  
    LPDM_VARKEY lpdmVarKey,  
    DWORD       dwItems,  
    LPCMN_ERROR lpdmError);
```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

lpdmVarKey

Pointer to the DM_VARKEY (Page 1590) structure to be completed with the tag keys.

dwItems

Number of completed structures.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Get tag key.

FALSE

Error.

Comment

Do not use the call alternating with calls for creating tags. If DMGetVarInfo is used after creating a tag the first time, search lists are resorted, which costs an additional execution time.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

OnTestVariablenGetVarInfo (Page 1853) "TESTCDoc.cpp"

See also

DM_VARKEY (Page 1590)

OnTestVariablenGetVarInfo (Page 1853)

3.2.6.17 DMGetVarInfoExStr (RT Professional)

Declaration

```
BOOL DMGetVarInfoExStr (  
    LPCTSTR        lpszProjectFile,  
    DWORD          dwFlags,  
    LPVARIANT       lpvdmVarKeyIn,  
    LPVARIANT       lpvdmVarKeyOut,  
    LPCMN_ERROR     lpdmError);
```

Description

Determines the complete tag key of a tag. This command enables you to obtain the tag name to a tag ID and vice versa.

In contrast to DMGetVarInfo, the tag list is passed on as pointer to VARIANT with VT_ARRAY | VT_VARIANT. It is returned in a second VARIANT. As a result, no length limit applies to tag names.

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMEnumOpenedProjects, or with DMGetRuntimeProject in Runtime.

dwFlags

If the tag name needs to be returned in the VT_LPSTR format in the VARIANT lpvdmVarKeyOut, DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR can be specified here.

lpvdmVarKeyIn

Pointer to VARIANT with VT_ARRAY | VT_VARIANT to pass the tag list or single VARIANT when only one tag is specified.

A TagID is entered in the relevant list element with the type VT_I4 and the tag name with VT_BSTR. In addition, VT_LPSTR (PROPVARIANT) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. VT_LPSTR is then converted internally into the required VT_BSTR type.

lpvdmVarKeyOut

Pointer to VARIANT used to return the corresponding values for lpvdmVarKeyIn.

Then, the tag ID for a given tag name in lpvdmVarKeyIn is returned or vice versa.

If the VARIANT is initialized with VT_EMPTY, a corresponding array is created and filled with VT_ARRAY | VT_VARIANT of the same size as in lpvdmVarKeyIn.

If the VARIANT is not empty, a VariantClear is called first before a new one is created. A VARIANT which is not initialized can result in an exception.

If the tag does not exist or cannot be accessed, the type of the corresponding entry remains VT_EMPTY.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Tag keys have been determined

FALSE

Error

Comment

This call should not be used alternately with calls for creating tags. Whenever the DMGetVarInfo or DMGetVarInfoExStr function is used for the first time after creating a tag search lists must be sorted again. This requires additional runtime.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Examples

Script sample Button DMGetVarInfoExStr:

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("kernel32.dll")
#define CP_ACP 0
int MultiByteToWideChar(UINT CodePage,
                        DWORD dwFlags,
                        LPCSTR lpMultiByteStr,
                        int cbMultiByte,
```

3.2 Data management functions (RT Professional)

```

        LPWSTR lpWideCharStr,
        int cchWideChar);
int WideCharToMultiByte(UINT CodePage,
        DWORD dwFlags,
        LPCWSTR lpWideCharStr,
        int cchWideChar,
        LPSTR lpMultiByteStr,
        int cbMultiByte,
        LPCSTR lpDefaultChar,
        LPBOOL lpUsedDefaultChar);

#pragma code()

#pragma code ("OleAut32.dll")
#include "OleAuto.h"
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound, unsigned
int cElements );
HRESULT SafeArrayPutElement(SAFEARRAY FAR* psa, long FAR* rgIndices,
void FAR* pv );
HRESULT SafeArrayGetElement(SAFEARRAY FAR* psa, long FAR* rgIndices,
void FAR* pv );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR* rgIndices,
void HUGE* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayUnlock(SAFEARRAY FAR* psa);
typedef WCHAR OLECHAR;
BSTR SysAllocString(OLECHAR FAR* sz);
HRESULT SysFreeString(BSTR bstr);
#pragma code()
VARIANT vVarkeyIn;
VARIANT vVarkeyOut;
VARIANT vElem;
VARIANT* pvElem;
HRESULT hr;
CHAR szProjectName[256];
CMN_ERROR err;
DWORD dwFlags;
BOOL bRet;
DWORD dwVarID1, dwVarID2, dwVarID3, dwVarID4;
CHAR szVarNam1[256], szVarNam2[256], szVarNam3[256], szVarNam4[256];
SAFEARRAY* parrayIN;
SAFEARRAY* parrayOUT;
long lInx;
WCHAR wszBuffer[256];
int nRet;
memset(&err, 0, sizeof(err));
bRet = FALSE;
szProjectName[0] = 0;
dwFlags = 0;
VariantInit(&vVarkeyIn);
VariantInit(&vVarkeyOut);
VariantInit(&vElem);
parrayIN = NULL;

```

```

parrayOUT = NULL;
//hr = E_FAIL;
nRet = 0;
printf("\r\n\r\n##### enter Test with DMGetVarInfoExStr
#####");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n error DMGetRuntimeProject =>[%s], err=%ld,%ld,%ld,
%ld,%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
dwVarID1 = GetTagDWord("dwVarKeyID_1");
dwVarID2 = GetTagDWord("dwVarKeyID_2");
dwVarID3 = GetTagDWord("dwVarKeyID_3");
dwVarID4 = GetTagDWord("dwVarKeyID_4");
strncpy(szVarNam1, GetTagChar("szVarKeyName_1"), 256);
strncpy(szVarNam2, GetTagChar("szVarKeyName_2"), 256);
strncpy(szVarNam3, GetTagChar("szVarKeyName_3"), 256);
strncpy(szVarNam4, GetTagChar("szVarKeyName_4"), 256);
printf("\r\n set vVarkeyIn:");
printf("\r\n dwVarID1=%ld, szVarNam1=[%s] (use name as VT_BSTR)",
dwVarID1, szVarNam1);
printf("\r\n dwVarID2=%ld, szVarNam2=[%s] (use name as VT_LPSTR)",
dwVarID2, szVarNam2);
printf("\r\n dwVarID3=%ld, szVarNam3=[%s] (use ID)", dwVarID3,
szVarNam3);
printf("\r\n dwVarID4=%ld, szVarNam4=[%s] (useID)", dwVarID4,
szVarNam4);
//preseting input Variant: name1, name2, id3, id4
// will deliver on output Variant: id1, id2, name3, name4
parrayIN = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vVarkeyIn.vt = (VT_ARRAY | VT_VARIANT);
vVarkeyIn.u.parray = parrayIN;
// name1 as VT_BSTR
vElem.vt = VT_BSTR;
nRet = MultiByteToWideChar(CP_ACP, 0, szVarNam1, -1, wszBuffer,
256);
vElem.u.bstrVal = SysAllocString(wszBuffer);
lInx = 0;
hr = SafeArrayPutElement(parrayIN, &lInx, &vElem);
VariantClear(&vElem);
// name2 als VT_LPSTR
SafeArrayLock(parrayIN);
lInx = 1;
hr = SafeArrayPtrOfIndex(parrayIN, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam2;
SafeArrayUnlock(parrayIN);
if (hr)

```

```

{
    printf("\r\nerror SafeArrayPutElement: hr = %08lx", hr);
}
VariantClear(&vElem);
vElem.vt = VT_I4;
vElem.u.lVal = dwVarID3;
lInx = 2;
hr = SafeArrayPutElement(parrayIN, &lInx, &vElem);
vElem.vt = VT_I4;
vElem.u.lVal = dwVarID4;
lInx = 3;
hr = SafeArrayPutElement(parrayIN, &lInx, &vElem);

memset(&err, 0, sizeof(err));
bRet = DMGetVarInfoExStr(szProjectName, dwFlags, &vVarkeyIn,
&vVarkeyOut, &err);
if (!bRet)
{
    printf("\r\n error DMGetVarInfoExStr[%s], err=%ld,%ld,%ld,%ld,
%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
    if (err.dwError1 == DM_E_DONT_EXIST)
    {
        //reset error, elements will be checked each for error in
type (VT_EMPTY == not exist)
        bRet = TRUE;
    }
}
else
{
    printf("\r\n DMGetVarInfoExStr return OK");
}
if (bRet)
{
    if (vVarkeyOut.vt == VT_EMPTY)
    {
        printf("\r\n vVarkeyOut is VT_EMPTY");
    }
    else
    {
        parrayOUT = vVarkeyOut.u.parray;
        lInx = 0;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
        if (vElem.vt != VT_I4)
        {
            printf("\r\n vVarkeyOut[%ld].vt = [%ld] != VT_I4", lInx,
vElem.vt);
            dwVarID1 = 0L;
        }
        else
        {

```



```

        dwVarID1 = vElem.u.lVal;
    }
    SetTagDWord("dwVarKeyID_1", dwVarID1);
    lInx = 1;
    SafeArrayGetElement(parrayOUT, &lInx, &vElem);
    if (vElem.vt != VT_I4)
    {
        printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_I4", lInx,
vElem.vt);
        dwVarID2 = 0L;
    }
    else
    {
        dwVarID2 = vElem.u.lVal;
    }
    SetTagDWord("dwVarKeyID_2", dwVarID2);
    lInx = 2;
    SafeArrayGetElement(parrayOUT, &lInx, &vElem);
    if (vElem.vt != VT_BSTR)
    {
        printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_BSTR",
lInx, vElem.vt);
        strcpy(szVarNam3, "error!!!");
    }
    else
    {
        nRet = WideCharToMultiByte(CP_ACP, 0L,
(WCHAR*)vElem.u.bstrVal, -1,
                                szVarNam3, 256, NULL, NULL);
    }
    SetTagChar("szVarKeyName_3", szVarNam3);
    lInx = 3;
    SafeArrayGetElement(parrayOUT, &lInx, &vElem);
    if (vElem.vt != VT_BSTR)
    {
        printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_BSTR",
lInx, vElem.vt);
        strcpy(szVarNam4, "error!!!");
    }
    else
    {
        nRet = WideCharToMultiByte(CP_ACP, 0L,
(WCHAR*)vElem.u.bstrVal, -1,
                                szVarNam4, 256, NULL, NULL);
    }
    SetTagChar("szVarKeyName_4", szVarNam4);
    printf("\r\n  get vVarkeyOut:");
    printf("\r\n    dwVarID1=%ld, szVarNam1=[%s] (delivered ID)",
dwVarID1, szVarNam1);
    printf("\r\n    dwVarID2=%ld, szVarNam2=[%s] (delivered ID)",
dwVarID2, szVarNam2);

```

3.2 Data management functions (RT Professional)

```

        printf("\r\n      dwVarID3=%ld, szVarNam3=[%s] (delivered name
VT_BSTR)", dwVarID3, szVarNam3);
        printf("\r\n      dwVarID4=%ld, szVarNam4=[%s] (delivered name
VT_BSTR)", dwVarID4, szVarNam4);
    }
}
printf("\r\n  set flag DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR and
repeat DMGetVarInfoExStr");
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
memset(&err, 0, sizeof(err));
hr = VariantClear(&vVarkeyOut);
if (hr)
{
    printf("\r\nerror VariantClear(vVarkeyOut: hr = %08lx", hr);
}
bRet = DMGetVarInfoExStr(szProjectName, dwFlags, &vVarkeyIn,
&vVarkeyOut, &err);
if (!bRet)
{
    printf("\r\n  error DMGetVarInfoExStr[%s], err=%ld,%ld,%ld,%ld,
%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
    if (err.dwError1 == DM_E_DONT_EXIST)
    {
        //reset error, elements will be checked each for error in
type (VT_EMPTY == not exist)
        bRet = TRUE;
    }
}
else
{
    printf("\r\n  DMGetVarInfoExStr return OK");
}
if (bRet)
{
    if (vVarkeyOut.vt == VT_EMPTY)
    {
        printf("\r\n  vVarkeyOut is VT_EMPTY");
    }
    else
    {
        parrayOUT = vVarkeyOut.u.parray;
        lInx = 0;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
        if (vElem.vt != VT_I4)
        {
            printf("\r\n  vVarkeyOut[%ld].vt = [%ld] != VT_I4", lInx,
vElem.vt);
            dwVarID1 = 0L;
        }
        else

```

```

    {
        dwVarID1 = vElem.u.lVal;
    }
    SetTagDWord("dwVarKeyID_1", dwVarID1);
    lInx = 1;
    SafeArrayGetElement(parrayOUT, &lInx, &vElem);
    if (vElem.vt != VT_I4)
    {
        printf("\r\n vVarKeyOut[%ld].vt = [%ld] != VT_I4", lInx,
vElem.vt);
        dwVarID2 = 0L;
    }
    else
    {
        dwVarID2 = vElem.u.lVal;
    }
    SetTagDWord("dwVarKeyID_2", dwVarID2);

    SafeArrayLock(parrayOUT);
    lInx = 2;
    SafeArrayPtrOfIndex(parrayOUT, &lInx, &pvElem);
    if (pvElem->vt != VT_LPSTR)
    {
        printf("\r\n vVarKeyOut[%ld].vt = [%ld] != VT_LPSTR",
lInx, pvElem->vt);
        strcpy(szVarNam3, "error!!!");
    }
    else
    {
        strncpy(szVarNam3, (LPSTR)pvElem->u.pbVal, 255);
        szVarNam3[255] = 0;
    }
    SetTagChar("szVarKeyName_3", szVarNam3);
    lInx = 3;
    SafeArrayPtrOfIndex(parrayOUT, &lInx, &pvElem);
    if (pvElem->vt != VT_LPSTR)
    {
        printf("\r\n vVarKeyOut[%ld].vt = [%ld] != VT_LPSTR",
lInx, pvElem->vt);
        strcpy(szVarNam4, "error!!!");
    }
    else
    {
        strncpy(szVarNam4, (LPSTR)pvElem->u.pbVal, 255);
        szVarNam4[255] = 0;
    }
    SetTagChar("szVarKeyName_4", szVarNam4);
    SafeArrayUnlock(parrayOUT);

    printf("\r\n get vVarkeyOut:");
    printf("\r\n dwVarID1=%ld, szVarNam1=[%s] (delivered ID)",
dwVarID1, szVarNam1);

```

3.2 Data management functions (RT Professional)

```

        printf("\r\n    dwVarID2=%ld, szVarNam2=[%s] (delivered ID)",
dwVarID2, szVarNam2);
        printf("\r\n    dwVarID3=%ld, szVarNam3=[%s] (delivered name
VT_LPSTR)", dwVarID3, szVarNam3);
        printf("\r\n    dwVarID4=%ld, szVarNam4=[%s] (delivered name
VT_LPSTR)", dwVarID4, szVarNam4);
    }
}

printf("\r\n  only 1 Input (ID = VT_I4 and dwVarID1) without
VT_ARRAY, set flag DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR and repeat
DMGetVarInfoExStr");
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
memset(&err, 0, sizeof(err));
hr = VariantClear(&vVarkeyIn);
if (hr)
{
    printf("\r\n  error VariantClear(vVarkeyIn): hr = %08lx", hr);
    // know that error, have to free the PROPVARIANT in Array manually
    //this can be set to VT_EMPTY direct, while no alloc of string
was used
    parrayIN = vVarkeyIn.u.parray;
    SafeArrayLock(parrayIN);
    lInx = 1;
    hr = SafeArrayPtrOfIndex(parrayIN, &lInx, &pvElem);
    pvElem->vt = VT_EMPTY;
    pvElem->u.pbVal = NULL;
    SafeArrayUnlock(parrayIN);
    hr = VariantClear(&vVarkeyIn);
    if (hr)
    {
        printf("\r\n  error VariantClear(vVarkeyIn) again after
manually reset the PROPVARIANT in 2. elem: hr = %08lx", hr);
    }
    else
    {
        printf("\r\n  VariantClear(vVarkeyIn) OK after manually reset
the PROPVARIANT in 2. elem");
    }
}
hr = VariantClear(&vVarkeyOut);
if (hr)
{
    printf("\r\n  error VariantClear(vVarkeyOut): hr = %08lx", hr);
}
vVarkeyIn.vt = VT_I4;
vVarkeyIn.u.lVal = dwVarID1;
bRet = DMGetVarInfoExStr(szProjectName, dwFlags, &vVarkeyIn,
&vVarkeyOut, &err);
if (!bRet)
{

```

```

        printf("\r\n  error DMGetVarInfoExStr[%s], err=%ld,%ld,%ld,%ld,
%ld,[%s]", szProjectName,
            err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
        if (err.dwError1 == DM_E_DONT_EXIST)
        {
            //reset error, elements will be checked each for error in
type (VT_EMPTY == not exist)
            bRet = TRUE;
        }
    }
else
{
    printf("\r\n  DMGetVarInfoExStr return OK");
}
if (bRet)
{
    if (vVarkeyOut.vt == VT_EMPTY)
    {
        printf("\r\n  vVarkeyOut is VT_EMPTY");
    }
    else
    {
        if (vVarkeyOut.vt != VT_LPSTR)
        {
            printf("\r\n  vVarKeyOut.vt = [%ld] != VT_LPSTR",
vVarkeyOut.vt);
            strcpy(szVarNam1, "error!!!");
        }
        else
        {
            strncpy(szVarNam1, (LPSTR)vVarkeyOut.u.pbVal, 255);
            szVarNam1[255] = 0;
        }
        SetTagChar("szVarKeyName_1",szVarNam1);
    }
}

printf("\r\n##### exit Test with DMGetVarInfoExStr #####\r
\n");
}

```

3.2.6.18 DMGetVarLimits (RT Professional)

Use

DMGetVarLimits determines the limits for the specified tag, within which write operations are possible with DMSetValue functions.

Declaration

```
BOOL DMGetVarLimits (
    LPCSTR      lpszProjectFile,
    LPDM_VARKEY lpdmVarKey,
    DWORD       dwItems,
    LPDM_VARLIMIT lpdmVarLimit,
    LPCMN_ERROR lpdmError );
```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

lpdmVarKey

Pointer to a DM_VARKEY (Page 1590) type structure for specification of the tags.

dwItems

Number of completed structures.

lpdmVarLimit

Pointer to the DM_VARLIMIT (Page 1593) type structure in which the limits are stored.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Limits determined.

FALSE

Error.

Comment

In addition to the high and low limits of a tag defined in the configuration, a further limit to the allowed number range is set by any format specifications made.

The following example shows the relationships for a DM_VARTYPE_BYTE type tag that is stored in an automation system as a byte in BCD format.

Error messages

DM_E_MAX_RANGE	High limit of format conversion violated
DM_E_MIN_RANGE	Low limit of format conversion violated
DM_E_MAX_LIMIT	High limit of tag violated
DM_E_MIN_LIMIT	Low limit of tag violated
DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found
DM_E_INVALID_TYPE	Invalid tag type

Required files

dmclient.h
dmclient.lib
dmclient.dll

Examples

OnTestVariablenGetvarlimits (Page 1854) "TESTCDoc.cpp"

See also

DM_VARKEY (Page 1590)
DM_VARLIMIT (Page 1593)
OnTestVariablenGetvarlimits (Page 1854)

3.2.6.19 DMGetVarLimitsExStr (RT Professional)

Declaration

```

BOOL DMGetVarLimitsExStr (
    LPCTSTR          lpszProjectFile,
    LPVARIANT         lpvdmVarKey,
    LPDM_VARLIMIT     lpdmVarLimit,
    LPCMN_ERROR       lpdmError );

```

Description

For the tags specified, DMGetVarLimitsExStr determines the limits within which it is possible to execute write operations using the DMSetValueExStr functions.

In contrast to DMGetVarLimits, the tag list is passed as pointer to VARIANT with VT_ARRAY | VT_VARIANT. As a result, no length limit applies to tag names.

Parameters

lpzProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMEnumOpenedProjects, or with DMGetRuntimeProject in Runtime.

lpvdmVarKey

Pointer to VARIANT with VT_ARRAY | VT_VARIANT to pass the tag list or single VARIANT when only one tag is specified.

A TagID is entered in the relevant list element with the type VT_I4 and the tag name with VT_BSTR. In addition, VT_LPSTR (PROPVARIANT) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. VT_LPSTR is then converted internally into the required VT_BSTR type.

lpdmVarLimit

Pointer to structures of the DM_VARLIMIT type to store the limit values.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Limit values have been determined

FALSE

Error

Comment

In addition to the configured high and low limits of a tag, a stored format instruction leads to a further restriction of the valid numerical value range.

The following sample explains this situation for a tag of the DM_VARTYPE_BYTE type stored in the AS in the BCD byte format.

Error messages

DM_E_MAX_RANGE	High limit of format conversion violated
DM_E_MIN_RANGE	Low limit of format conversion violated
DM_E_MAX_LIMIT	High limit of tag violated
DM_E_MIN_LIMIT	Low limit of tag violated
DM_E_NOT_CONNECTED	No connection to data manager

DM_E_INVALID_KEY	Tag not found
DM_E_INVALID_TYPE	Invalid tag type

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Examples

Script sample Button DMGetVarLimitsExStr:

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("OleAut32.dll")
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound, unsigned
int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR* rgIndices,
void HUGE* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayUnlock(SAFEARRAY FAR* psa);
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
#pragma code()

VARIANT vdmVarkey;
VARIANT* pvElem;
VARIANT vElem;
HRESULT hr;
CHAR szProjectName[256];
CMN_ERROR err;
BOOL bRet;
CHAR szVarNam1[256], szVarNam2[256], szVarNam3[256], szVarNam4[265];
SAFEARRAY* parray;
long lInx;
int nRet;
DM_VARLIMIT dmLimits[4];
int i;

memset(&err, 0, sizeof(err));
bRet = FALSE;
szProjectName[0] = 0;
VariantInit(&vElem);
VariantInit(&vdmVarkey);
parray = NULL;
nRet = 0;
i = 0;
```

3.2 Data management functions (RT Professional)

```

printf("\r\n\r\n##### enter Test with DMGetVarLimitsExStr
#####");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n error DMGetRuntimeProject =>[%s], err=%ld,%ld,%ld,
%ld,%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

strncpy(szVarNam1, GetTagChar("szVarKeyName_1"), 256);
strncpy(szVarNam2, GetTagChar("szVarKeyName_2"), 256);
strncpy(szVarNam3, GetTagChar("szVarKeyName_3"), 256);
strncpy(szVarNam4, GetTagChar("szVarKeyName_4"), 256);
printf("\r\n set the vdmVarkey with names in VT_LPSTR:");
parray = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vdmVarkey.vt = (VT_ARRAY | VT_VARIANT);
vdmVarkey.u.parray = parray;
SafeArrayLock(parray);
lInx = 0;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam1;
lInx = 1;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam2;
lInx = 2;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam3;
lInx = 3;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam4;
SafeArrayUnlock(parray);

memset(&err, 0, sizeof(err));
memset(dmLimits, 0, sizeof(DM_VARLIMIT)*4);
bRet = DMGetVarLimitsExStr(szProjectName, &vdmVarkey, dmLimits,
&err);
if (!bRet)
{
    printf("\r\n error DMGetVarLimitsExStr[%s], err=%ld,%ld,%ld,%ld,
%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
    bRet = TRUE; //clear error flag, the defect data is set to VT_EMPTY
}
else

```

```

{
    printf("\r\n  DMGetVarLimitsExStr return OK");
}
if (bRet)
{
    for (i = 0; i < 4; i++)
    {
        VariantClear(&vElem);
        printf("\r\n    dmLimits[%d]:{", i);
        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMaxRange, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMaxRange.vt))
        {
            printf("dmMaxRange=[%g], ", vElem.u.dblVal);
        }
        else
        {
            printf("dmMaxRange{VariantChangeType error hr = 0x%08lx,
vt=0x%04x}",
                hr, dmLimits[i].dmMaxRange.vt);
        }
        VariantClear(&vElem);
        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMinRange, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMinRange.vt))
        {
            printf("dmMinRange=[%g], ", vElem.u.dblVal);
        }
        else
        {
            printf("dmMinRange{VariantChangeType error hr = 0x%08lx,
vt=0x%04x}",
                hr, dmLimits[i].dmMinRange.vt);
        }
        VariantClear(&vElem);
        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMaxLimit, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMaxLimit.vt))
        {
            printf("dmMaxLimit=[%g], ", vElem.u.dblVal);
        }
        else
        {
            printf("dmMaxLimitVariantChangeType error hr = 0x%08lx,
vt=0x%04x}",
                hr, dmLimits[i].dmMaxLimit.vt);
        }
        VariantClear(&vElem);
    }
}

```

```

        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMinLimit, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMinLimit.vt))
        {
            printf("dmMinLimit=[%g]", vElem.u.dblVal);
        }
        else
        {
            printf("dmMinLimitVariantChangeType error hr = 0x%08lx,
vt=0x%04x}",
                    hr, dmLimits[i].dmMinLimit.vt);
        }
        printf(")");
    }
}

printf("\r\n##### exit Test with DMGetVarLimitsExStr #####
\r\n");
}

```

3.2.6.20 DMGetVarType (RT Professional)

Use

DMGetVarType determines the type reference for the passed tags.

Declaration

```

BOOL DMGetVarType (
    LPCSTR        lpszProjectFile,
    LPDM_VARKEY   lpdmVarKey,
    DWORD         dwItems,
    LPDM_TYPEREF  lpdmTypeRef,
    LPCMN_ERROR    lpdmError);

```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

IpdmVarKey

Pointer to the DM_VARKEY (Page 1590) structure with which the tag keys determined for the type.

dwItems

Number of completed structures.

IpdmTypeRef

Pointer to the list of DM_TYPEREF (Page 1576) structures for returning the details about the tag type.

IpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Data type determined.

FALSE

Error.

Comment

The configuration functions for creating tags pass the completed DM_VARREF and DM_TYPEREF structures to the calling application with details of the data type of the tag.

To save memory space within the application, it does not necessarily have to remember the type reference to each tag itself; it can query this information with DMGetVarType at any time from the data manager.

Do not use the call alternating with calls for creating tags. If DMGetVarType is used after creating a tag the first time, search lists are resorted, which costs an additional execution time.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

OnTestVariablenGetvartype (Page 1855) "TESTCDoc.cpp"

See also

DM_VARKEY (Page 1590)

DM_TYPEREF (Page 1576)

OnTestVariablenGetvartype (Page 1855)

3.2.6.21 DMGetVarTypeExStr (RT Professional)

Declaration

```

BOOL DMGetVarTypeExStr (
    LPCTSTR                lpszProjectFile,
    LPVARIANT              lpvdmVarKey,
    LPDM_TYPEREF_EXSTR     lpdmTypeRef,
    LPCMN_ERROR            lpdmError);

```

Description

Determines the type reference for the tags passed.

In contrast to DMGetVarType, the tag list is passed as pointer to VARIANT with VT_ARRAY | VT_VARIANT. As a result, no length limit applies to tag names. This applies also for the type names in the DM_TYPEREF_EXSTR structures.

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMEnumOpenedProjects, or with DMGetRuntimeProject in Runtime.

lpvdmVarKey

Pointer to VARIANT with VT_ARRAY | VT_VARIANT for passing the tag list.

A TagID is entered in the relevant list element with the type VT_I4 and the tag name with VT_BSTR. In addition, VT_LPSTR (PROPVARIANT) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. VT_LPSTR is then converted internally into the required VT_BSTR type.

lpdmTypeRef

Pointer to the list of DM_TYPEREF_EXSTR structures that return tag type information.

The list must have at least the size of the variant array in `lpvdmVarKey`. The pointers `lpszTypeName` and `dwBufferCount` must be initialized and point to a separate string array.

If `lpszTypeName` is NULL check the return. Allocated pointer has to enable it again.

lpdmError

Pointer to the data of the extended error message within the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Data type has been determined

FALSE

Error

Comment

In order to economize the memory management within the application, the application is not forced to store the type reference for each tag. Instead, it can obtain this information from the data manager anytime by means of `DMGetVarType`.

This call should not be used alternately with calls for creating tags. Whenever the "`DMGetVarType`" or "`DMGetVarTypeExStr`" function is used for the first time after creating a tag, search lists must be sorted again. This requires additional runtime.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found

Required files

`dmclient_exstr.h`

`dmclient.lib`

`dmclient.dll`

Examples

Script sample Button DMGetVarTypeExStr:

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("OleAut32.dll")
```

3.2 Data management functions (RT Professional)

```

SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound, unsigned
int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR* rgIndices,
void HUGE* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayUnlock(SAFEARRAY FAR* psa);
#pragma code()

VARIANT vdmVarkey;
VARIANT* pvElem;
HRESULT hr;
CHAR szProjectName[256];
CMN_ERROR err;
BOOL bRet;
CHAR szVarNam1[256], szVarNam2[256], szVarNam3[256], szVarNam4[256];
SAFEARRAY* parray;
long lInx;
int nRet;
DM_TYPEREF_EXSTR dmTypeRef[4];
CHAR szTypeName[4][256];
int i;

memset(&err, 0, sizeof(err));
memset(dmTypeRef, 0, sizeof(DM_TYPEREF_EXSTR)*4);
szTypeName[0][0] = 0;
szTypeName[1][0] = 0;
szTypeName[2][0] = 0;
szTypeName[3][0] = 0;
bRet = FALSE;
szProjectName[0] = 0;
VariantInit(&vdmVarkey);
parray = NULL;
nRet = 0;
i = 0;

printf("\r\n\r\n##### enter Test with DMGetVarTypeExStr
#####");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n error DMGetRuntimeProject =>[%s], err=%ld,%ld,%ld,
%ld,%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

strncpy(szVarNam1, GetTagChar("szVarKeyName_1"), 256);
strncpy(szVarNam2, GetTagChar("szVarKeyName_2"), 256);
strncpy(szVarNam3, GetTagChar("szVarKeyName_3"), 256);
strncpy(szVarNam4, GetTagChar("szVarKeyName_4"), 256);
printf("\r\n set the vdmVarkey with names in VT_LPSTR:");

```



```

parray = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vdmVarkey.vt = (VT_ARRAY | VT_VARIANT);
vdmVarkey.u.parray = parray;
SafeArrayLock(parray);
lInx = 0;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam1;
lInx = 1;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam2;
lInx = 2;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam3;
lInx = 3;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam4;
SafeArrayUnlock(parray);

printf("\r\n  call DMGetVarTypeExStr with NULL for names");
memset(&err, 0, sizeof(err));
bRet = DMGetVarTypeExStr(szProjectName, &vdmVarkey, dmTypeRef, &err);
if (!bRet)
{
    printf("\r\n  error DMGetVarTypeExStr[%s], err=%ld,%ld,%ld,%ld,
%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMGetVarTypeExStr return OK");
}
if (bRet)
{
    for (i = 0; i < 4; i++)
    {
        printf("\r\n      dmTypeRef[%d]:{", i);
        printf("dwType=[%ld]", dmTypeRef[i].dwType);
        printf(", dwSize=[%ld]", dmTypeRef[i].dwSize);
        if (dmTypeRef[i].lpszTypeName)
        {
            printf(", lpszTypeName=[%s]", dmTypeRef[i].lpszTypeName);
            printf(", dwNameCharCount=[%ld]",
dmTypeRef[i].dwNameCharCount);
            //clear the delivered strings that no memoryleak
            free(dmTypeRef[i].lpszTypeName);
            dmTypeRef[i].lpszTypeName = NULL;
            dmTypeRef[i].dwNameCharCount = 0L;
        }
    }
}

```

```

        //set name store for next read (no need to free after
call)
        dmTypeRef[i].lpszTypeName = szTypeName[i];
        dmTypeRef[i].dwNameCharCount = 255;
    }
    printf("{}");
}
}
printf("\r\n  call DMGetVarTypeExStr with pointers and size for name
storage");
memset(&err, 0, sizeof(err));
bRet = DMGetVarTypeExStr(szProjectName, &vdmVarkey, dmTypeRef, &err);
if (!bRet)
{
    printf("\r\n  error DMGetVarTypeExStr[%s], err=%ld,%ld,%ld,%ld,
%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMGetVarTypeExStr return OK");
}
if (bRet)
{
    for (i = 0; i < 4; i++)
    {
        printf("\r\n    dmTypeRef[%d]:{", i);
        printf("dwType=[%ld]", dmTypeRef[i].dwType);
        printf(", dwSize=[%ld]", dmTypeRef[i].dwSize);
        printf(", lpszTypeName=[%s]", dmTypeRef[i].lpszTypeName);
        printf(", dwNameCharCount=[%ld]",
dmTypeRef[i].dwNameCharCount);
        printf("{}");
    }
}
printf("\r\n##### exit Test with DMGetVarTypeExStr #####\r
\n");
}

```

3.2.6.22 DMSetValue (RT Professional)

Use

The function changes the values of the tags described by `lpdmVarKey` to the values specified in `lpdmValue`.

Changes are made to the tags asynchronously, according to the motto "fire and forget".

Declaration

```

BOOL DMSetValue (
    LPDM_VARKEY    lpdmVarKey,
    DWORD          dwItems,
    LPVARIANT       lpdmValue,
    LPDWORD         lpdmVarState,
    LPCMN_ERROR     lpdmError);

```

Parameters

lpdmVarKey

Pointer to the first DM_VARKEY (Page 1590) structure that identifies the tag values to be changed.

dwItems

Number of passed structures (corresponds to the number of tag values to be changed).

lpdmValue

Pointer to the first of the new values for the tags to be changed.

lpdmVarState

Pointer to the first memory location where information is stored indicating whether the tag value could be changed successfully or whether errors have occurred.

The value 0 (OK) means that the transmission/modification has been completed successfully and that at least one of the specified tags exists. An error with the DM_VARSTATE_INVALID_KEY status only occurs if none of the specified tags exist.

DMSetValueWait must be used to check if the value has been applied.

DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	Connection to partner not established
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	Protocol error
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	Network module faulty
DM_VARSTATE_MAX_LIMIT	(0x0008)	Configured high limit violated
DM_VARSTATE_MIN_LIMIT	(0x0010)	Configured low limit violated
DM_VARSTATE_MAX_RANGE	(0x0020)	High format limit violated
DM_VARSTATE_MIN_RANGE	(0x0040)	Low format limit violated
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	Display conversion error (in conjunction with DM_VARSTATE_..._RANGE)
DM_VARSTATE_STARTUP_VALUE	(0x0100)	Initialization value of tag
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	Substitute value of tag
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	Addressing error in channel
DM_VARSTATE_INVALID_KEY	(0x0800)	Tag not found / not available
DM_VARSTATE_ACCESS_FAULT	(0x1000)	Access to tag not permitted
DM_VARSTATE_TIMEOUT	(0x2000)	Timeout / no feedback from channel
DM_VARSTATE_SERVERDOWN	(0x4000)	Server is down

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tags changed.

FALSE

Error.

Comment

Specification of an access rule for lpdmValue is not necessary, since the data manager can map each type based on the passed tag key.

Write access is performed in the process for external tags. For WinCC client PCs, the request is forwarded to the respective server.

Normally, write requests of the applications are executed asynchronously. All write requests are listed in the queue of the data manager on the server and passed sequentially to the appropriate channel DLL.

If the channel DLL positively confirms the write request, the new value becomes valid in the process image of the server responsible for the request. The update is then performed within the WinCC stations which are interconnected by means of the system bus.

As a result, a tag can temporarily have different values on different stations.

Error messages

DM_E_MAX_LIMIT	High limit of tag violated
DM_E_MIN_LIMIT	Low limit of tag violated
DM_E_MAX_RANGE	High limit of format conversion violated
DM_E_MIN_RANGE	Low limit of format conversion violated
DM_E_ACCESS_FAULT	Prohibited write access to a tag
DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMGetValue (Page 1687)	Get tag values
DMSetValueMessage (Page 1738)	Change tag values with alarm output
DMSetValueWait (Page 1742)	Change tag values with notification
DMSetValueWaitMessage (Page 1747)	Change tag values with notification and alarm output

Examples

Write tag (Page 1867) "DM02.cpp"

OnTestVariablenSetvalue (Page 1857) "TESTCDoc.cpp"

See also

DM_VARKEY (Page 1590)

DMGetValue (Page 1687)

DMGetValueEx (Page 1689)

DMSetValueMessage (Page 1738)

DMSetValueWait (Page 1742)

DMSetValueWaitMessage (Page 1747)

Write tag (Page 1867)

OnTestVariablenSetvalue (Page 1857)

3.2.6.23 DMSetValueExStr (RT Professional)**Declaration**

```

BOOL DMSetValueExStr (
    LPVARIANT    lpvdmVarKey,
    LPVARIANT    lpvdmValue,
    LPDWORD      lpdmVarState,
    LPCMN_ERROR  lpdmError);

```

Description

This function changes the values of the tags described by lpvdmVarKey to obtain the values specified in lpvdmValue.

Tag modifications are transmitted in asynchronous mode according to the principle "fire and forget".

In contrast to `DMSetValue`, the tag list is passed on as pointer to `VARIANT` with `VT_ARRAY` | `VT_VARIANT`. As a result, no length limit applies to tag names.

Parameters

lpvdmVarKey

Pointer to `VARIANT` with `VT_ARRAY` | `VT_VARIANT` for passing on the tag list or single `VARIANT` if only one tag is specified.

A TagID is entered in the relevant list element with the type `VT_I4` and the tag name with `VT_BSTR`. In addition, `VT_LPSTR` (`PROPVARIANT`) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. `VT_LPSTR` is then converted internally into the required `VT_BSTR` type.

For a multi-client project in WinCC Version 5.0 or higher, it may be required here to add a server prefix to each name (refer to "Project types and versions").

lpvdmValue

Pointer to the first of the new values for the tags to be changed.

lpdmVarState

Pointer to a memory location which stores the information whether the value of the tag could be changed successfully or if errors occurred.

The value "0" (OK) means that the transmission/modification has been completed successfully and that at least one of the specified tags is existing. The `DM_VARSTATE_INVALID_KEY` error status is only returned if none of the specified tags exists.

In order to check if the value has been applied by the PLC also, `DMSetValueWaitExStr` must be used.

<code>DM_VARSTATE_NOT_ESTABLISHED</code>	(0x0001)	Connection to partner not established
<code>DM_VARSTATE_HANDSHAKE_ERROR</code>	(0x0002)	Protocol error
<code>DM_VARSTATE_HARDWARE_ERROR</code>	(0x0004)	Network module faulty
<code>DM_VARSTATE_MAX_LIMIT</code>	(0x0008)	Configured high limit violated
<code>DM_VARSTATE_MIN_LIMIT</code>	(0x0010)	Configured low limit violated
<code>DM_VARSTATE_MAX_RANGE</code>	(0x0020)	High format limit violated
<code>DM_VARSTATE_MIN_RANGE</code>	(0x0040)	Low format limit violated
<code>DM_VARSTATE_CONVERSION_ERROR</code>	(0x0080)	Conversion error display (in connection with <code>DM_VARSTATE_...._RANGE</code>)
<code>DM_VARSTATE_STARTUP_VALUE</code>	(0x0100)	Initialization value of tag
<code>DM_VARSTATE_DEFAULT_VALUE</code>	(0x0200)	Substitute value of tag
<code>DM_VARSTATE_ADDRESS_ERROR</code>	(0x0400)	Addressing error in the channel
<code>DM_VARSTATE_INVALID_KEY</code>	(0x0800)	Tag not found / not available
<code>DM_VARSTATE_ACCESS_FAULT</code>	(0x1000)	Access to tag not permitted
<code>DM_VARSTATE_TIMEOUT</code>	(0x2000)	Timeout / no feedback from channel
<code>DM_VARSTATE_SERVERDOWN</code>	(0x4000)	Server is down

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tags changed.

FALSE

Error

Comment

It is not necessary to specify an access instruction for lpdmValue because the tag keys passed enable the data manager to allocate the information appropriately.

For external tags, write access to the process is obtained while the job is passed on to the appropriate server by WinCC client computers.

The write jobs of the application are normally executed in asynchronous mode, i.e. all write jobs are lined up in the queue which the data manager keeps on the server, and then passed on to the appropriate channel DLL sequentially.

If the channel DLL confirms the write job affirmatively, the new value becomes valid in the process image of the server responsible for the job. The update is then performed within the WinCC stations which are interconnected by means of the system bus.

As a result, a tag can temporarily have different values on different stations.

Error messages

DM_E_MAX_LIMIT	High limit of tag violated
DM_E_MIN_LIMIT	Low limit of tag violated
DM_E_MAX_RANGE	High limit of format conversion violated
DM_E_MIN_RANGE	Low limit of format conversion violated
DM_E_ACCESS_FAULT	Prohibited write access to a tag
DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMGetValueExStr	Get tag values
DMSetValueMessageExStr	Change tag values with alarm output
DMSetValueWaitExStr	Change tag values with notification
DMSetValueWaitMessageExStr	Change tag values with notification and alarm output

Examples

You can find samples on the DMGetValueExStr page.

3.2.6.24 DMSetValueMessage (RT Professional)**Use**

The function changes the value of the tag described by `lpdmVarKey` to the value specified in `lpdmValue`.

Only one tag is changed, in contrast to the `DMSetValue` function. After successful update of the tag value, a freely definable alarm text is generated.

Declaration

```

BOOL DMSetValueMessage (
    LPDM_VARKEY    lpdmVarKey,
    LPVARIANT      lpdmValue,
    DWORD          fFlags,
    LPSTR          lpszMessage,
    LPCMN_ERROR    lpdmError);

```

Parameters**lpdmVarKey**

Pointer to the `DM_VARKEY` (Page 1590) structure that identifies the tag to be changed.

A server prefix (see project types and versions) may also be specified in `lpdmVarKey->szName` for multiclient projects as of WinCC Version 5.0.

lpdmValue

Pointer to the new value of the tag to be changed.

fFlags

fFlags determines how the alarm text is processed:

DMSVM_OPERATIONREPORT	The alarm is stored in the operator input log.
DMSVM_OPERATIONMESSAGE	The alarm is output as an operator input alarm.

lpzMessage

Text of the alarm to be transmitted.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tag changed.

FALSE

Error.

Comment

You can find more information about changing tag values at the DMSetValue function.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DMSetValue (Page 1730)	Change tag values
DMSetValueWait (Page 1742)	Change tag values with notification
DMSetValueWaitMessage (Page 1747)	Change tag values with notification and alarm output

See also

DM_VARKEY (Page 1590)

DMSetValue (Page 1730)

DMSetValueWait (Page 1742)

DMSetValueWaitMessage (Page 1747)

3.2.6.25 DMSetValueMessageExStr (RT Professional)**Declaration**

```

BOOL DMSetValueMessageExStr (
    LPVARIANT      lpvdmVarKey,
    LPVARIANT      lpvdmValue,
    DWORD          fFlags,
    LPTSTR         lpszMessage,
    LPCMN_ERROR    lpdmError);

```

Description

This function changes the value of the tag described by `lpvdmVarKey` to obtain the value specified in `lpdmValue`.

In contrast to the `DMSetValueExStr` function, only one tag can be changed at a time. If the tag value has been changed successfully, a alarm text of your choice is transmitted.

Parameters**`lpvdmVarKey`**

Pointer to VARIANT for passing on the tag name or TagID or single VARIANT, if only one tag is specified that identifies the tags to be changed.

A TagID is entered in the relevant list element with the type `VT_I4` and the tag name with `VT_BSTR`. In addition, `VT_LPSTR` (`PROPVARIANT`) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. `VT_LPSTR` is then converted internally into the required `VT_BSTR` type.

If you specify an array using `VT_ARRAY | VT_VARIANT`, only the first element is evaluated.

For a multi-client project in WinCC Version 5.0 or higher, it may be required in this case to add a server prefix to each name (refer to "Project types and versions").

`lpvdmValue`

Pointer to the new value of the tag to be changed.

fFlags

fFlags specifies how to process the alarm text:

DMSVM_OPERATIONREPORT	The alarm is stored in the operator input log.
DMSVM_OPERATIONMESSAGE	The alarm is output as an operator input alarm.

lpzMessage

Text of the alarm to be transmitted.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tags have been changed

FALSE

Error

Comment

For more information on changing the values of tags refer to the DMSetValueExStr function.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMSetValueExStr	Change tag values
DMSetValueWaitExStr	Change tag values with notification
DMSetValueWaitMessageExStr	Change tag values with notification and alarm output

3.2.6.26 DMSetValueWait (RT Professional)

Use

The function changes the values of the tags described by `lpdmVarKey` to the values specified in `lpdmValue`.

In contrast to `DMSetValue`, `DMSetValueWait` allows the application to be notified when the update is successfully completed on the local machine.

Declaration

```

BOOL DMSetValueWait (
    LPDWORD                pdwTAID,
    LPDM_VARKEY            lpdmVarKey,
    DWORD                  dwItems,
    LPVARIANT              lpdmValue,
    DWORD                  dwTimeOut,
    DM_COMPLETION_PROC     lpfnCompletion,
    LPVOID                 lpvUser,
    LPCMN_ERROR            lpdmError);

```

Parameters

pdwTAID

Pointer to a tag that contains the transaction ID assigned by the data manager after a successful call of the function.

lpdmVarKey

Pointer to the first `DM_VARKEY` (Page 1590) structure that identifies the tag values to be changed.

dwItems

Number of passed structures (corresponds to the number of tag values to be changed).

lpdmValue

Pointer to the first of the new values for the tags to be changed.

dwTimeout

Maximum waiting time of the application in ms. If, after expiration of the waiting period, not all tags have been written, the callback function is called with appropriate error code.

lpfnCompletion

Pointer to your callback function, which is called either after the update of all requested tags or after the waiting period.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

IpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the first of dwItems error structure with the CMN_ERROR type. When an error occurs during the writing of a tag, the system writes the error information into the appropriate structure. Therefore, remember to reserve space for these structures.

Return value

TRUE

Tags changed.

FALSE

The error which occurred can be identified by means of the error structures.

Comment

You can find more information about changing tag values at the DMSetValue function.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DMSetValue (Page 1730)	Change tag values
DMSetValueMessage (Page 1736)	Change tag values with alarm output
DMSetValueWaitMessage (Page 1747)	Change tag values with notification and alarm output
DM_COMPLETION_PROC (Page 1752)	Change tag values (callback)

Examples

OnTestVariablenSetvaluawait (Page 1858) "TESTCDoc.cpp"

See also

DM_VARKEY (Page 1590)

DMGetValueWait (Page 1700)

DMGetValueWaitEx (Page 1702)

DMSetValue (Page 1730)

DMSetValueMessage (Page 1736)

DMSetValueWaitMessage (Page 1747)

DM_COMPLETITION_PROC (Page 1752)

OnTestVariablenSetvaluawait (Page 1858)

3.2.6.27 DMSetValueWaitExStr (RT Professional)

Declaration (

```

BOOL DMSetValueWaitExStr (
    LPDWORD          pdwTAID,
    LPVARIANT         lpvdmVarKey,
    DWORD            dwItems,
    LPVARIANT         lpvdmValue,
    LPDWORD          lpdwState,
    DWORD            dwTimeOut,
    DM_COMPLETITION_PROC lpfnCompetition,
    LPVOID            lpvUser,
    LPCMN_ERROR       lpdmError);

```

Description

This function changes the values of the tags described by lpvdmVarKey to obtain the values specified in lpvdmValue.

In contrast to the DMSetValueExStr function, DMSetValueWaitExStr enables your application to display a notify message on the local computer if the update has been executed.

In contrast to DMSetValueWait, the tag list is passed on as pointer to VARIANT with VT_ARRAY | VT_VARIANT. As a result, no length limit applies to tag names.

Parameters

pdwTAID

Pointer to a tag that contains the transaction ID assigned by the data manager after a successful call of the function.

lpvdmVarKey

Pointer to VARIANT with VT_ARRAY | VT_VARIANT for passing on the tag list or single VARIANT if only one tag is specified.

A TagID is entered in the relevant list element with the type VT_I4 and the tag name with VT_BSTR. In addition, VT_LPSTR (PROPVARIANT) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. VT_LPSTR is then converted internally into the required VT_BSTR type.

For multi-client projects in WinCC Version 5.0 or higher, it may be required here to add a server prefix (refer to "Project types and versions").

dwItems

Number of values passed in lpvdmValue and error structures in lpdmError. The number must be at least as large as the input array in lpvdmVarKey.

If the number is larger than the input array, the excess values are ignored and DM_E_PARAM is assigned to the affected error structures, as no allocation is possible.

If the number is less than the input array, the error DM_E_PARAM is returned only in the first error structure and no values are written.

lpvdmValue

Pointer to the first of the new values for the tags to be changed.

lpdwState

Pointer to a memory location which stores the information whether the value of the tag could be changed successfully or if errors occurred.

The value 0 (OK) means that the transmission/modification has been completed successfully and that at least one of the specified tags exists. The DM_VARSTATE_INVALID_KEY error status is only returned if none of the specified tags exists.

DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	Connection to partner not established
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	Protocol error
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	Network module faulty
DM_VARSTATE_MAX_LIMIT	(0x0008)	Configured high limit violated
DM_VARSTATE_MIN_LIMIT	(0x0010)	Configured low limit violated
DM_VARSTATE_MAX_RANGE	(0x0020)	High format limit violated
DM_VARSTATE_MIN_RANGE	(0x0040)	Low format limit violated
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	Conversion error display (in connection with DM_VARSTATE_..._RANGE)
DM_VARSTATE_STARTUP_VALUE	(0x0100)	Initialization value of tag
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	Substitute value of tag
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	Addressing error in the channel

3.2 Data management functions (RT Professional)

DM_VARSTATE_INVALID_KEY	(0x0800)	Tag not found / not available
DM_VARSTATE_ACCESS_FAULT	(0x1000)	Access to tag not permitted
DM_VARSTATE_TIMEOUT	(0x2000)	Timeout / no feedback from channel
DM_VARSTATE_SERVERDOWN	(0x4000)	Server is down

dwTimeout

Maximum waiting time of the application in ms. If not all tag values have been updated when the waiting time expires, the callback function with corresponding error codes is called.

lpfnCompletion

Pointer to the callback function which is called either after all requested tags have been updated or after the waiting time has expired.

If a program requests a notify routine, it has to clear its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

On rare occasions, the notify may be returned before the function call has returned.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the callback function.

lpdmError

Pointer to the first dwltems error structure of the CMN_ERROR type. If a tag writing error occurs, the system writes the error information into the corresponding structure. Therefore, you should not forget to reserve space for these structures.

Return value**TRUE**

Tags have been changed

FALSE

The error which occurred can be identified by means of the error structures.

Comment

For more information on changing the values of tags refer to the DMSetValueExStr function.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMSetValueExStr	Change tag values
DMSetValueMessageExStr	Change tag values with alarm output
DMSetValueWaitMessageExStr	Change tag values with notification and alarm output
DM_COMPLETITION_PROC	Change tag values (callback)

Examples

You can find samples on the DMGetValueExStr page.

3.2.6.28 DMSetValueWaitMessage (RT Professional)**Use**

The function combines the functions of DMSetValueMessage and DMSetValueWait. Only one tag can be changed, in contrast to the DMSetValueWait function.

The value of the tag described by lpdmVarKey is changed to the value specified in lpdmValue. Once the tag is successfully changed, a freely definable alarm text is output and the application can be notified that the update has been successfully performed on the local machine.

Declaration

```

BOOL DMSetValueWaitMessage (
    LPDWORD                pdwTAID,
    LPDPM_VARKEY           lpdmVarKey,
    LPVARIANT              lpdmValue,
    DWORD                  dwTimeOut,
    DM_COMPLETITION_PROC   lpfnCompletion,
    DWORD                  fFlags,
    LPSTR                  lpszMessage,
    LPVOID                 lpvUser,
    LPCMN_ERROR             lpdmError);

```

Parameters**pdwTAID**

Pointer to a tag that contains the transaction ID assigned by the data manager after a successful call of the function.

lpdmVarKey

Pointer to the DM_VARKEY (Page 1590) structure that identifies the tag to be changed.

lpdmValue

Pointer to the new value of the tag to be changed.

dwTimeOut

Maximum waiting time of the application in ms. If, after expiration of the waiting period, not all tags have been written, the callback function is called with appropriate error code.

lpfnCompletion

Pointer to your callback function, which is called either after the update of all requested tags or after the waiting period.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

fFlags

fFlags determines how the alarm text is processed:

DMSVM_OPERATIONREPORT	The alarm is stored in the operator input log.
DMSVM_OPERATIONMESSAGE	The alarm is output as an operator input alarm.

lpaszMessage

Alarm text that is output.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the first of dwItems error structure with the CMN_ERROR type. When an error occurs during the writing of a tag, the system writes the error information into the appropriate structure. Therefore, remember to reserve space for these structures.

Return value**TRUE**

Tags changed.

FALSE

Error.

Comment

You can find more information about changing tag values at the DMSetValue function.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DM_COMPLETION_PROC (Page 1752)	Change tag values (callback)
DMSetValue (Page 1730)	Change tag values
DMSetValueMessage (Page 1736)	Change tag values with alarm output
DMSetValueWait (Page 1740)	Change tag values with notification

See also

DM_VARKEY (Page 1590)
DMSetValue (Page 1730)
DMSetValueMessage (Page 1736)
DMSetValueWait (Page 1740)
OnTestVariablenSetvaluwait (Page 1858)
DM_COMPLETION_PROC (Page 1752)

3.2.6.29 DMSetValueWaitMessageExStr (RT Professional)**Declaration**

```

BOOL DMSetValueWaitMessageExStr (
    LPDWORD                pdwTAID,
    LPVARIANT              lpvdmVarKey,
    LPVARIANT              lpvdmValue,
    DWORD                  dwTimeOut,
    DM_COMPLETION_PROC     lpfnCompletion,
    DWORD                  fFlags,
    LPTSTR                 lpszMessage,
    LPVOID                 lpvUser,
    LPCMN_ERROR             lpdmError);

```

Description

This function combines the functionalities of `DMSetValueMessageExStr` and `DMSetValueWaitExStr`.

In contrast to the `DMSetValueWaitExStr` function, only one tag can be changed at a time.

The value of the tag described by `lpvdmVarKey` is changed to obtain the value specified in `lpvdmValue`. If the tag value has been changed successfully, a alarm text of your choice is transmitted. The application also has the option to display a notify message on the local computer if the update has been executed.

In contrast to `DMSetValueWaitMessage`, the tag list is passed on as pointer to `VARIANT`. As a result, no limitation applies to the length of the tag names.

Parameters

pdwTAID

Pointer to a tag that contains the transaction ID assigned by the data manager after a successful call of the function.

lpvdmVarKey

Pointer to `VARIANT` for passing on the tag name, TagID or single `VARIANT`, if only one tag is specified that identifies the tags to be changed.

A TagID is entered in the relevant list element with the type `VT_I4` and the tag name with `VT_BSTR`. In addition, `VT_LPSTR` (`PROPVARIANT`) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. `VT_LPSTR` is then converted internally into the required `VT_BSTR` type.

For multi-client projects in WinCC Version 5.0 or higher, it may be required here to add a server prefix (refer to "Project types and versions").

lpvdmValue

Pointer to the new value of the tag to be changed.

dwTimeOut

Maximum waiting time of the application in ms. If not all tag values have been updated when the waiting time expires, the callback function with corresponding error codes is called.

lpfnCompletion

Pointer to the callback function which is called either after all requested tags have been updated or after the waiting time has expired.

If a program requests a notify routine, it has to clear its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

On rare occasions, the notify may be returned before the function call has returned.

fFlags

fFlags specifies how to process the alarm text:

DMSVM_OPERATIONREPORT	The alarm is stored in the operator input log.
DMSVM_OPERATIONMESSAGE	The alarm is output as an operator input alarm.

lpszMessage

Text of the alarm to be transmitted.

lppvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the first dwItems error structure of the CMN_ERROR type. If a tag writing error occurs, the system writes the error information into the corresponding structure. Therefore, remember to reserve space for these structures.

Return value**TRUE**

Tags have been changed

FALSE

Error

Comment

For more information on changing the values of tags refer to the DMSetValueExStr function.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_PARAM	Invalid parameter
DM_E_OOM	Out of memory

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DM_COMPLETION_PROC	Change tag values (callback)
DMSetValueExStr	Change tag values
DMSetValueMessageExStr	Change tag values with alarm output
DMSetValueWaitExStr	Change tag values with notification

3.2.6.30 DM_COMPLETION_PROC (RT Professional)**Description**

You need to provide a DM_COMPLETION_PROC type callback function in order for your application to be notified that tag values have been successfully changed.

Declaration

```

BOOL ( * DM_COMPLETION_PROC) (
    DWORD      dwTAID,
    LPDWORD    lpdmVarState,
    DWORD      dwItems,
    LPVOID     lpvUser);

```

Parameters**dwTAID**

Transaction ID that was assigned by the data manager for the function to change the tag values.

lpdmVarState

Pointer to the first memory location where information is stored indicating whether the tag values could be changed successfully or whether errors have occurred.

A 0 (OK) means successfully sent/changed and at least one of the specified tag exists. An error with the DM_VARSTATE_INVALID_KEY status only occurs if none of the specified tags exist.

DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	Connection to partner not established
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	Protocol error
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	Network module faulty
DM_VARSTATE_MAX_LIMIT	(0x0008)	Configured high limit violated
DM_VARSTATE_MIN_LIMIT	(0x0010)	Configured low limit violated
DM_VARSTATE_MAX_RANGE	(0x0020)	High format limit violated
DM_VARSTATE_MIN_RANGE	(0x0040)	Low format limit violated
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	Display conversion error (in conjunction with DM_VARSTATE_..._RANGE)
DM_VARSTATE_STARTUP_VALUE	(0x0100)	Initialization value of tag

DM_VARSTATE_DEFAULT_VALUE	(0x0200)	Substitute value of tag
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	Addressing error in the channel
DM_VARSTATE_INVALID_KEY	(0x0800)	Tag not found / not available
DM_VARSTATE_ACCESS_FAULT	(0x1000)	Access to tag not permitted
DM_VARSTATE_TIMEOUT	(0x2000)	Timeout / no feedback from channel
DM_VARSTATE_SERVERDOWN	(0x4000)	Server is down

dwItems

Number of tag changes, the status of which is passed in `lpdmVarState`.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

The return values depend on your implementation

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: `SendMessage`
- API functions from the same DLL
- Enumerations that call other enumerations

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

Required files

`dmclient.h`

Related functions

DMSetValueWait (Page 1740)	Change tag values with notification
DMSetValueWaitMessage (Page 1745)	Change tag values with notification and alarm output

See also

[DMSetValueWait \(Page 1740\)](#)

[DMSetValueWaitMessage \(Page 1745\)](#)

3.2.6.31 DMShowVarPropertiesExStr (RT Professional)

Declaration

```
BOOL DMShowVarPropertiesExStr (  
    LPCTSTR      lpszProjectFile,  
    HWND         hwndParent,  
    LPCTSTR      lpszVariableName,  
    DWORD        dwVariableID,  
    LPCMN_ERROR  lpdmError);
```

Description

This function opens the dialog box for editing the properties of tags.

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMEnumOpenedProjects, or with DMGetRuntimeProject in Runtime.

hwndParent

Handle to the window that is used as the parent window for the dialog.

lpszVariableName

Pointer to the name of the tag whose properties are displayed.

If lpszVariableName is "NULL" you must specify a valid dwVariableID. If not, the DM_E_PARAM error is returned.

dwVariableID

ID of the tag. Is not used if a name is specified in lpszVariableName.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Close the dialog with "OK".

FALSE

Error or close dialog with "Cancel"

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_KEY	Tag not found
DM_E_CANCEL	User has selected "Cancel" in a dialog
DM_E_PARAM	At least one parameter is passed on with errors

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Examples**Script sample Button DMShowVarDatabaseExStr:**

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
    BOOL bRet = FALSE;
    CHAR szProjectName[256];
    CHAR szVariableName[256];
    DWORD dwVariableID;
    CMN_ERROR err;
    HWND hwndParent;

    szProjectName[0] = 0;
    szVariableName[0] = 0;
    dwVariableID = 0L;
    memset(&err, 0, sizeof(err));
    hwndParent = NULL;

    printf("\r\n\r\n##### enter Test with DMShowVarDatabaseExStr
#####");

    bRet = DMGetRuntimeProject(szProjectName, 256, &err);
    if (!bRet)
    {
        printf("\r\n  error DMGetRuntimeProject =>[%s], err=%ld,%ld,%ld,
%ld,%ld,[%s]", szProjectName,
            err.dwError1, err.dwError2, err.dwError3, err.dwError4,
            err.dwError5, err.szErrorText);
    }
}
```

```

memset(&err, 0, sizeof(err));

dwVariableID = GetTagDWord("dwVarKeyID_1");
printf("\r\n  call DMSHOWVarPropertiesExStr with ID");
bRet = DMSHOWVarPropertiesExStr(szProjectName, hwndParent, NULL,
dwVariableID, &err);
if (!bRet)
{
    printf("\r\n  error DMSHOWVarPropertiesExStr: err=%ld,%ld,%ld,
%ld,%ld,[%s]",
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMSHOWVarPropertiesExStr OK!");
}

strncpy(szVariableName, GetTagChar("szVarKeyName_1"), 255);
szVariableName[255] = 0;

printf("\r\n  call DMSHOWVarPropertiesExStr with VarName");
bRet = DMSHOWVarPropertiesExStr(szProjectName, hwndParent,
szVariableName, 0L, &err);
if (!bRet)
{
    printf("\r\n  error DMSHOWVarPropertiesExStr: err=%ld,%ld,%ld,
%ld,%ld,[%s]",
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMSHOWVarPropertiesExStr OK!");
}

printf("\r\n##### exit Test with DMSHOWVarPropertiesExStr
#####\r\n");
}

```

3.2.6.32 DMSHOWVarDatabase (RT Professional)

Use

This function opens the tag selection dialog for the project specified. Only one tag is selected, in contrast to the ShowVarDatabaseMulti function.

Declaration

```

BOOL DMSHOWVarDatabase (
    LPCSTR          lpszProjectFile,
    HWND            hwndParent,
    LPDM_DLGOPTIONS lpdmOptions,
    LPDM_VARFILTER  lpdmFilter,
    LPDM_VARKEY     lpdmVarKey,
    LPCMN_ERROR     lpdmError);

```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

hwndParent

Handle to the window that is used as the parent window for the dialog.

lpdmOptions

Pointer to the DM_DLGOPTIONS (Page 1568) structure with specifications on how the dialog should react. When the pointer is NULL, the standard dialog is selected.

lpdmFilter

Pointer to the DM_VARFILTER (Page 1582) filter structure. When the pointer is NULL, all tags are displayed.

Only filtering by tag name and tag type is supported. Tag group and tag connection filtering is not implemented.

lpdmVarKey

Pointer to the DM_VARKEY (Page 1590) structure. If the properties of a tag to be displayed, lpdmVarKey contains the key to this tag after the dialog closed.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Close the dialog with "OK".

FALSE

Error or close dialog with "Cancel".

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INV_PRJ	Specified project not found / loaded

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DMSHOWVARDATABASEMULTI (Page 1761)	Open tag selection dialog

See also

DM_VARKEY (Page 1590)
DM_VARFILTER (Page 1582)
DM_DLGOPTIONS (Page 1568)
DMSHOWVARDATABASEMULTI (Page 1761)

3.2.6.33 DMSHOWVARDATABASEEXSTR (RT Professional)**Declaration**

```

BOOL DMSHOWVARDATABASEEXSTR (
    LPCTSTR          lpszProjectFile,
    HWND             hwndParent,
    LPDM_DLGOPTIONS  lpdmOptions,
    LPDM_VARFILTER   lpdmFilter,
    LPTSTR*          lppszVariableName,
    LPDWORD           lpdwVarNameCharCount,
    LPDWORD           lpdwVariableID,
    LPCMN_ERROR       lpdmError);

```

Description

This function opens the tag selection dialog for the project specified. In contrast to ShowVarDatabaseMultiExStr, only one tag can be selected.

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMEnumOpenedProjects, or with DMGetRuntimeProject in Runtime.

hwndParent

Handle to the window that is used as the parent window for the dialog.

lpdmOptions

Pointer to the DM_DLGOPTIONS structure containing instructions for the dialog; if NULL => standard dialog.

lpdmFilter

Pointer to the DM_VARFILTER filter structure. If NULL, all tags are displayed.

Only tag name and tag type filtering are supported. Tag group and tag connection filtering is not implemented.

lpszVariableName

Pointer to pointer of the return buffer for the tag name.

If NULL, a valid pointer to lpdwVariableID must be present. Then, only the ID and no name is returned. "dwVarNameCharCount" is not considered. If both pointers are NULL, DM_E_PARAM is returned.

If the internal pointer is NULL, the buffer is allocated and returned. The size is returned in lpdwVarNameCharCount.

lpdwVarNameCharCount

Pointer to a DWORD which stores the size of the return buffer. The size must be selected big enough to store the tag name with zero termination.

If the buffer is too small, the truncated name is stored and the DM_E_OOM error returned.

lpdwVariableID

Pointer to a DWORD which stores the tag ID.

If NULL, a valid pointer to lpdwVariableName and a valid size in dwVarNameCharCount must be present. Then, only the tag name will be returned.

Return of the tag ID cannot always be guaranteed, e.g. PackageTag, specific S7 tags etc. In these cases, 0 is returned.

Under certain conditions, the information can be complemented by a subsequent call of DMGetVarInfoExStr.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Close the dialog with "OK".

FALSE

Error or close dialog with "Cancel".

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INV_PRJ	Specified project not found / loaded
DM_E_PARAM	Parameter not consistent
DM_E_OOM	Memory error: the buffer lpszVariableName is specified too small in dwVarNameCharCount and the name was truncated

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMSHOWVARDATABASEMULTIEXSTR	Open tag selection dialog

Examples**Script sample Button DMSHOWVARDATABASEEXSTR:**

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
    BOOL bRet = FALSE;
    CMN_ERROR err;
    CHAR szProjectName[256];
    HWND hwndParent;
    CHAR szVariableName[256];
    LPSTR pszVariableName;
    DWORD dwVarNamCharCount = 256;
    DWORD dwVarID = 0L;

    szProjectName[0] = 0;
    szVariableName[0] = 0;
    hwndParent = NULL;
    pszVariableName= szVariableName;
```

```

memset(&err, 0, sizeof(err));

printf("\r\n\r\n##### enter Test with DMShowVarDatabaseExStr
#####");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s], err=%ld,%ld,%ld,
%ld,%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

bRet = DMShowVarDatabaseExStr(szProjectName, hwndParent, NULL, NULL,
        &pszVariableName, &dwVarNamCharCount,
&dwVarID, &err);
if (!bRet)
{
    printf("\r\n  error DMShowVarDatabaseExStr: err=%ld,%ld,%ld,%ld,
%ld,[%s]",
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMShowVarDatabaseEsStr OK!  (set to first line)");
    printf("\r\n  szVarName=[%s], dwVarID=[%ld]", szVariableName,
dwVarID);
    SetTagChar("szVarKeyName_1",szVariableName);
    SetTagDWord("dwVarKeyID_1",dwVarID);
}
}

```

3.2.6.34 DMShowVarDatabaseMulti (RT Professional)

Use

This function opens the tag selection dialog for the project specified. Several tags can be selected, in contrast to ShowVarDatabase.

Declaration

```

BOOL DMShowVarDatabaseMulti (
    LPCSTR          lpszProjectFile,
    HWND            hwndParent,
    LPDM_DLGOPTIONS lpdmOptions,
    LPDM_VARFILTER  lpdmFilter,
    LPDWORD         lpdwItems,
    DM_NOTIFY_SELECT_VAR_PROC lpfnVariables,
    LPVOID          lpvUser,
    LPCMN_ERROR     lpdmError);

```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in RT.

hwndParent

Handle to the window that is used as the parent window for the dialog.

lpdmOptions

Pointer to the DM_DLGOPTIONS (Page 1568) structure with specifications on how the dialog should react. The standard dialog is used with NULL.

lpdmFilter

Pointer to the DM_VARFILTER (Page 1582) filter structure. All tags are displayed with NULL,.

Only filtering by tag name and tag type is supported. Tag group and tag connection filtering is not implemented.

lpdwItems

Pointer to a DWORD buffer that receives the number of selected tags.

lpfnVariables

Pointer to your callback function that is called for every selected tag.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Close the dialog with "OK".

FALSE

Error or close dialog with "Cancel".

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INV_PRJ	Specified project not found / loaded

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DMSHOWVARDATABASE (Page 1754)	Open tag selection dialog
DM_NOTIFY_SELECT_VAR_PROC (Page 1767)	Open tag selection dialog (callback)

See also

DM_VARFILTER (Page 1582)
DMSHOWVARDATABASE (Page 1754)
DM_DLGOPTIONS (Page 1568)
DM_NOTIFY_SELECT_VAR_PROC (Page 1767)

3.2.6.35 DMShowVarDatabaseMultiExStr (RT Professional)

Declaration

```

BOOL DMShowVarDatabaseMultiExStr (
    LPCTSTR                lpszProjectFile,
    HWND                   hwndParent,
    LPDM_DLGOPTIONS         lpdmOptions,
    LPDM_VARFILTER          lpdmFilter,
    LPDWORD                 lpdwItems,
    DM_NOTIFY_SELECT_VAR_PROC_EXSTR lpfnVariables,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);

```

Description

This function opens the tag selection dialog for the project specified. In contrast to ShowVarDatabaseExStr, multiple tags can be selected.

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMEnumOpenedProjects, or with DMGetRuntimeProject in Runtime.

hwndParent

Handle to the window that is used as the parent window for the dialog.

lpdmOptions

Pointer to the DM_DLGOPTIONS structure containing instructions for the dialog; if NULL => standard dialog.

lpdmFilter

Pointer to the DM_VARFILTER filter structure. If NULL, all tags are displayed.

Only tag name and tag type filtering are supported. Tag group and tag connection filtering is not implemented.

lpdwItems

Pointer to a DWORD buffer to store the number of tags selected in total.

lpfnVariables

Pointer to the callback function which is called for each selected tag.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the callback function.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Close the dialog with "OK".

FALSE

Error or close dialog with "Cancel".

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INV_PRJ	Specified project not found / loaded

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMSHOWVARDATABASEEXSTR	Open tag selection dialog
DM_NOTIFY_SELECT_VAR_PROC_EXSTR	Open tag selection dialog (callback)

Examples**Script sample Button DMSHOWVARDATABASEMULTIEXSTR:**

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
extern BOOL  DM_NotifySelectVarProcA(LPCSTR lpszVariableName, DWORD
dwVariableID, LPVOID lpvUser);

BOOL bRet = FALSE;
CMN_ERROR err;
CHAR szProjectName[256];
```

3.2 Data management functions (RT Professional)

```

HWND hwndParent;
DWORD dwItems;
DWORD dwInxDecr;

szProjectName[0] = 0;
hwndParent = NULL;
memset(&err, 0, sizeof(err));
dwItems = 0;
dwInxDecr = 4; /*for decrement index to save in DM tags from callback
set to lpvUser*/

printf("\r\n\r\n##### enter Test with
DMSHOWVarDatabaseMultiExStr #####");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s], err=%ld,%ld,%ld,
%ld,%ld,[%s]", szProjectName,
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

bRet = DMSHOWVarDatabaseMultiExStr(szProjectName, hwndParent, NULL,
NULL,
                                &dwItems, DM_NotifySelectVarProcA,
&dwInxDecr, &err);
if (!bRet)
{
    printf("\r\n  error DMSHOWVarDatabaseMultiExStr: err=%ld,%ld,%ld,
%ld,%ld,[%s]",
        err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMSHOWVarDatabaseExStrMulti OK!");
    printf("\r\n  [%ld] Var's selected", dwItems);
}

printf("\r\n##### exit Test with DMSHOWVarDatabaseMultiExStr
#####\r\n");
}

```

Script sample Project-Function DM_NotifySelectVarProcA:

```

BOOL  DM_NotifySelectVarProcA(LPCSTR lpszVariableName, DWORD
dwVariableID, LPVOID lpvUser)
{
    BOOL bRet = TRUE;
    DWORD* pdwInxDecr = NULL;

```

```

pdwInxDcr = (DWORD*)lpvUser;

printf("\r\n\r\n#### DM_NotifySelectVarProc entry: ####");
printf("\r\n    VarName=[%s], VarID=[%ld], lpvUser=[0x%08lx]",
    lpzVariableName, dwVariableID, lpvUser);
if (pdwInxDcr)
{
    // handle max 4 outputs, ignore all others
    if(4 == *pdwInxDcr )
    {
        SetTagChar("szVarKeyName_4",lpzVariableName);
        SetTagDWord("dwVarKeyID_4",dwVariableID);
        printf("\r\n        save in szVarKeyName_4 and dwVarKeyID_4");
    }
    if(3 == *pdwInxDcr )
    {
        SetTagChar("szVarKeyName_3",lpzVariableName);
        SetTagDWord("dwVarKeyID_3",dwVariableID);
        printf("\r\n        save in szVarKeyName_3 and dwVarKeyID_3");
    }
    if(2 == *pdwInxDcr )
    {
        SetTagChar("szVarKeyName_2",lpzVariableName);
        SetTagDWord("dwVarKeyID_2",dwVariableID);
        printf("\r\n        save in szVarKeyName_2 and dwVarKeyID_2");
    }
    if(1 == *pdwInxDcr )
    {
        SetTagChar("szVarKeyName_1",lpzVariableName);
        SetTagDWord("dwVarKeyID_1",dwVariableID);
        printf("\r\n        save in szVarKeyName_1 and dwVarKeyID_1");
    }
}
if (pdwInxDcr && (0L < *pdwInxDcr))
{
    *pdwInxDcr = *pdwInxDcr - 1L;
    printf("\r\n        Inx=%ld", *pdwInxDcr);
}

printf("\r\n#### DM_NotifySelectVarProc exit ####\r\n");

return bRet;
}

```

3.2.6.36 DM_NOTIFY_SELECT_VAR_PROC (RT Professional)

Description

In order to evaluate the tags selected by the `DMSHOWVARDATABASEMULTI` function, you must provide a `DM_NOTIFY_SELECT_VAR_PROC` type callback function.

Declaration

```
BOOL ( * DM_NOTIFY_SELECT_VAR_PROC) (
    LPDM_VARKEY    lpdmVarKey,
    DWORD          dwItem,
    LPVOID          lpvUser );
```

Parameters

- lpdmVarKey**
Pointer to first DM_VARKEY (Page 1590) type structure with the name and ID of a tag.
- dwItem**
Number of structures passed in lpdmVarKey.
- lpvUser**
Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

- TRUE**
The enumeration is continued.
- FALSE**
The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

dmclient.h

Related functions

DMShowVarDatabaseMulti (Page 1759)	Open tag selection dialog

See also

DM_VARKEY (Page 1590)

DMSHOWVARDATABASEMULTI (Page 1759)

3.2.6.37 GAPICreateNewVariable (RT Professional)**Use**

Creates a new tag or check whether a tag exists. This function can only be used for temporary \$ tags.

Declaration

```
BOOL GAPICreateNewVariable (
    LPMCP_NEWVARIABLE_DATA    pData,
    LPCMN_ERROR                lpdmError);
```

Parameters**pData**

Pointer to the MCP_NEWVARIABLE_DATA (Page 1594) structure with the data of the tag.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tag created.

When checked: Tag does not yet exist.

FALSE

Error.

When checked with ErrorCode1 = DM_E_ALREADY_EXIST: Tag exists

Comment

There are more advanced functions.

GAPICreateNewVariable4
GAPICreateNewVariableEx4
GAPICreateNewVariable5

Error messages

DM_E_NOT_CONNECTED	No connection to the Data Manager
DM_E_ALREADY_EXIST	Object to be created already exists

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

GAPICreateNewVariable4 (Page 1770)	Create tag
GAPICreateNewVariableEx4 (Page 1773)	Create tag with Creator ID
GAPICreateNewVariable5 (Page 1772)	Create tag with Creator ID

See also

MCP_NEWVARIABLE_DATA (Page 1594)

GAPICreateNewVariable4 (Page 1770)

GAPICreateNewVariable5 (Page 1772)

GAPICreateNewVariableEx4 (Page 1773)

3.2.6.38 GAPICreateNewVariable4 (RT Professional)**Use**

Creates a new tag or check whether a tag exists. This function can only be used for temporary \$ tags.

This function differs from GAPICreateNewVariable due to the additional specification of scaling information.

Declaration

```

BOOL WINAPI CreateNewVariable4 (
    LPMCP_NEWVARIABLE_DATA_4    pData,
    LPCMN_ERROR                  lpdmError);

```

Parameters

pData

Pointer to the MCP_NEWVARIABLE_DATA_4 (Page 1596) structure with the data of the tag.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Tag created.

When checked: Tag does not yet exist.

FALSE

Error.

When checked with Errorcode1 = DM_E_ALREADY_EXIST: Tag exists.

Comment

There are more advanced functions.

GAPICreateNewVariableEx4	Create tag with Creator ID
GAPICreateNewVariable5	Create tag with Creator ID

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_ALREADY_EXIST	Object to be created already exists

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

GAPICreateNewVariable (Page 1767)	Create tag
GAPICreateNewVariableEx4 (Page 1773)	Create tag with Creator ID
GAPICreateNewVariable5 (Page 1772)	Create tag with Creator ID

See also

[GAPICreateNewVariable \(Page 1767\)](#)
[MCP_NEWVARIABLE_DATA_4 \(Page 1596\)](#)
[GAPICreateNewVariable5 \(Page 1772\)](#)
[GAPICreateNewVariableEx4 \(Page 1773\)](#)

3.2.6.39 GAPICreateNewVariable5 (RT Professional)

Use

Creates a new tag or check whether a tag exists. This function can only be used for temporary \$ tags.

This function differs from GAPICreateNewVariable4 due to the specification of the creator ID and the start and substitute values that can also be specified for text tags.

Declaration

```

BOOL GAPICreateNewVariable5 (
    DWORD                dwCreatorID,
    LPMCP_NEWVARIABLE_DATA_5  pData,
    LPCMN_ERROR           lpdmError);

```

Parameters

dwCreatorID

In the creator identification, you can determine who created an object.

The values 0 – 10100 and 11000 – 11100 are reserved for internal or specific systems.

pData

Pointer to the MCP_NEWVARIABLE_DATA_5 (Page 1598) structure with the data of the tag.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tag created.

When checked: Tag does not yet exist.

FALSE

Error.

When checked with Errorcode1 = DM_E_ALREADY_EXIST: Tag exists.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_ALREADY_EXIST	Object to be created already exists

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

GAPICreateNewVariable (Page 1767)	Create tag
GAPICreateNewVariable4 (Page 1768)	Create tag
GAPICreateNewVariableEx4 (Page 1773)	Create tag with Creator ID

See also

GAPICreateNewVariable (Page 1767)

GAPICreateNewVariable4 (Page 1768)

MCP_NEWVARIABLE_DATA_5 (Page 1598)

GAPICreateNewVariableEx4 (Page 1773)

MCP_VARIABLE_LIMITS5 (Page 1610)

3.2.6.40 GAPICreateNewVariableEx4 (RT Professional)**Use**

Creates a new tag or check whether a tag exists. This function can only be used for temporary \$ tags.

This function differs from GAPICreateNewVariable4 due to the specification of the creator ID.

Declaration

```
BOOL GAPICreateNewVariableEx4 (
    DWORD                dwCreatorID,
    LPMCP_NEWVARIABLE_DATA_4  pData,
    LPCMN_ERROR           lpdmError);
```

Parameters

dwCreatorID

In the creator identification, you can determine who created an object.

The values 0 – 10100 and 11000 – 11100 are reserved for internal or specific systems.

pData

Pointer to the MCP_NEWVARIABLE_DATA_4 (Page 1596) structure with the data of the tag.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Tag created.

When checked: Tag does not yet exist.

FALSE

Error.

When checked with Errorcode1 = DM_E_ALREADY_EXIST: Tag exists.

Comment

There is an advanced function.

GAPICreateNewVariable5

Error messages

DM_E_NOT_CONNECTED	No connection to the Data Manager
DM_E_ALREADY_EXIST	Object to be created already exists

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

GAPICreateNewVariable (Page 1767)	Create tag
GAPICreateNewVariable4 (Page 1768)	Create tag
GAPICreateNewVariable5 (Page 1770)	Create tag with Creator ID

See also

GAPICreateNewVariable (Page 1767)

GAPICreateNewVariable4 (Page 1768)

MCP_NEWVARIABLE_DATA_4 (Page 1596)

GAPICreateNewVariable5 (Page 1770)

3.2.7 Functions for processing structured tags (RT Professional)**3.2.7.1 GAPIEnumTypeMembers (RT Professional)****Use**

The function returns the names of the tags belonging to a structured tag.

Declaration

```

BOOL GAPIEnumTypeMembers (
    LPCSTR                lpszProjectFile,
    LPCSTR                lpszTypeName,
    DM_ENUM_TYPEMEMBERS_PROC lpfnCallback,
    LPVOID                lpvUser,
    LPCMN_ERROR            lpdmError);

```

Parameters**lpszProjectFile**

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in runtime.

If an empty string is entered, an internal DMEnumOpenedProjects is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected with the error, DM_E_NOT_CONNECTED.

lpszTypeName

Name of a type of structured tag whose tags are to be listed.

lpfnCallback

Pointer to your callback function that is called for every tag.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

List tags in a structured tag.

FALSE

Error.

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_TYMEMBERS_PROC (Page 1777)	List tags in a structured tag (callback)
GAPIEnumTypeMembersEx (Page 1778)	List tags in a structured tag
GAPIEnumTypeMembersEx4 (Page 1783)	List tags in a structured tag

See also

DM_ENUM_TYPEMEMBERS_PROC (Page 1777)

GAPIEnumTypeMembersEx (Page 1778)

GAPIEnumTypeMembersEx4 (Page 1783)

3.2.7.2 DM_ENUM_TYPEMEMBERS_PROC (RT Professional)**Description**

In order to evaluate the names of tags listed by the system, you must provide a DM_ENUM_TYPEMEMBERS_PROC type callback function.

Declaration

```
BOOL ( * DM_ENUM_TYPEMEMBERS_PROC ) (  
    LPCSTR    lpszMemberName,  
    LPVOID    lpvUser );
```

Parameters**lpszStructTypeName**

Pointer to the name of the tags associated with a structured tag.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value**TRUE**

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - ODK functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

dmclient.h
 dmclient.lib
 dmclient.dll

Related functions

GAPIEnumTypeMembers (Page 1773)	List tags in a structured tag

See also

GAPIEnumTypeMembers (Page 1773)

3.2.7.3 GAPIEnumTypeMembersEx (RT Professional)**Use**

The function returns a description, including all the predefined values of tags belonging to a structured tag but without scaling data.

If you want to read the scaling data, you need the GAPIEnumTypeMembersEx4 function.

Declaration

```

BOOL GAPIEnumTypeMembersEx (
    LPCSTR          lpszProjectFile,
    LPCSTR          lpszTypeName,
    DM_ENUM_TYPEMEMBERS_PROC_EX lpfnCallback,
    LPVOID          lpvUser,
    LPCMN_ERROR     lpdmError);

```

Parameters**lpszProjectFile**

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in runtime.

If an empty string is entered, an internal DMEnumOpenedProjects is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected with the error, DM_E_NOT_CONNECTED.

IpszTypeName

Name of a type of structured tag whose tags are to be listed.

Only local tags can be enumerated on clients.

lpfnCallback

Pointer to your callback function that is called for every tag.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tags of a structured tag listed.

FALSE

Error.

Comment

There is an advanced function, GAPIEnumTypeMembersEx4..

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_TYPEMEMBERS_PROC_EX (Page 1781)	List tags in a structured tag (callback)
GAPIEnumTypeMembers (Page 1773)	List tags in a structured tag
GAPIEnumTypeMembersEx4 (Page 1783)	List tags in a structured tag

See also

GAPIEnumTypeMembers (Page 1773)

DM_ENUM_TYPEMEMBERS_PROC_EX (Page 1781)

GAPIEnumTypeMembersEx4 (Page 1783)

3.2.7.4 GAPIEnumTypeMembersExStr (RT Professional)**Declaration**

```
BOOL GAPIEnumTypeMembersExStr (
    LPCTSTR                lpszProjectFile,
    LPCTSTR                lpszTypeName,
    DM_ENUM_TYPEMEMBERS_PROC_EXSTR lpfnCallback,
    LPVOID                 lpvUser,
    LPCMN_ERROR             lpcmError);
```

Description

This function returns a description, including all defaults of the tags belonging to a structured tag; however no scaling data are returned.

Parameters**lpszProjectFile**

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with `DMEnumOpenedProjects`, or with `DMGetRuntimeProject` in Runtime.

In WinCC version V5.0 SP2 or higher, entering a blank string will cause an internal `DMEnumOpenedProjects` to be executed for a currently open project.

In Runtime, only the currently open project may be entered. Other entries are rejected with error `DM_E_NOT_CONNECTED` in WinCC V5.0 SP2 or higher.

lpszTypeName

Name of the type of a structured tag whose tags you want to have enumerated.

Only local tag types can be enumerated on multi-clients (V5) or clients (V6).

lpfnCallback

Pointer to the callback function which is called for each tag.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the callback function.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tags of a structured tag have been enumerated

FALSE

Error

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_TYPEMEMBERS_PROC_EXSTR	List tags in a structured tag (callback)
GAPIEnumTypeMembersEx4	List tags in a structured tag

3.2.7.5 DM_ENUM_TYPEMEMBERS_PROC_EX (RT Professional)**Description**

In order to evaluate the descriptions of tags listed by the system, you must provide a DM_ENUM_TYPEMEMBERS_PROC_EX type callback function.

Declaration

```

BOOL ( * DM_ENUM_TYPEMEMBERS_PROC_EX ) (
    LPDM_VARKEY          lpdmVarKey,
    LPMCP_NEWVARIABLE_DATA_EX lpdmVarDataEx,
    LPVOID               lpvUser );

```

Parameters

- lpdmVarKey**
Pointer to the first of the AUTOHOTSPOT type structures with the keys (ID and name) of a tag
- lpdmVarDataEx**
Pointer to a MCP_NEWVARIABLE_DATA_EX (Page 1600) type structure with the description of a tag.
- lpvUser**
Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

- TRUE**
The enumeration is continued.
- FALSE**
The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

dmclient.h

Related functions

GAPIEnumTypeMembersEx (Page 1776)	List tags in a structured tag

See also

- GAPIEnumTypeMembersEx (Page 1776)
- MCP_NEWVARIABLE_DATA_EX (Page 1600)

3.2.7.6 GAPIEnumTypeMembersEx4 (RT Professional)

Use

The function returns a complete description, including all the predefined values of tags belonging to a structured tag.

Declaration

```

BOOL GAPIEnumTypeMembersEx4 (
    LPCSTR                lpszProjectFile,
    LPCSTR                lpszTypeName,
    DM_ENUM_TPEMEMBERS_PROC_EX4 lpfnCallback,
    LPVOID                lpvUser,
    LPCMN_ERROR            lpdmError);

```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in runtime.

If an empty string is entered, an internal DMEnumOpenedProjects is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected (DM_E_NOT_CONNECTED).

lpszTypeName

Name of a type of structured tag whose tags are to be listed.

Only local tags can be enumerated on clients.

lpfnCallback

Pointer to your callback function that is called for every tag.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE
Tags of a structured tag listed.

FALSE
Error.

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DM_ENUM_TYPEMEMBERS_PROC_EX4 (Page 1784)	List tags in a structured tag (callback)
GAPIEnumTypeMembers (Page 1773)	List tags in a structured tag

See also

GAPIEnumTypeMembers (Page 1773)
GAPIEnumTypeMembersEx (Page 1776)
DM_ENUM_TYPEMEMBERS_PROC_EX4 (Page 1784)

3.2.7.7 DM_ENUM_TYPEMEMBERS_PROC_EX4 (RT Professional)

Description

In order to evaluate the descriptions of tags listed by the system, you must provide a DM_ENUM_TYPEMEMBERS_PROC_EX4 type callback function. It differs from DM_ENUM_TYPEMEMBERS_PROC_EX due to the additional specification of scaling information.

Declaration

```
BOOL ( * DM_ENUM_TYPEMEMBERS_PROC_EX4) (
    LPDM_VARKEY          lpdmVarKey,
    LPMCP_NEWVARIABLE_DATA_EX4 lpdmVarDataEx,
    LPVOID                lpvUser );
```

Parameters

lpdmVarKey

Pointer to the first of the DM_VARKEY (Page 1590) type structures with the keys (ID and name) of a tag

lpdmVarDataEx

Pointer to a MCP_NEWVARIABLE_DATA_EX4 (Page 1602) type structure with the description of a tag.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

dmclient.h

Related functions

GAPIEnumTypeMembersEx4 (Page 1781)	List tags in a structured tag

See also

GAPIEnumTypeMembersEx4 (Page 1781)

DM_VARKEY (Page 1590)

MCP_NEWVARIABLE_DATA_EX4 (Page 1602)

3.2.7.8 GAPIEnumTypes (RT Professional)

Use

This function lists the names and ID numbers of the configured structured tag types.

Declaration

```
BOOL GAPIEnumTypes (
    LPCSTR          lpszProjectFile,
    DM_ENUM_TYPES_PROC lpfnCallback,
    LPVOID          lpvUser,
    LPCMN_ERROR      lpdmError);
```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEnumOpenedProjects or with DMGetRuntimeProject in runtime.

If an empty string is entered, an internal DMEnumOpenedProjects is performed on the currently open project.

Only the currently open project can be specified in runtime. Any other specification is rejected with the error, DM_E_NOT_CONNECTED.

lpfnCallback

Pointer to your callback function, which receives the data of the tag types.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Types of structured tags listed.

FALSE

Error.

Required files

dmclient.h
 dmclient.lib
 dmclient.dll

Related functions

DM_ENUM_TYPES_PROC (Page 1787)	List types of structured tags (callback)

Examples

Enumerate all structured types (Page 1830) "DM01.cpp"

See also

DM_ENUM_TYPES_PROC (Page 1787)
 Enumerate all structured types (Page 1830)

3.2.7.9 DM_ENUM_TYPES_PROC (RT Professional)**Description**

In order to evaluate the tag types listed by the system, you must provide a DM_ENUM_TYPES_PROC type callback function.

Declaration

```

BOOL ( * DM_ENUM_TYPES_PROC) (
    LPCSTR    lpszTypeName,
    DWORD     dwTypeID,
    DWORD     dwCreatorID,
    LPVOID    lpvUser );

```

Parameters**lpszTypeName**

Pointer on the name of the tag type.

dwTypeID

dwTypeID corresponds to the ID as assigned by GAPICreateType for the tag type.

dwCreatorID

In the creator identification, you can determine who created an object.
The values 0 – 10100 and 11000 – 11100 are reserved for internal or specific systems.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

dmclient.h

Related functions

GAPIEnumTypes (Page 1784)	List types of structured tags

Examples

Enumerate all structured types (Page 1830) "DM01.cpp"

See also

GAPIEnumTypes (Page 1784)
Enumerate all structured types (Page 1830)

3.2.8 Functions for processing connections (RT Professional)

3.2.8.1 DMEEnumConnectionData (RT Professional)

Use

The function determines the information about a configured logical connection.

Declaration

```

BOOL DMEEnumConnectionData (
    LPCSTR                lpszProjectFile,
    LPDM_CONNKEY          lpdmConnKey,
    DWORD                 dwItems,
    DM_ENUM_CONNECTION_PROC lpfnCallback,
    LPVOID                 lpvUser,
    LPCMN_ERROR            lpdmError);

```

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file is determined with DMEEnumOpenedProjects or with DMGetRuntimeProject in runtime.

lpdmConnKey

Pointer to the first DM_CONNKEY (Page 1564) type structure. These structures are used to specify the logical connections, the data of which are to be enumerated.

dwItems

The number of logical connections, the data of which is to be listed.

A value of 0 triggers the listing of all connections.

lpfnCallback

Pointer to your callback function, which receives the data of the logical connection.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Data listed.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DM_ENUM_CONNECTION_PROC (Page 1792)	List connection data (callback)

Examples

Enumerate all connections (Page 1832) "DM01.cpp"

OnTestEnumConnectionDataAll (Page 1837) "TESTCDoc.cpp"

See also

DM_CONNKEY (Page 1564)

DM_ENUM_CONNECTION_PROC (Page 1792)

Enumerate all connections (Page 1832)

OnTestEnumConnectionDataAll (Page 1837)

3.2.8.2 DMEnumConnectionDataExStr (RT Professional)

Declaration

```

BOOL DMEnumConnectionDataExStr (
    LPCTSTR                lpszProjectFile,
    DM_ENUM_CONNECTION_PROC_EXSTR lpfnCallback,
    LPVOID                 lpvUser,
    LPDWORD                lpdwConnectionCount,
    LPCMN_ERROR            lpdmError);

```

Description

This function is used to determine all configured logical connections.

Parameters

lpszProjectFile

Pointer to the name of the project file, including path and extension.

The name of the project file can be retrieved with DMEnumOpenedProjects, or with DMGetRuntimeProject in Runtime.

lpfnCallback

Pointer to your callback function which receives the data of the logical connection.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the callback function.

lpdwConnectionCount

Pointer to a "DWORD", in which the number of existing connections is returned.

Thus, "lpfnCallback = NULL" first determines how much memory is required for the connections.

lpdmError

Pointer to the data of the extended error message within the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Data have been enumerated

FALSE
Error

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient_exstr.h
dmclient.lib
dmclient.dll

Related functions

DM_ENUM_CONNECTION_PROC_EXSTR	List connection data (callback)

3.2.8.3 DM_ENUM_CONNECTION_PROC (RT Professional)

Description

In order to evaluate information about a logical connection listed by the system, you must provide a DM_ENUM_CONNECTION_PROC type callback function.

Declaration

```
BOOL ( * DM_ENUM_CONNECTION_PROC) (
    LPDM_CONNECTION_DATA    lpdmConData,
    LPVOID                   lpvUser);
```

Parameters

- lpdmConData**
Pointer to a DM_CONNECTION_DATA (Page 1563) type structure in which the data of a logical connection are stored.
- lpvUser**
Pointer to application-specific data. This pointer is made available again in the callback function.

Return value**TRUE**

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

dmclient.h

Related functions

DMEnumConnectionData (Page 1787)	List connection data

Examples

Enumerate all connections (Page 1832) "DM01.cpp"

See also

DM_CONNECTION_DATA (Page 1563)

DMEnumConnectionData (Page 1787)

Enumerate all connections (Page 1832)

3.2.9 Functions for the work platform (RT Professional)**3.2.9.1 DMGetOSVersion (RT Professional)****Use**

Determines the operating system used on the PC.

Declaration

```
DWORD DMGetOSVersion (
    VOID);
```

Parameters

None.

Return value

The return value indicates the operating system:

DM_OS_UNKNOWN	0	Unknown operating system
DM_OS_NT	1	Windows NT
DM_OS_32S	2	Windows 3.x with Win 32s
DM_OS_CHICAGO	3	Windows 95
DM_OS_2000	4	Windows 2000
DM_OS_XP	5	Windows XP
DM_OS_2003	6	Windows 2003
DM_OS_VISTA_32	0x0600 1	Windows Vista 32-bit
DM_OS_VISTA_SP1_32	0x0601 1	Windows Vista 32-bit Service Pack 1
DM_OS_VISTA_SP2_32	0x0602 1	Windows Vista 32-bit Service Pack 2
DM_OS_VISTA_64	0x9600 1	Windows Vista 64-bit
DM_OS_VISTA_SP1_64	0x9601 1	Windows Vista 64-bit Service Pack 1
DM_OS_VISTA_SP2_64	0x9602 1	Windows Vista 64-bit Service Pack 2
DM_OS_S2008_32	0x0600 3	Windows Server 2008 32-bit
DM_OS_S2008_SP1_32	0x0601 3	Windows Server 2008 32-bit Service Pack 1
DM_OS_S2008_SP2_32	0x0602 3	Windows Server 2008 32-bit Service Pack 2
DM_OS_S2008_64	0x9600 3	Windows Server 2008 64-bit
DM_OS_S2008_SP1_64	0x9601 3	Windows Server 2008 64-bit Service Pack 1
DM_OS_S2008_SP2_64	0x9602 3	Windows Server 2008 64-bit Service Pack 2
DM_OS_W7_32	0x0610 1	Windows 7 32-bit

DM_OS_W7_SP1_32	0x0611 1	Windows 7 32-bit Service Pack 1
DM_OS_W7_64	0x9610 1	Windows 7 64-bit
DM_OS_W7_SP1_64	0x9611 1	Windows 7 64-bit Service Pack 1
DM_OS_S2008R2_32	0x0610 3	Windows Server 2008 R2 (32-bit)
DM_OS_S2008R2_SP1_32	0x0611 3	Windows Server 2008 R2 32-bit Service Pack 1
DM_OS_S2008R2_64	0x9610 3	Windows Server 2008 R2 (64-bit)
DM_OS_S2008R2_SP1_64	0x9611 3	Windows Server 2008 R2 64-bit Service Pack 1
DM_OS_W8_32	0x0620 1	Windows 8 (32-bit)
DM_OS_W8_64	0x9620 1	Windows 8 (64-bit)
DM_OS_S2012_32	0x0620 3	Windows Server 2012 32-bit
DM_OS_S2012_64	0x9620 3	Windows Server 2012 64-bit
DM_OS_W81_32	0x0630 1	Windows 8.1 (32-bit)
DM_OS_W81_64	0x9630 1	Windows 8.1 (64-bit)
DM_OS_S2012R2_32	0x0630 3	Windows Server 2012 R2 (32-bit)
DM_OS_S2012R2_64	0x9630 3	Windows Server 2012 R2 (64-bit)
DM_OS_W10_64	0x9640 1	Windows 10 (64-bit)
DM_OS_S2016_64	0x9640 3	Windows Server 2016 64-bit

Required files

dmclient.h
dmclient.lib
dmclient.dll

3.2.9.2 DMGetSystemLocale (RT Professional)**Use**

Gets the code of the currently used configuration language.

Declaration

```
BOOL DMGetSystemLocale (
    LPDWORD      lpdwLocaleID,
    LPCMN_ERROR  lpdmError);
```

Parameters

lpdwLocaleID

Pointer to the code of the currently configured language.

Possible return values are the language codes for:

German	0x0407
English	0x0409
French	0x040C

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Currently used configuration language determined.

FALSE

Error

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

OnTestSystemLocale (Page 1843) "TESTCDoc.cpp"

See also

OnTestSystemLocale (Page 1843)

3.2.9.3 DMSetLanguage (RT Professional)**Use**

Switches the configuration language.

Declaration

```

BOOL DMSetLanguage (
    DWORD          dwLocaleID,
    LPCMN_ERROR     lpdmError );

```

Parameters**dwLocaleID**

Pointer to the code of the language to be set. Possible values are the language codes for:

German	0x0407
English	0x0409
French	0x040C

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Language switched.

FALSE

Error.

Error messages

DM_E_NOT_SUPPORTED	Requested service unavailable

Required files

dmclient.h

dmclient.lib

dmclient.dll

3.2.9.4 DMShowLanguageDialog (RT Professional)

Use

The function opens a dialog box for selecting the language.

Declaration

```

BOOL DMShowLanguageDialog (
    HWND          hwndParent,
    DWORD          dwFlags,
    DWORD          dwSetLocaleIDs[],
    UINT           uSetIDArraySize,
    LPDWORD        lpdwGetLocaleID,
    LPCMN_ERROR    lpdmError);

```

Parameters

hwndParent

Handle to the window that is used as the parent window for the dialog.

dwFlags

dwFlags indicates the languages that appear in the dialog and can be selected:

DM_LANGDLG_REMOVE	Set	All languages that can be registered with dwSetLocaleIDs are displayed, except for the languages in the EnumSystemLocales (LCID_SUPPORTED) array.
	Not set	All languages in the dwSetLocaleIDs array are displayed.
DM_LANGDLG_ONLY_PRIMARY	Set	Only the main languages (PRIMARY_LANGUAGE) are displayed.
	Not set	Sublanguages are also displayed.
DM_LANGDLG_NO_NOTIFY		Applications are not notified when the dialog is closed with "OK"

dwSetLocaleIDs

Array with the IDs of the languages to be displayed in the dialog.

uSetIDArraySize

Size of the dwSetLocaleIDs array.

lpdwGetLocaleID

Pointer to a DWORD, in which the selected language ID should be stored.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Close the dialog with "OK".

FALSE

Error or close dialog with "Cancel".

Error messages

DM_E_CANCEL	User has selected "Cancel" in a dialog

Required files

dmclient.h
dmclient.lib
dmclient.dll

3.2.10 Functions for updating tags (RT Professional)**3.2.10.1 DMBeginStartVarUpdate (RT Professional)****Use**

Trigger an update request. A transaction ID for identifying transaction is assigned with this function.

Declaration

```

BOOL DMBeginStartVarUpdate (
    LPDWORD          pdwTAID
    LPCMN_ERROR      lpdmError )

```

Parameters

pdwTAID

Pointer to a tag that contains the transaction ID assigned by the data manager after a successful call of the function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Transaction ID assigned.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h

dmclient.lib

dmclient.dll

Examples

OnTestVariablenBeginstartvarupdate (Page 1844) "TESTCDoc.cpp"

See also

OnTestVariablenBeginstartvarupdate (Page 1844)

DMEndStartVarUpdate (Page 1801)

DMResumeVarUpdate (Page 1802)

DMStartVarUpdate (Page 1804)

DMStartVarUpdateEx (Page 1807)

DMStopVarUpdate (Page 1819)

DMSuspendVarUpdate (Page 1821)

3.2.10.2 DMEndStartVarUpdate (RT Professional)

Use

After the function is successfully called, data manager updates the tags and forwards the new values to your application as required.

Declaration

```
BOOL DMEndStartVarUpdate (
    DWORD          dwTAID,
    LPCMN_ERROR     lpdmError );
```

Parameters

dwTAID

dwTAID contains the transaction ID assigned with the call of the DMBeginStartVarUpdate function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Update started.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_TAID	Invalid transaction ID

Required files

dmclient.h

Related functions

DMBeginStartVarUpdate (Page 1797)	Trigger an update request
DMStartVarUpdate (Page 1804)	Define update tags

Examples

OnTestVariablenEndstartvarupdate (Page 1845) "TESTCDoc.cpp"

See also

OnTestVariablenEndstartvarupdate (Page 1845)

DMBeginStartVarUpdate (Page 1797)

DMStartVarUpdate (Page 1804)

3.2.10.3 DMResumeVarUpdate (RT Professional)

Use

Starts the updating of all tags defined by the transaction, when the transaction reaches the value 0 after calling the reference counter.

The function is the counterpart to DMSuspendVarUpdate and should always be called together with it in pairs.

Declaration

```
BOOL DMResumeVarUpdate (  
    DWORD          dwTAID,  
    LPCMN_ERROR    lpdmError);
```

Parameters

dwTAID

dwTAID contains the transaction ID assigned with the call of the DMBeginStartVarUpdate function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Continue update.

FALSE

Error.

Comment

Transactions have a reference counter that is incremented each time the DMSuspendVarUpdate function is called. If the update of the tags should be continued, DMResumeVarUpdate must be called repeatedly until the reference counter returns to the 0 position.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_TAID	Invalid transaction ID

Required files

dmclient.h
dmclient.lib
dmclient.dll

Related functions

DMBeginStartVarUpdate (Page 1797)	Trigger an update request
DMSuspendVarUpdate (Page 1821)	Stop updating of tags

Examples

OnTestVariablenResumevarupdate (Page 1856) "TESTCDoc.cpp"

See also

DMBeginStartVarUpdate (Page 1797)
DMSuspendVarUpdate (Page 1821)
OnTestVariablenResumevarupdate (Page 1856)

3.2.10.4 DMStartVarUpdate (RT Professional)

Use

This function is used to specify the tags to be updated by the data manager. The data manager passes the new tag values to your application in the callback function when an update is performed.

DMStartVarUpdateEx is a more advanced function, which additionally returns the quality code in the DM_NOTIFY_VARIABLE_EX_PROC callback in the DM_VAR_UPDATE_STRUCTEX structure.

Declaration

```

BOOL DMStartVarUpdate (
    DWORD                dwTAID,
    LPDM_VARKEY          lpdmVarKey,
    DWORD                dwItems,
    DWORD                dwCycle,
    DM_NOTIFY_VARIABLE_PROC lpfnVariable,
    LPVOID               lpvUser,
    LPCMN_ERROR          lpdmError);

```

Parameters

dwTAID

dwTAID contains the transaction ID assigned with the call of the DMBeginStartVarUpdate function.

lpdmVarKey

Pointer to the first of the DM_VARKEY (Page 1590) type structures, through which the tags to be updated are specified.

dwItems

Number of tags to be updated in the passed DM_VARKEY structures.

dwCycle

The update cycle defined in dwCycle applies to all tags specified by lpdmVarKey. The upgrade cycle is defined by the index of the entries in the list of update cycles.

"On change"	Acyclic	Index: 0
"250 ms"	250	Index: 1
"500 ms"	500	Index: 2
"1 s"	1000	Index: 3
"2 s"	2000	Index: 4
"5 s"	5000	Index: 5
"10 s"	10000	Index: 6

"1 min"	60000	Index: 7
"5 min"	300000	Index: 8
"10 min"	600000	Index: 9
"1 h"	3600000	Index: 10
"User cycle 1"	(e.g. 2000)	Index: 11
"User cycle 2"	(e.g. 3000)	Index: 12
"User cycle 3"	(e.g. 4000)	Index: 13
"User cycle 4"	(e.g. 5000)	Index: 14
"User cycle 5"	(e.g. 10000)	Index: 15

lpfnVariable

Pointer to the DM_NOTIFY_VARIABLE_PROC callback function for data transfer by the data manager.

With lpfnVariable == NULL, the update in the process image of the data manager is performed at least in the required cycle, to the extent physically possible based on the current system load, computer performance, coupling medium etc. The determination of the current value needs to be performed independently by the application by calling the DMGetValue function.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the notification may already be delivered before the function call returns.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Update tags specified.

FALSE

Error.

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMBeginStartVarUpdate (Page 1797)	Trigger an update request
DM_NOTIFY_VARIABLE_PROC (Page 1806)	Specify update tags (callback)

See also

DMEndStartVarUpdate (Page 1799)

DM_VARKEY (Page 1590)

DMBeginStartVarUpdate (Page 1797)

DM_NOTIFY_VARIABLE_PROC (Page 1806)

3.2.10.5 DM_NOTIFY_VARIABLE_PROC (RT Professional)**Description**

In order to evaluate the data determined by the system, you must provide a DM_NOTIFY_VARIABLE_PROC type callback function.

Declaration

```

BOOL ( * DM_NOTIFY_VARIABLE_PROC) (
    DWORD                dwTAID,
    LPDM_VAR_UPDATE_STRUCT lpdmvus,
    DWORD                dwItems,
    LPVOID                lpvUser);

```

Parameters**dwTAID**

Transaction ID that was assigned by the data manager for the calling function.

lpdmvus

Pointer to the first DM_VAR_UPDATE_STRUCT (Page 1577) type structure that contains the values of the tags requested.

dwItems

Number of structures passed in lpdmvus (corresponding to the number of returned tag values).

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

The return values depend on your implementation

Comment**Note**

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

Required files

dmclient.h

Related functions

DMGetValueWait (Page 1700)	Get tag values with update
DMStartVarUpdate (Page 1802)	Define update tags

See also

DMStartVarUpdate (Page 1802)

DM_VAR_UPDATE_STRUCT (Page 1577)

DMGetValueWait (Page 1700)

3.2.10.6 DMStartVarUpdateEx (RT Professional)**Use**

This function is used to specify the tags to be updated by the data manager. Within the callback function, the data manager passes on the new tag values to the application when an update is executed.

In contrast to DMStartVarUpdate, the quality code is also provided in the DM_VAR_UPDATE_STRUCTEX structure, which is returned in the DM_NOTIFY_VARIABLESEX_PROC callback.

Declaration

```

BOOL DMStartVarUpdateEx (
    DWORD                dwTAID,
    LPDM_VARKEY          lpdmVarKey,
    DWORD                dwItems,
    DWORD                dwCycle,
    DM_NOTIFY_VARIABLEEX_PROC lpfnVariable,
    LPVOID               lpvUser,
    LPCMN_ERROR          lpdmError);

```

Parameters

dwTAID

dwTAID contains the transaction ID assigned with the call of the DMBeginStartVarUpdate function.

lpdmVarKey

Pointer to the first of the DM_VARKEY (Page 1590) type structures, through which the tags to be updated are specified.

dwItems

Number of tags to be updated in the passed DM_VARKEY structures.

dwCycle

The update cycle defined here applies to all tags specified by lpdmVarKey . The upgrade cycle is defined by the index of the entries in the list of update cycles.

"On change"	Acyclic	Index: 0
"250 ms"	250	Index: 1
"500 ms"	500	Index: 2
"1 s"	1000	Index: 3
"2 s"	2000	Index: 4
"5 s"	5000	Index: 5
"10 s"	10000	Index: 6
"1 min"	60000	Index: 7
"5 min"	300000	Index: 8
"10 min"	600000	Index: 9
"1 h"	3600000	Index: 10
"User cycle 1"	(e.g. 2000)	Index: 11
"User cycle 2"	(e.g. 3000)	Index: 12
"User cycle 3"	(e.g. 4000)	Index: 13
"User cycle 4"	(e.g. 5000)	Index: 14
"User cycle 5"	(e.g. 10000)	Index: 15

lpfnVariable

Pointer to the DM_NOTIFY_VARIABLEEX_PROC callback function for data transfer by the data manager.

With lpfnVariable == NULL, the update in the process image of the data manager is performed at least in the required cycle, to the extent physically possible based on the current system load, computer performance, coupling medium etc. The determination of the current value needs to be performed independently by the application by calling the DMGetValue function.

When a program reports a notify routine, it must empty its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Update tags specified.

FALSE

Error.

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMBeginStartVarUpdate (Page 1797)	Trigger an update request
DM_NOTIFY_VARIABLEEX_PROC (Page 1817)	Specify update tags (callback)

See also

DM_VARKEY (Page 1590)

DMBeginStartVarUpdate (Page 1797)

DM_NOTIFY_VARIABLEEX_PROC (Page 1817)

3.2.10.7 DMStartVarUpdateExStr (RT Professional)

Declaration

```

BOOL DMStartVarUpdateExStr (
    DWORD                dwTAID,
    DWORD                dwFlags,
    LPVARIANT            lpvdmVarKey,
    LPVARIANT            lpvCookie,
    DWORD                dwCycle,
    DM_NOTIFY_VARIABLE_PROC_EXSTR lpfnVariable,
    LPVOID               lpvUser,
    LPCMN_ERROR          lpdmError);

```

Description

This function is used to specify the tags to be updated by the data manager. Within the callback function, the data manager passes on the new tag values to the application when an update is executed.

Parameters

dwTAID

dwTAID contains the transaction ID allocated when the DMBeginStartVarUpdate function was called.

dwFlags

If the tag name needs to be returned in the VT_LPSTR format in the VARIANT of the DM_VAR_UPDATE_STRUCT_EXSTR structure of DM_NOTIFY_VARIABLE_PROC_EXSTR, the DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR flag can be specified here.

lpvdmVarKey

Pointer to VARIANT with VT_ARRAY | VT_VARIANT for passing on the tag list that identifies the tags to be changed.

A TagID is entered in the relevant list element with the type VT_I4 and the tag name with VT_BSTR. In addition, VT_LPSTR (PROPVARIANT) can be entered for passing on as allocated ASCII string in order, for example, to pass on constant names from the script. VT_LPSTR is then converted internally into the required VT_BSTR type.

For multi-client projects in WinCC Version 5.0 or higher, it may be required here to add a server prefix (refer to "Project types and versions").

lpvCookie

Pointer to a VARIANT for an additional list of user-dependent data for each tag.

This is intended as substitute for `lpvUserData` of the previous `DM_VARKEY` and is also returned in the `DM_VAR_UPDATE_STRUCT_EXSTR` structure for each tag.

dwCycle

The update cycle defined here is applied to all tags specified by means of `lpvdmVarKey`. The upgrade cycle is defined by the index of the entries in the list of update cycles.

"On change"	Acyclic	Index: 0
"250 ms"	250	Index: 1
"500 ms"	500	Index: 2
"1 s"	1000	Index: 3
"2 s"	2000	Index: 4
"5 s"	5000	Index: 5
"10 s"	10000	Index: 6
"1 min"	60000	Index: 7
"5 min"	300000	Index: 8
"10 min"	600000	Index: 9
"1 h"	3600000	Index: 10
"User cycle 1"	(e.g. 2000)	Index: 11
"User cycle 2"	(e.g. 3000)	Index: 12
"User cycle 3"	(e.g. 4000)	Index: 13
"User cycle 4"	(e.g. 5000)	Index: 14
"User cycle 5"	(e.g. 10000)	Index: 15

lpfnVariable

Pointer to the `DM_NOTIFY_VARIABLEEX_PROC` callback function for data transfer through the data manager.

If "`lpfnVariable`" = "NULL", the update in the process image of the data manager is performed at least in the required cycle, to the extent that this is physically possible based on the current system load, computer performance, coupling medium etc.; however, each value must be determined individually by calling the `DMGetValue` function through the application.

If a program requests a notify routine, it has to clear its message queue at regular intervals. Unread messages could block WinCC notifications or finally block WinCC altogether.

lpvUser

Pointer to application-specific data. The function does not evaluate this pointer, but makes it available again within the callback function.

lpdmError

Pointer to the data of the extended error message within the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Update tags have been specified

FALSE

Error

Required files

dmclient_exstr.h

dmclient.lib

dmclient.dll

Related functions

DMBeginStartVarUpdate	Trigger an update request

Examples**Script sample "Button DMStartVarUpdateExStr":**

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("OleAut32.dll")
// #include "OleAuto.h"
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound, unsigned
int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR* rgIndices,
void HUGE* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayUnlock(SAFEARRAY FAR* psa);
#pragma code()

extern BOOL DM_NOTIFY_VARIABLE_PROC_EXSTR
DM_NotifyVariableProcExStr_VarUpdate(
    DWORD dwTAID,
    LPDM_VAR_UPDATE_STRUCT_EXSTR lpdmvus,
    DWORD dwItems,
    LPVOID lpvUser);

VARIANT vVarKey;
VARIANT* pvElem;
SAFEARRAY* parrayKeys;
HRESULT hr;
CMN_ERROR err;
BOOL bRet;
long lInx;
DWORD dwTAID;
DWORD dwFlags;
DWORD dwCycle;
```

```

memset(&err, 0, sizeof(err));
bRet = FALSE;
parrayKeys = NULL;
lInx = 0L;
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
//dwCycle = 3; /* 1sec*/
dwCycle = 0; /* on change*/

printf("\r\n\r\n##### enter Test with DMStartVarUpdateExStr
#####");

dwTAID = GetTagDWord("dwUpdTAID");

if (dwTAID)
{
    printf("\r\nStartVarUpdate always running. Stop it before start
again!");
    printf("\r\n##### exit Test with DMStartVarUpdateExStr
(nothing done) #####\r\n");
    return;
}

VariantInit(&vVarKey);
parrayKeys = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vVarKey.vt = VT_ARRAY | VT_VARIANT;
vVarKey.u.parray = parrayKeys;

SafeArrayLock(parrayKeys);
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_1";
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_2";
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_3";
lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_4";
SafeArrayUnlock(parrayKeys);

bRet = DMBeginStartVarUpdate(&dwTAID, &err);
if (FALSE == bRet)
{
    printf("\r\n error DMBeginStartVarUpdate, err=%ld,%ld,%ld,%ld,
%ld,[%s]",

```

3.2 Data management functions (RT Professional)

```

        err.dwError1, err.dwError2, err.dwError3,
        err.dwError4, err.dwError5, err.szErrorText);
    }
    else
    {
        printf("\r\n  DMBeginStartVarUpdate  (dwTAID=%ld) OK.", dwTAID);
    }
    if (dwTAID)
    {
        SetTagDWord("dwUpdTAID", dwTAID);

        memset(&err, 0, sizeof(err));
        bRet = DMStartVarUpdateExStr(dwTAID, dwFlags, &vVarKey, dwCycle,
                                     DM_NotifyVariableProcExStr_VarUpdate,
NULL, &err);
        if (FALSE == bRet)
        {
            printf("\r\n  error DMStartVarUpdateExStr  (dwTAID=%ld): err=
%ld,%ld,%ld,%ld,%ld,[%s]",
                dwTAID, err.dwError1, err.dwError2, err.dwError3,
                err.dwError4, err.dwError5, err.szErrorText);
        }
        else
        {
            printf("\r\n  DMStartVarUpdateExStr  (dwTAID=%ld) OK.",
dwTAID);
        }

        memset(&err, 0, sizeof(err));
        bRet = DMEndStartVarUpdate(dwTAID, &err);
        if (FALSE == bRet)
        {
            printf("\r\n  error DMEndStartVarUpdate  (dwTAID=%ld): err=
%ld,%ld,%ld,%ld,%ld,[%s]",
                dwTAID, err.dwError1, err.dwError2, err.dwError3,
                err.dwError4, err.dwError5, err.szErrorText);
        }
        else
        {
            printf("\r\n  DMEndStartVarUpdate  (dwTAID=%ld) OK.", dwTAID);
        }
    }
    else
    {
        printf("\r\n  dwTAID == 0L ???");
    }

    printf("\r\n##### exit Test with DMStartVarUpdateExStr  (dwTAID=
%ld) #####\r\n", dwTAID);
}

```

Script sample "Project-Function DM_NotifyVariableProcExStr_VarUpdate":

```
#pragma code ("OleAut32.dll")
//#include "OleAuto.h"
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
#pragma code()

BOOL DM_NotifyVariableProcExStr_VarUpdate(
    DWORD                dwTAID,
    LPDMMVAR_UPDATE_STRUCT_EXSTRA lpdmvus,
    DWORD                dwItems,
    LPVOID                lpvUser)
{
    BOOL bRet = FALSE;
    int i = 0;
    HRESULT hr = 0L;

    printf("\r\n*** DM_NotifyVariableProcExStr_VarUpdate entry (dwTAID=
    %ld) ***", dwTAID);

    for (i = 0; i < dwItems; i++)
    {
        hr = VariantChangeType((VARIANTARG*)&(lpdmvus[i].vdmValue),
                                (VARIANTARG*)&(lpdmvus[i].vdmValue), 0,
                                VT_R8);
        if (VT_LPSTR == lpdmvus[i].vdmVarKey.vt)
        {
            printf("\r\n    [%d]:{VarKey=[%s]", i,
                (LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal);
        }
        else if (VT_I4 == lpdmvus[i].vdmVarKey.vt)
        {
            printf("\r\n    [%d]:{VarKey=[%ld]", i,
                lpdmvus[i].vdmVarKey.u.lVal);
        }
        else
        {
            printf("\r\n    [%d]:{unexpected VarKey.vt=[%d]",
                lpdmvus[i].vdmVarKey.vt);
        }
        printf(", dmValue=[%g]", lpdmvus[i].vdmValue.u.dblVal);
        printf(", dwState=[%ld]", lpdmvus[i].dwState);
        printf(", dwType=[%ld]", lpdmvus[i].dmTypeRef.dwType);
        printf(", dwSize=[%ld]", lpdmvus[i].dmTypeRef.dwSize);
        if (lpdmvus[i].dmTypeRef.lpszTypeName)
        {
            printf(", lpszTypeName=[%s]",
                lpdmvus[i].dmTypeRef.lpszTypeName);
            printf(", dwNameCharCount=[%ld]",
                lpdmvus[i].dmTypeRef.dwNameCharCount);
        }
    }
}
```

```

    }
    else
    {
        printf(", lpszTypeName=[NULL]");
    }
    printf(")");
}

printf("\r\n*** DM_NotifyVariableProcExStr_VarUpdate exit (dwTAID=
%d)***", dwTAID);
return bRet;
}

```

Script sample "Button DMStopVarUpdate":

```

#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
    DWORD dwTAID;
    CMN_ERROR err;
    BOOL bRet;

    dwTAID = GetTagDWord("dwUpdTAID");
    memset(&err, 0, sizeof(err));
    bRet = FALSE;

    if (dwTAID)
    {
        bRet = DMStopVarUpdate(dwTAID, &err);
        if (FALSE == bRet)
        {
            printf("\r\n!!! error DMStopVarUpdate (dwTAID=%ld): err=%ld,
            %ld,%ld,%ld,%ld,[%s] !!!\r\n",
                dwTAID, err.dwError1, err.dwError2, err.dwError3,
                err.dwError4, err.dwError5, err.szErrorText);
            memset(&err, 0, sizeof(err));
            bRet = DMStopAllUpdates(&err);
            if (FALSE == bRet)
            {
                printf("\r\n!!! error DMStopAllUpdates (dwTAID=%ld):
                err=%ld,%ld,%ld,%ld,%ld,[%s] !!!\r\n",
                    dwTAID, err.dwError1, err.dwError2, err.dwError3,
                    err.dwError4, err.dwError5, err.szErrorText);
            }
            else
            {
                printf("\r\n DMStopAllUpdates OK.");
            }
        }
        else
        {

```

```

        printf("\r\n  DMStopVarUpdate   (dwTAID=%ld) OK.", dwTAID);
    }
}

dwTAID = 0L;

SetTagDWord("dwUpdTAID", dwTAID);
}

```

3.2.10.8 DM_NOTIFY_VARIABLEEX_PROC (RT Professional)

Description

In order to evaluate the data determined by the system, you must provide a DM_NOTIFY_VARIABLEEX_PROC type callback function.

Declaration

```

BOOL ( * DM_NOTIFY_VARIABLEEX_PROC) (
    DWORD                dwTAID,
    LPDM_VAR_UPDATE_STRUTEX lpdmvus,
    DWORD                dwItems,
    LPVOID                lpvUser);

```

Parameters

dwTAID

Transaction ID that was assigned by the data manager for the calling function.

lpdmvus

Pointer to the first DM_VAR_UPDATE_STRUTEX (Page 1579) type structure that contains the values of the tags requested.

dwItems

Number of structures passed in lpdmvus (corresponding to the number of returned tag values).

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

The return values depend on your implementation

Comment

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

Required files

dmclient.h

Related functions

DMGetValueWaitEx (Page 1702)	Get tag values with update
DMStartVarUpdateEx (Page 1805)	Define update tags

See also

DMStartVarUpdateEx (Page 1805)

DM_VAR_UPDATE_STRUCTEX (Page 1579)

DMGetValueWaitEx (Page 1702)

3.2.10.9 DMStopAllUpdates (RT Professional)

Use

Stops all tag updates requested by the application. This function should be preferably called after closing the corresponding module.

Declaration

```
BOOL DMStopAllUpdates (  
    LPCMN_ERROR    lpdmError);
```


Parameters**lpdmError**

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Updating of tags stopped.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager

Required files

dmclient.h
dmclient.lib
dmclient.dll

Examples

OnTestVariablenStopallupdates (Page 1859) "TESTCDoc.cpp"

See also

OnTestVariablenStopallupdates (Page 1859)

3.2.10.10 DMStopVarUpdate (RT Professional)**Use**

Stops the updating of tags for a specific transaction.

Declaration

```
BOOL DMStopVarUpdate (
    DWORD          dwTAID,
    LPCMN_ERROR    lpdmError);
```

Parameters

dwTAID

dwTAID contains the transaction ID assigned with the call of the DMBeginStartVarUpdate function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Updating of tags stopped.

FALSE

Error.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_TAID	Invalid transaction ID

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMBeginStartVarUpdate (Page 1797)	Trigger an update request

Examples

OnTestVariablenStopvarupdate (Page 1859) "TESTCDoc.cpp"

See also

OnTestVariablenStopvarupdate (Page 1859)

DMBeginStartVarUpdate (Page 1797)

3.2.10.11 DMSuspendVarUpdate (RT Professional)

Use

Interrupts the updating of all tags defined by the transaction.

Tags continue to be updated within the process image of the data manager. However, the values are no longer transferred to the application.

Declaration

```
BOOL DMSuspendVarUpdate (  
    DWORD          dwTAID,  
    LPCMN_ERROR    lpdmError);
```

Parameters

dwTAID

dwTAID contains the transaction ID assigned with the call of the DMBeginStartVarUpdate function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Updating suspended.

FALSE

Error.

Comment

Transactions have a reference counter, therefore the DMSuspendVarUpdate function can be called multiple times for a transaction. If the transaction is already in the "SUSPENDED" state, only the reference counter is incremented.

To continue updating, DMResumeVarUpdate must be called repeatedly until the reference counter returns to the 0 position.

Error messages

DM_E_NOT_CONNECTED	No connection to data manager
DM_E_INVALID_TAID	Invalid transaction ID

Required files

dmclient.h

dmclient.lib

dmclient.dll

Related functions

DMBeginStartVarUpdate (Page 1797)	Trigger an update request
DMResumeVarUpdate (Page 1800)	Continue updating of tags

Examples

OnTestVariablenSuspendvarupdate (Page 1860) "TESTCDoc.cpp"

See also

DMResumeVarUpdate (Page 1800)

DMBeginStartVarUpdate (Page 1797)

OnTestVariablenSuspendvarupdate (Page 1860)

3.2.11 Examples (RT Professional)

3.2.11.1 Connection to DM (RT Professional)

Example

```
// =====
// =====
// Desc. : Modul with examples for Data-Manager
// *****
#include "stdafx.h" // if MFC classes
// #include "odkapi.h" // if console application
#include <time.h>
TCHAR g_szProjectFile[255] = {0};
TCHAR g_szDSNName[255] = {0};
#include "DM01.h"

//{{ODK_EXAMPLE}Connection to DM (MCP)}
//{{FUNCTION}DMGetConnectionState (MCP)}
//{{FUNCTION}DMConnect (MCP)}
//{{FUNCTION}DM_NOTIFY_PROC (MCP)}
//{{FUNCTION}DMDisconnect (MCP)}
//{{FUNCTION}(END)}

// =====
// Function: MyDMConnect(void) ODK DM CS
// =====
BOOL MyDMConnect(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szAppName[255];
    VOID* pvUser = AfxGetApp();
    _tcsncpy_s(szAppName, _countof(szAppName), _T("MyODKApp_23"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = DMGetConnectionState(&Error);
    if(FALSE == ret) // not connected
    {
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = DMConnect(szAppName, MyDMNotifyCallback, pvUser, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMConnect"));
        }
        ODKTrace(szText);
    }
}
```

3.2 Data management functions (RT Professional)

```

else // already connected
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMGetConnectionState: OK"));
}
else // already connected
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMGetConnectionState: OK"));
    ODKTrace(szText);
}
return(ret);
}

// IMPLEMENTATION
// =====
// Function: MyDMGetConnectionState(void) ODK DM CS
// =====
BOOL MyDMGetConnectionState(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szAppName[255];
    VOID* pvUser = AfxGetApp();
    _tcsncpy_s(szAppName, _countof(szAppName), _T("MyODKApp_23"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = DMGetConnectionState(&Error);
    if(FALSE == ret) // not connected
    {
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = DMConnect(szAppName, MyDMNotifyCallback, pvUser, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMConnect"));
        }
        ODKTrace(szText);
        //printf("%s\r\n", szText);
    }
    else // already connected
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMGetConnectionState: OK"));
        ODKTrace(szText);
    }
    return(ret);
}

// =====
// Function: MyDMDisconnect(void) ODK DM CS
// =====
BOOL MyDMDisconnect(void)
{

```

```

    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = DMDisconnect(&Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMConnect: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMDisconnect"));
    }
    ODKTrace(szText);
    return(ret);
}

// =====
// Function: MyDMNotifyCallback
// =====
BOOL MyDMNotifyCallback(DWORD dwNotifyClass, DWORD dwNotifyCode, LPBYTE lpbyData,
    DWORD dwItems, LPVOID lpvUser)
{
    lpvUser;
    lpbyData;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("***DMNotifyCallback**"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  NotifyClass = 0x
%08X"), dwNotifyClass);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  NotifyCode = 0x%08X"),
dwNotifyCode);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  Items = %d"), dwItems);
    ODKTrace(szText);
    //printf("%s\r\n", szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("*****"));
    ODKTrace(szText);
    return(TRUE);
}

//{{ODK_EXAMPLE}(END)}

```

See also

[DMConnect \(Page 1626\)](#)
[DMDisconnect \(Page 1632\)](#)
[DMGetConnectionState \(Page 1644\)](#)

3.2.11.2 Enum Data of Tags (RT Professional)

Example

```

//{{ODK_EXAMPLE}Enum Data of Tags (MCP)}
//{{FUNCTION}DMEEnumVarData4 (MCP)}
//{{FUNCTION}DM_ENUM_VARIABLE_PROC4 (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: MyDMEEnumVarData4(void) ODK DM CS
// =====
// Desc. :
//-----
BOOL MyDMEEnumVariable4Callback(LPDM_VARKEY lpdmVarKey, LPDM_VARIABLE_DATA4 lpdmVarData,
LPVOID lpvUser)
{
    lpvUser;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEEnumVariableCallback"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...szName=%s dwVarProperty=0x
%04X"), lpdmVarKey->szName, lpdmVarData->dwVarProperty);
    ODKTrace(szText);
    return TRUE;
}

void MyDMEEnumVarData4()
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    DM_VARKEY dmVarKey[2];
    TCHAR szText[255];
    CMN_ERROR Error;
    VOID* pUser = NULL;
    memset(&Error, 0, sizeof(CMN_ERROR));
    memset(&dmVarKey, 0, 2 * sizeof(DM_VARKEY));
    dmVarKey[0].dwKeyType = DM_VARKEY_NAME;
    _tcsncpy_s(dmVarKey[0].szName, _countof(dmVarKey[0].szName), _T("INT_TEST_VAR"),
    _TRUNCATE);
    dmVarKey[1].dwKeyType = DM_VARKEY_NAME;
    _tcsncpy_s(dmVarKey[1].szName, _countof(dmVarKey[1].szName), _T("EXT_TEST_VAR"),
    _TRUNCATE);
    MyDMEEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    ret = DMEEnumVarData4 (/*PROJ_PATH*/g_szProjectFile,
        &dmVarKey[0],
        2,
        MyDMEEnumVariable4Callback,
        pUser,
        &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMEEnumVarData4: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
}

```



```
    }  
}  
// {{ODK_EXAMPLE}} (END) }
```

See also

[DMEnumVarData4 \(Page 1675\)](#)

[DM_ENUM_VARIABLE_PROC4 \(Page 1678\)](#)

3.2.11.3 Enum open projects (RT Professional)

Example

```

//{{ODK_EXAMPLE}Enum open projects (MCP)}
//{{FUNCTION}DMEEnumOpenedProjects (MCP)}
//{{FUNCTION}DM_ENUM_OPENED_PROJECTS_PROC (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: MyDMEEnumOpenedProjects(void) ODK DM CS
// =====
// Desc. : Inquire project informations
// =====
BOOL MyDMEEnumOpenProjectsCallback( LPDM_PROJECT_INFO lpInfo, LPVOID lpvUser )
{
    lpvUser;
    // this callback is only called once by every call of DMEEnumOpenedProjects
    // because, there can only opened one project at the same time
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("**DMEEnumOpenProjects**"));
    ODKTrace(szText);
    // copy the project file name to g_szProjectFile for global use
    // many API-functions need this name to select the project database
    _tcsncpy_s(g_szProjectFile, _countof(g_szProjectFile), lpInfo->szProjectFile,
    _TRUNCATE);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  ProjectFile = %s"),lpInfo-
    >szProjectFile);
    ODKTrace(szText);

    _tcsncpy_s(g_szDSNName, _countof(g_szDSNName), lpInfo->szDSNName, _TRUNCATE);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  DSNName = %s"),lpInfo-
    >szDSNName);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  DataLocale = 0x%08X"),lpInfo-
    >dwDataLocale);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("*****"));
    ODKTrace(szText);
    return( TRUE ); // only 1. element
}

BOOL MyDMEEnumOpenedProjects(void)
{
    CMN_ERROR Error;
    BOOL ret= FALSE;
    DWORD dwItems;
    TCHAR szText[255];
    VOID* pvUser = AfxGetApp();
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = MyDMConnect();
    if(TRUE == ret)
    {
        ret = DMEEnumOpenedProjects(&dwItems, MyDMEEnumOpenProjectsCallback, pvUser, &Error);
        if(FALSE == ret)

```

```
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
DMEnumOpenedProjects: E1= 0x%08lx ; E2= 0x%08lx ; %s"), %s",
                Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEnumOpenedProjects"));
}
ODKTrace(szText);
}
return(ret);
}
//{{ODK_EXAMPLE}(END)}
```

See also

DMEnumOpenedProjects (Page 1652)

DM_ENUM_OPENED_PROJECTS_PROC (Page 1653)

3.2.11.4 Enumerate all structured types (RT Professional)

Example

```

//{{ODK_EXAMPLE}Enumerate all structured types (MCP)}
//{{FUNCTION}GAPIEnumTypes (MCP)}
//{{FUNCTION}DM_ENUM_TYPES_PROC (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: MyGAPIEnumTypes(void) ODK DM CS
// =====
// Desc. :
//-----
BOOL MyDMEnumTypeCallback(LPCSTR lpszTypeName,DWORD dwTypeID,
                          DWORD dwCreatorID, LPVOID lpvUser )
{
    lpvUser;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("***DMEnumTypeCallback**"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TypeName = %s TypeID = 0x%08x
CreatorID = %d"),
                lpszTypeName,dwTypeID,dwCreatorID);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("*****"));
    ODKTrace(szText);
    return TRUE;
}

void MyGAPIEnumTypes()
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szProjectFile[255];
    VOID* pvUser = NULL;
    ret = FALSE;
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = MyDMGetConnectionState(); //check the connection state to DM
    if(FALSE != ret)
    {
        MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
        _tcsncpy_s(szProjectFile, _countof(szProjectFile), /*PROJ_PATH*/g_szProjectFile,
        _TRUNCATE);
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = GAPIEnumTypes(szProjectFile,
                            MyDMEnumTypeCallback,
                            pvUser,
                            &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMEnumTypes: E1=
0x%08lx ; E2= 0x%08lx ; %s"),

```

```
        Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEnumTypes"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE} (END) }
```

See also

[GAPIEnumTypes \(Page 1784\)](#)

[DM_ENUM_TYPES_PROC \(Page 1785\)](#)

3.2.11.5 Enumerate all connections (RT Professional)

Example

```

//{{ODK_EXAMPLE}Enumerate all connections (MCP)}
//{{FUNCTION}DMEEnumConnectionData (MCP)}
//{{FUNCTION}DM_ENUM_CONNECTION_PROC (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: MyDMEEnumConnectionData(void) ODK DM CS
// =====
// Desc. :
//-----
BOOL MyDMEEnumConnectionCallback(LPDM_CONNECTION_DATA lpdmConData, LPVOID lpvUser)
{
    lpvUser;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEEnumConnectionCallback"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Connection = %s"), lpdmConData-
>szConnection);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...UnitName = %s"), lpdmConData-
>szUnitName);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Common=%s"), lpdmConData-
>szCommon);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Specific=%s"), lpdmConData-
>szSpecific);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...VarNum=%d"), lpdmConData-
>dwVarNum);
    ODKTrace(szText);
    return TRUE;
}

void MyDMEEnumConnectionData()
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    TCHAR szText[255];
    DWORD dwItems = 0;
    TCHAR szProjectFile[255];
    DM_CONNKEY ConnKey;
    memset(&ConnKey, 0, sizeof(ConnKey));
    memset(&Error, 0, sizeof(Error));
    ret = MyDMGetConnectionState();
    if(FALSE != ret)
    {
        MyDMEEnumOpenedProjects(); // open the DM and set the g_szProjectFile
        _tcsncpy_s(szProjectFile, _countof(szProjectFile), /*PROJ_PATH*/g_szProjectFile,
        _TRUNCATE);
        ret = DMEEnumConnectionData(szProjectFile,

```

```

        &ConnKey,
        dwItems,
        MyDMEEnumConnectionCallback,
        NULL,
        &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
DMEEnumConnectionData: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEEnumConnectionData"));
    }
    ODKTrace(szText);
}
}
//{{ODK_EXAMPLE}(END)}

```

See also

DMEEnumConnectionData (Page 1787)

DM_ENUM_CONNECTION_PROC (Page 1790)

3.2.11.6 Inquire project informations (RT Professional)

Example

```

//{{ODK_EXAMPLE}Inquire project informations (MCP)}
//{{FUNCTION}DMGetProjectInformation (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: DMGetProjectInformation(void) ODK DM CS
// =====
// Desc. : Inquire project informations
// =====
void MyDMGetProjectInformation(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szProjectFile[_MAX_PATH +1];
    VOID* pvUser = AfxGetApp();
    DM_PROJECT_INFO Info;
    memset(&Error, 0, sizeof(CMN_ERROR));
    memset(&Info, 0, sizeof(DM_PROJECT_INFO));
    ret = MyDMGetConnectionState(); //check the connection state
    if(FALSE != ret)
    {
        MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
        _tcsncpy_s(szProjectFile, _countof(szProjectFile), /*PROJ_PATH*/g_szProjectFile,
        _TRUNCATE);
        ret = DMGetProjectInformation(szProjectFile, &Info, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
DMGetProjectInformation: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("***DMGetProjectInformation:
%d**"), ret);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  ProjectFile=
%s"), Info.szProjectFile);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  DSNName=
%s"), Info.szDSNName);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  DataLocale=0x
%08X"), Info.dwDataLocale);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE,
            _T("*****"), ret);
            ODKTrace(szText);
        }
    }
}

```



```

    }
}
//{{ODK_EXAMPLE} (END) }

```

See also

DMGetProjectInformation (Page 1656)

3.2.11.7 OnTestDeactivateRuntimeProject (RT Professional)

Example

```

//{{ODK_EXAMPLE}OnTestDeactivateRuntimeProject (MCP)}
//{{FUNCTION}DMDeactivateRTProject (MCP)}
//{{FUNCTION} (END) }
void CTestCliDoc::OnTestDeactivateRuntimeProject()
{
    CCmnError cmnError;
    if(!::DMDeactivateRTProject(cmnError))
    {
        cmnError.Show(__T("DMDeactivateRTProject failed\n"));
    }
}
//{{ODK_EXAMPLE} (END) }

```

3.2.11.8 OnTestEnumGroupsAll (RT Professional)

Example

```

//{{ODK_EXAMPLE}OnTestEnumGroupsAll (MCP)}
//{{FUNCTION}DMEEnumVarGrpData (MCP)}
//{{FUNCTION} (END) }
void CTestCliDoc::OnTestEnumGroupsAll()
{
    CCmnError cmnError;
    if(!::DMEEnumVarGrpData((LPSTR)(LPCTSTR) m_strProject, NULL, 0, EnumVarGrpProc, this, cmnError))
    {
        cmnError.Show(__T("DMEEnumVarGrpData failed\n"));
    }
}
//{{ODK_EXAMPLE} (END) }

```

See also

DMEEnumVarGrpData (Page 1679)

3.2.11.9 OnTestEnumVariables (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestEnumVariables (MCP)}
//{{FUNCTION}DMEEnumVariables (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestEnumVariables()
{
    DM_VARFILTER dmVarFilter;
    memset( &dmVarFilter, 0, sizeof(DM_VARFILTER) );
    //-----
    // Initialisierung des Filters //
    //-----
    // //
    dmVarFilter.dwFlags = DM_VARFILTER_TYPE;
    dmVarFilter.dwNumTypes = 3;
    DWORD dwFilterTypes[3];
    dwFilterTypes[0] = DM_VARTYPE_BIT;
    dwFilterTypes[1] = DM_VARTYPE_DWORD;
    dwFilterTypes[2] = DM_VARTYPE_DOUBLE;
    dmVarFilter.pdwTypes = dwFilterTypes;
    dmVarFilter.lpszName = __T("VAR_1_BIT");
    dmVarFilter.lpszGroup = __T("VARGROUP_1");
    dmVarFilter.lpszConn = __T("TF_CONN_1");
    // //
    //-----
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMEEnumVariables((LPSTR)(LPCTSTR) m_strProject, NULL/*&dmVarFilter*/,
EnumVariablesProc, this, &Error))
    {
        Error.Show(__T("DMEEnumVariables failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

DMEEnumVariables (Page 1684)

3.2.11.10 OnTestEnumConnectionDataAll (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestEnumConnectionDataAll (MCP)}  
//{{FUNCTION}DMEEnumConnectionData (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestEnumConnectionDataAll()  
{  
    CCmnError Error;  
    memset(&Error,0,sizeof(CCmnError));  
    if( !::DMEEnumConnectionData( m_strProject, NULL, 0, EnumConnectionDataProc, this,  
&Error))  
    {  
        Error.Show(__T("DMEEnumConnectionData failed\n"));  
    }  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

DMEEnumConnectionData (Page 1787)

3.2.11.11 OnTestMachines (RT Professional)

Example

```

//{{ODK_EXAMPLE}OnTestMachines (MCP)}
//{{FUNCTION}DMGetMachineTable (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestMachines()
{
    CMN_ERROR Error;
    memset(&Error,0,sizeof(CMN_ERROR));
    DM_MACHINE_TABLE dmMachineTable;
    memset(&dmMachineTable, 0, sizeof(dmMachineTable));
    if(!DMGetMachineTable(m_strProject, &dmMachineTable, &Error))
    {
        // Error.Show(__TEXT("DMGetMachineTable failed\n"));
        // AfxMessageBox( strError );
    }
    else
    {
        for( int i = 0; i < dmmachinetable.nNumMachines; i++ )
        {
            CString strData;
            strData.Format( _T( "Rechner : %s, Typ : %s, Standort : %s" ),
                dmMachineTable.tm[i].szMachineName,
                dmMachineTable.tm[i].fServer ? _T( "Server" ) : _T( "Client oder ES" ),
                dmMachineTable.tm[i].fLocal ? _T( "local" ) : _T( "remote" ));
            PutStr( strData);
        }
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

DMGetMachineTable (Page 1648)

3.2.11.12 OnTestProjectInfo (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestProjectInfo (MCP)}
//{{FUNCTION}DMGetProjectInformation (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestProjectInfo()
{
    CCmnError cmnError;
    DM_PROJECT_INFO ProjectInfo;
    memset(&ProjectInfo, 0, sizeof(DM_PROJECT_INFO));
    if(!DMGetProjectInformation(m_strProject, &ProjectInfo, cmnError))
    {
        cmnError.Show(__TEXT("DMGetProjectInformation failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("Projekt : %s, Data Source : %s, DataLocale : %08X"),
            ProjectInfo.szProjectFile,
            ProjectInfo.szDSNName,
            ProjectInfo.dwDataLocale );
        PutStr(strData);
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

DMGetProjectInformation (Page 1656)

3.2.11.13 OnTestProjectPaths (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestProjectPaths (MCP)}
//{{FUNCTION}DMGetProjectDirectory (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestProjectPaths()
{
    DM_DIRECTORY_INFO dmDirInfo;
    CCmnError Error;
    memset(&dmDirInfo, 0, sizeof(DM_DIRECTORY_INFO));
    memset(&Error, 0, sizeof(CCmnError));
    if(!DMGetProjectDirectory(m_strAppName, m_strProject,
        &dmDirInfo, &Error))
    {
        Error.Show(__TEXT("DMGetProjectDirectory failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("szProjectDir = %s"), dmDirInfo.szProjectDir);
        PutStr(strData);
        strData.Format(_T("szProjectAppDir = %s"), dmDirInfo.szProjectAppDir);
        PutStr(strData);
        strData.Format(_T("szProjectGlobalLibDir = %s"), dmDirInfo.szGlobalLibDir);
        PutStr(strData);
        strData.Format(_T("szProjectLibDir = %s"), dmDirInfo.szProjectLibDir);
        PutStr(strData);
        strData.Format(_T("szLokalProjectAppDir = %s"), dmDirInfo.szLokalProjectAppDir);
        PutStr(strData);
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

[DMGetProjectDirectory \(Page 1655\)](#)

3.2.11.14 OnTestOpenProject (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestOpenProject (MCP)}
//{{FUNCTION}DmOpenProject (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestOpenProject()
{
    #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    TCHAR szProject[_MAX_PATH + 1];
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    CTestCliView* pView = GetFirstView();
    memset(szProject,0, sizeof(szProject)); // Delete projectname to call dialog
    // Or set fixed projectname
    strcpy( szProject, _T(PROJ_PATH) );
    if(!DmOpenProject(pView->GetSafeHwnd(),
        szProject, NELEM(szProject), &Error))
    {
        Error.Show(__TEXT("DmOpenProject failed.\n"));
    }
    else
    {
        AfxMessageBox(szProject);
        m_strProject = szProject;
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

DmOpenProjectPlus (Page 1660)

3.2.11.15 OnTestOpenProjects (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestOpenProjects (MCP)}  
//{{FUNCTION}DMLenumOpenedProjects (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestOpenProjects()  
{  
    CCmnError Error;  
    memset(&Error,0,sizeof(CCmnError));  
    if(!DMLenumOpenedProjects(NULL, OpenProjectsProc, this, &Error))  
    {  
        Error.Show(__TEXT("DMLenumOpenedProjects failed\n"));  
    }  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

DMLenumOpenedProjects (Page 1652)

3.2.11.16 OnTestRuntimeProject (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestRuntimeProject (MCP)}  
//{{FUNCTION}DMLGetRuntimeProject (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestRuntimeProject()  
{  
    CCmnError Error;  
    memset(&Error,0,sizeof(CCmnError));  
    TCHAR szBuffer[_MAX_PATH + 1];  
    if(!DMLGetRuntimeProject(szBuffer, NELEM(szBuffer), &Error))  
    {  
        Error.Show(__TEXT("DMLGetRuntimeProject failed\n"));  
    }  
    else  
    {  
        PutStr(szBuffer);  
    }  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

DMLGetRuntimeProject (Page 1658)

3.2.11.17 OnTestSystemLocale (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestSystemLocale (MCP)}
//{{FUNCTION}DMGetSystemLocale (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestSystemLocale()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    DWORD dwLocaleID = 0;
    if(!DMGetSystemLocale(&dwLocaleID, &Error))
    {
        Error.Show(__TEXT("DMGetSystemLocale failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("Systemlocale : %08X"), dwLocaleID);
        PutStr(strData);
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

DMGetSystemLocale (Page 1793)

3.2.11.18 OnTestUpdateCycles (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestUpdateCycles (MCP)}
//{{FUNCTION}DMLenumUpdateCycles (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestUpdateCycles()
{
    DWORD dwNumCalls = 0;
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !DMLenumUpdateCycles(m_strProject, &dwNumCalls,
        EnumCyclesProc, this, &Error))
    {
        Error.Show(__TEXT("DMLenumUpdateCycles failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

[DMEnumUpdateCycles \(Page 1637\)](#)

3.2.11.19 OnTestVariablenBeginstartvarupdate (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestVariablenBeginstartvarupdate (MCP)}  
//{{FUNCTION}DMBeginStartVarUpdate (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestVariablenBeginstartvarupdate()  
{  
    CCmnError Error;  
    memset(&Error,0,sizeof(CCmnError));  
    if(!DMBeginStartVarUpdate(&m_dwTAID, &Error))  
    {  
        Error.Show(_T("DMBeginStartVarUpdate failed\n"));  
    }  
    else  
    {  
        CString strData;  
        strData.Format(_T("DMBeginStartVarUpdate: TAID:%lu."), m_dwTAID);  
    }  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

[DMBeginStartVarUpdate \(Page 1797\)](#)

3.2.11.20 OnTestVariablenEndstartvarupdate (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestVariablenEndstartvarupdate (MCP)}
//{{FUNCTION}DMEndStartVarUpdate (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenEndstartvarupdate()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !DMEndStartVarUpdate(m_dwTAID, &Error))
    {
        Error.Show(_T("DMEndStartVarUpdate failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMEndStartVarUpdate: TAID:%lu."), m_dwTAID);
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

DMEndStartVarUpdate (Page 1799)

3.2.11.21 OnTestVariablenGetvalue (RT Professional)

Example

```

//{{ODK_EXAMPLE}OnTestVariablenGetvalue (MCP)}
//{{FUNCTION}DMGetValue (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetvalue()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();

    LPDM_VAR_UPDATE_STRUCT lpdmvus = new DM_VAR_UPDATE_STRUCT[nNum];
    memset(lpdmvus, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    // the included VARIANT's in the DM_VAR_UPDATE_STRUCT's then initialized to VT_EMPTY
    with the memset,
    // don't do this later again, because VT_BSTR's can be present after DMGetValue(...)!

    CCmnError Error;
    memset(&Error, 0, sizeof(CCmnError));

    if( !DMGetValue(lpdmVarKey, nNum, lpdmvus, &Error))
    {
        Error.Show(_T("DMGetValue failed\n"));
    }
    else
    {
        CString strData;
        for( int i=0; i < nnum; i++)
        {
            LPDM_VAR_UPDATE_STRUCT lpdmvus2 = &lpdmvus[i];
            switch( lpdmvus2->dmTypeRef.dwType )
            {
                case DM_VARTYPE_BIT:
                    if (lpdmvus2->dmValue.boolVal)
                    {
                        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = TRUE"),
                            lpdmvus2->dmVarKey.szName,
                            lpdmvus2->dmVarKey.dwID);
                    }
                    else
                    {
                        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = FALSE"),
                            lpdmvus2->dmVarKey.szName,
                            lpdmvus2->dmVarKey.dwID);
                    }
                    break;
                case DM_VARTYPE_BYTE:
                    strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %u"),
                        lpdmvus2->dmVarKey.szName,
                        lpdmvus2->dmVarKey.dwID,
                        lpdmvus2->dmValue.bVal);
                    break;
                case DM_VARTYPE_SBYTE:

```

```

        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %d"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.bVal);
        break;
    case DM_VARTYPE_WORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %u"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.iVal);
        break;
    case DM_VARTYPE_SWORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %d"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.iVal);
        break;
    case DM_VARTYPE_DWORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %u"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.lVal);
        break;
    case DM_VARTYPE_SDWORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %d"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.lVal);
        break;
    case DM_VARTYPE_FLOAT:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %f"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.fltVal);
        break;
    case DM_VARTYPE_DOUBLE:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %f"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.dblVal);
        break;
    case DM_VARTYPE_TEXT_8:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %s"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.bstrVal);
        break;
/*
    case DM_VARTYPE_TEXT_16:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %s"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.bstrVal);
        break;
*/
    default:

```

```
        strData.Format(_T("Unbekannter Variablentyp !"));
    }
    PutStr(strData);

    // clear the given VARIANT's in every array element, because a VT_BSTR's can be
present    // do not only delete the array later without clearing the Variants here because
of memory leak's
        VariantClear(&(lpdmvus2->dmValue));
    }
    delete []lpdmVarKey;
    delete []lpdmvus;
}
//{{ODK_EXAMPLE}(END) }
```

See also

[DMGetValue \(Page 1687\)](#)

3.2.11.22 OnTestVariablenGetvaluawait (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestVariablenGetvaluawait (MCP)}
//{{FUNCTION}DMGetValueWait (MCP)}
//{{FUNCTION}(END)}
//-----< Notification-Callback für GetValueWait >-----
BOOL GetValueWaitNotify(DWORD dwTAID,
                        LPDM_VAR_UPDATE_STRUCT lpdmvus,
                        DWORD dwItems,
                        LPVOID lpvUser)
{
    dwTAID;
    CTestCliDoc* pDoc = (CTestCliDoc*)lpvUser;
    for(DWORD i=0; i < dwItems; i++)
    {
        LPDM_VAR_UPDATE_STRUCT lpdmvus2 = &lpdmvus[i];
        CString strData;
        switch(lpdmvus2->dmTypeRef.dwType)
        {
            case DM_VARTYPE_BIT:
                if (lpdmvus2->dmValue.boolVal)
                {
                    strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = TRUE,
Status=%04x"),
                                lpdmvus2->dmVarKey.szName,
                                lpdmvus2->dmVarKey.dwID,
                                lpdmvus2->dwState);
                }
                else
                {
                    strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = FALSE,
Status=%04x"),
                                lpdmvus2->dmVarKey.szName,
                                lpdmvus2->dmVarKey.dwID,
                                lpdmvus2->dwState);
                }
                break;
            case DM_VARTYPE_BYTE:
                strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %u, Status=
%04x"),
                                lpdmvus2->dmVarKey.szName,
                                lpdmvus2->dmVarKey.dwID,
                                lpdmvus2->dmValue.bVal,
                                lpdmvus2->dwState);
                break;
            case DM_VARTYPE_SBYTE:
                strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %d, Status=
%04x"),
                                lpdmvus2->dmVarKey.szName,
                                lpdmvus2->dmVarKey.dwID,
                                lpdmvus2->dmValue.bVal,
                                lpdmvus2->dwState);
        }
    }
}
```

3.2 Data management functions (RT Professional)

```

        break;
    case DM_VARTYPE_WORD:
        strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %u, Status=
%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.iVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_SWORD:
        strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %d, Status=
%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.iVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_DWORD:
        strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %u, Status=
%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.lVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_SDWORD:
        strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %d, Status=
%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.lVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_FLOAT:
        strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %f, Status=
%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.fltVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_DOUBLE:
        strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %f, Status=
%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.dblVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_TEXT_8:
        strData.Format(_T("GetValueWait: Variable: %s\t( ID = %d ):\tWert = %s, Status=
%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.bstrVal,
            lpdmvus2->dwState);

```



```

        break;
    case DM_VARTYPE_RAW:
    {
        LPBYTE pArray;
        HRESULT hr;
        hr = SafeArrayAccessData(lpdmvus2->dmValue.parray, (VOID **) &pArray);
        if (! FAILED(hr))
        {
            strData.Format(_T("Var:%s: Type:Rohdatum\nValues:%02x %02x %02x %02x\n"),
                lpdmvus2->dmVarKey.szName,
                pArray[0],
                pArray[1],
                pArray[2],
                pArray[3]);
            SafeArrayUnaccessData(lpdmvus2->dmValue.parray);
        }
        else
        {
            strData.Format(_T("SafeArrayAccessData fehlgeschlagen\n"));
        }
        break;
    }
    default:
        strData.Format(_T("GetValueWait: Unbekannter Variablentyp !"));
    }
    pDoc->PutStr(strData);
}
return(TRUE);
}

//-----< OnTestVariablenGetvaluawait >-----
void CTestCliDoc::OnTestVariablenGetvaluawait()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    LPDM_VAR_UPDATE_STRUCT lpdmvus = new DM_VAR_UPDATE_STRUCT[nNum];
    memset(lpdmvus, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    DWORD dwTAID = 5;
    DWORD dwTimeOut = 5000;
    CCmnError Error;
    memset(&Error, 0, sizeof(CCmnError));
    if(!DMGetValueWait(&dwTAID,
        lpdmVarKey,
        nNum,
        FALSE,
        dwTimeOut,
        GetValueWaitNotify,
        this,
        &Error))
    {
        Error.Show(_T("DMGetValueWait (FALSE) failed\n"));
    }
    delete []lpdmVarKey;
    delete []lpdmvus;
}

```

```
//-----< OnTestVariablenGetvaluawait >-----
void CTestCliDoc::OnTestVariablenGetvaluawaitTrue()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    LPDM_VAR_UPDATE_STRUCT lpdmvus = new DM_VAR_UPDATE_STRUCT[nNum];
    memset(lpdmvus, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    DWORD dwTAID = 5;
    DWORD dwTimeOut = 5000;
    CCmnError Error;
    memset(&Error, 0, sizeof(CCmnError));
    if( !DMGetValueWait(&dwTAID,
        lpdmVarKey,
        nNum,
        TRUE,
        dwTimeOut,
        GetValueWaitNotify,
        this,
        &Error))
    {
        Error.Show(_T("DMGetValueWait (TRUE) failed\n"));
    }
    delete []lpdmVarKey;
    delete []lpdmvus;
}
//{{ODK_EXAMPLE} (END) }
```

See also

[DMGetValueWait \(Page 1700\)](#)

3.2.11.23 OnTestVariablenGetVarInfo (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestVariablenGetVarInfo (MCP)}
//{{FUNCTION}DMGetVarInfo (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetVarInfo()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!::DMGetVarInfo(m_strProject, lpdmVarKey, nNum, &Error))
    {
        Error.Show(_T("DMGetVarInfo failed\n"));
    }
    else
    {
        CString strData;
        for(int i=0; i<nnum; i++)
        {
            strData.Format(_T("DMGetVarInfo: Variablenname: %s, VarID:%lu."),
                lpdmVarKey[i].szName,
                lpdmVarKey[i].dwID);
            PutStr(strData);
        }
        delete []lpdmVarKey;
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

DMGetVarInfo (Page 1706)

3.2.11.24 OnTestVariablenGetvarlimits (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestVariablenGetvarlimits (MCP)}
//{{FUNCTION}DMGetVarLimits (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetvarlimits()
{
    int nNum = GetVarCount();
    LPDM_VARLIMIT lpdmVarLimit = new DM_VARLIMIT[nNum];
    memset(lpdmVarLimit, 0, sizeof(DM_VARLIMIT) * nNum);
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    CCmnError Error;
    memset(&Error, 0, sizeof(CCmnError));
    if(!::DMGetVarLimits(m_strProject, lpdmVarKey, nNum,
        lpdmVarLimit, &Error))
    {
        Error.Show(__T("DMGetVarLimits failed\n"));
    }
    else
    {
        CString strData;
        for(int i=0; i<nnum; i++)
        {
            strData.Format(_T("DMGetVarLimits: Var:%s, MaxRange:%lf, MinRange:%lf, MinLimit:
%lf, MaxLimit:%lf."),
                lpdmVarKey[i].szName,
                lpdmVarLimit[i].dmMaxRange.dblVal,
                lpdmVarLimit[i].dmMinRange.dblVal,
                lpdmVarLimit[i].dmMaxLimit.dblVal,
                lpdmVarLimit[i].dmMinLimit.dblVal);
            PutStr(strData);
        }
    }
    delete []lpdmVarLimit;
    delete []lpdmVarKey;
}
//{{ODK_EXAMPLE}(END)}

```

See also

[DMGetVarLimits \(Page 1717\)](#)

3.2.11.25 OnTestVariablenGetvartype (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestVariablenGetvartype (MCP)}
//{{FUNCTION}DMGetVarType (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetvartype()
{
    int nNum = GetVarCount();
    LPDM_TYPEREF lpdmTypeRef = new DM_TYPEREF[nNum];
    memset(lpdmTypeRef, 0, sizeof(DM_TYPEREF) * nNum);
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    CCmnError Error;
    memset(&Error, 0, sizeof(CCmnError));
    if(!::DMGetVarType(m_strProject, lpdmVarKey, nNum,
        lpdmTypeRef, &Error))
    {
        Error.Show(_T("DMGetVarType failed\n"));
    }
    else
    {
        CString strData;
        for(int i=0; i<nnum; i++)
        {
            strData.Format(_T("DMGetVarType: Var:%s, Size:%lu, Name:%s."),
                lpdmVarKey[i].szName,
                lpdmVarKey[i].dwID);
            PutStr(strData);
        }
        delete []lpdmVarKey;
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

[DMGetVarType \(Page 1724\)](#)

3.2.11.26 OnTestVariablenResumevarupdate (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestVariablenResumevarupdate (MCP)}  
//{{FUNCTION}DMResumeVarUpdate (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestVariablenResumevarupdate()  
{  
    CCmnError Error;  
    memset(&Error,0,sizeof(CCmnError));  
    if(!DMResumeVarUpdate(m_dwTAID, &Error))  
    {  
        Error.Show(_T("DMResumeVarUpdate failed.\n"));  
    }  
    else  
    {  
        CString strData;  
        strData.Format(_T("DMResumeVarUpdate ok."));  
    }  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

[DMResumeVarUpdate \(Page 1800\)](#)

3.2.11.27 OnTestVariablenSetvalue (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestVariablenSetvalue (MCP)}
//{{FUNCTION}DMSetValue (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenSetvalue()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    LPDWORD lpdmVarState = new DWORD[nNum];
    INT i;
    CString strData;
    CCmnError cmnError;
    for (i = 0; i < 1; i++)</p>
    {
        DWORD dwStart = GetTickCount();
        DWORD dwEnd;
        if (! DMSetValue(lpdmVarKey, nNum, m_varValues, lpdmVarState, cmnError))
        {
            break;
        }
        dwEnd = GetTickCount();
        strData.Format("DMSetValue OK, Dauer %d ms", dwEnd - dwStart);
        PutStr(strData);
    }
    delete []lpdmVarKey;
    delete []lpdmVarState;
}
//{{ODK_EXAMPLE}(END)}

```

See also

[DMSetValue \(Page 1730\)](#)

3.2.11.28 OnTestVariablenSetvaluawait (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestVariablenSetvaluawait (MCP)}  
//{{FUNCTION}DMSetValueWait (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestVariablenSetvaluawait()  
{  
    int nNum = GetVarCount();  
    LPDM_VARKEY lpdmVarKey = GetVarKeys();  
    LPDWORD lpdmVarState = new DWORD[nNum];  
    DWORD dwTimeOut = 1000L;  
    CCmnError cmnError;  
    if(!::DMSetValueWait(&m_dwTAID, lpdmVarKey, nNum, m_varValues, dwTimeOut,  
        CompletionProc, this, cmnError))  
    {  
        cmnError.Show(_T("DMSetValueWait failed\n"));  
    }  
    else  
    {  
        CString strData;  
        strData.Format(_T("DMSetValueWait ok.));  
    }  
    delete []lpdmVarKey;  
    delete []lpdmVarState;  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

[DMSetValueWaitMessage \(Page 1745\)](#)

[DMSetValueWait \(Page 1740\)](#)

3.2.11.29 OnTestVariablenStopallupdates (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestVariablenStopallupdates (MCP)}
//{{FUNCTION}DMStopAllUpdates (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenStopallupdates()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMStopAllUpdates(&Error))
    {
        Error.Show(_T("DMStopAllUpdates failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMStopAllUpdates ok."));
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

DMStopAllUpdates (Page 1816)

3.2.11.30 OnTestVariablenStopvarupdate (RT Professional)**Example**

```

//{{ODK_EXAMPLE}OnTestVariablenStopvarupdate (MCP)}
//{{FUNCTION}DMStopVarUpdate (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenStopvarupdate()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !DMStopVarUpdate(m_dwTAID, &Error))
    {
        Error.Show(_T("DMStopVarUpdate failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMStopVarUpdate: TAID:%lu."), m_dwTAID);
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

[DMStopVarUpdate \(Page 1817\)](#)

3.2.11.31 OnTestVariablenSuspendvarupdate (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestVariablenSuspendvarupdate (MCP)}  
//{{FUNCTION}DMSuspendVarUpdate (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestVariablenSuspendvarupdate()  
{  
    CCmnError Error;  
    memset(&Error,0,sizeof(CCmnError));  
    if(!DMSuspendVarUpdate(m_dwTAID, &Error))  
    {  
        Error.Show(_T("DMSuspendVarUpdate failed\n"));  
    }  
    else  
    {  
        CString strData;  
        strData.Format(_T("DMSuspendVarUpdate ok."));  
    }  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

[DMSuspendVarUpdate \(Page 1819\)](#)

3.2.11.32 OnTestWinCCShutdown (RT Professional)

Example

```
//{{ODK_EXAMPLE}OnTestWinCCShutdown (MCP)}  
//{{FUNCTION}DMEExitWinCC (MCP)}  
//{{FUNCTION}(END)}  
void CTestCliDoc::OnTestWinCCShutdown()  
{  
    DMEExitWinCC();  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

[DMEExitWinCC \(Page 1640\)](#)

3.2.11.33 Open project by means of dialog (RT Professional)

Example

```
//{{ODK_EXAMPLE}Open projectby means of dialog (MCP)}
//{{FUNCTION}DMOpenProject (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: MyDMOpenProject(void) ODK DM CS
// =====
// Desc. : open project, give name through WinCC-Dialog-Box
// =====
void MyDMOpenProject(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwSize = 255;
    HWND handle = NULL;
    TCHAR szProjFile[255];
    TCHAR szText[255];
    _tcsncpy_s(szProjFile, _countof(szProjFile), _T("C:\\Siemens\\ODK\\Samples\\Projects\\
\\Demo\\odk.mcp"), _TRUNCATE);
    memset(&Error, 0, sizeof( CMN_ERROR ));
    ret = MyDMConnect(); // check connection to DataManager
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in MyDMConnect"));
        ODKTrace(szText);
    }
    else
    {
        memset(&Error, 0, sizeof( CMN_ERROR ));
        ret = DMOpenProject(handle, szProjFile, dwSize, &Error);
        if(ret == FALSE)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMOpenProject (%s)",
szProjFile);
        }
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

DMOpenProjectPlus (Page 1660)

3.2.11.34 Read tag (RT Professional)

Example

```
// =====
// =====
// short : Modul with examples to DataManager-API
// RUNTIME
// SC: PO = Projekt opened PG= Projekt closed
// *****
#include "stdafx.h" // if MFC classes
#include "odkapi.h" // if console application
#include <time.h>
#include "dm02.h"

// =====
// 2.1 Interface IMPORT
// =====
//extern void ODKTrace(LPCTSTR);
extern BOOL MyDMConnect (void); // Connect
extern BOOL MyDMGetConnectionState(void);

// =====
// 2.2 Interface EXPORT
// =====
// =====
// 2.3 Interface LOCAL
// =====
// =====
// 3. Definitions
// =====

//{{ODK_EXAMPLE}Read tag (MCP)}
//{{FUNCTION}DMGetRuntimeProject (MCP)}
//{{FUNCTION}DMGetValue (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: MyDMGetValue(void) ODK DM CS
// =====
// short : Read tag
// =====
void MyDMGetValue(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwSize = _MAX_PATH;
    TCHAR szText[255];
    TCHAR szProjFile[_MAX_PATH + 1];
    TCHAR BstrValue[255]; //
    const short int nNum = 11; // tagcount
    const short int nTextMax = 20; // textlength
    TCHAR TagNames[nNum][nTextMax]=
    {
        "VAR_1_BOOL", // boolVal; // VT_BOOL.
```

```

    "VAR_2_BYTE", // bVal; // VT_UI1
    "VAR_3_SBYTE", // iVal; // VT_I2
    "VAR_4_WORD", // lVal // VT_I4
    "VAR_5_SWORD", // iVal // VT_I2
    "VAR_6_DWORD", // dblVal // VT_R8
    "VAR_7_SDWORD", // lVal // VT_I4
    "VAR_8_FLOAT", // fltVal // VT_R4
    "VAR_9_DOUBLE", // dblVal // VT_R8
    "VAR_A_TEXT8", // bstrVal // VT_BSTR
    "VAR_B_TEXT16" // bstrVal // VT_BSTR
};
DM_VARKEY VarKey[nNum];

DM_VAR_UPDATE_STRUCT VarUp[nNum];
memset(&VarUp, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
// the included VARIANT's in the DM_VAR_UPDATE_STRUCT's then initialized to VT_EMPTY
with the memset,
// don't do this later again, because VT_BSTR's can be present after DMGetValue(...)!

memset(&VarKey, 0, sizeof(DM_VARKEY) * nNum);
memset(&Error, 0, sizeof(Error));
ret = MyDMGetConnectionState();
if(FALSE != ret)
{
    memset(&Error, 0, sizeof(Error));
    // RunTime project
    ret = DMGetRuntimeProject(szProjFile, dwSize, &Error);
    if(FALSE != ret)
    {
        // fill out DM_VARKEY
        for(int iRead = 0; iRead < nNum; iRead++)
        {
            VarKey[iRead].dwKeyType = DM_VARKEY_NAME;
            VarKey[iRead].dwID = 0;
            strcpy( VarKey[iRead].szName, &TagNames[iRead][0]);
            VarKey[iRead].lpvUserData = (VOID *) iRead;
        }
        memset(&Error, 0, sizeof(Error));
        ret = DMGetValue(VarKey, nNum, VarUp, &Error);
        if(FALSE != ret)
        {
            for(int iOut=0; iOut < nNum; iOut++)
            {
                switch(VarUp[iOut].dmTypeRef.dwType)
                {
                    {
                        case DM_VARTYPE_BIT: //vt = 3 VT_I4 = 3
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                iOut, VarUp[iOut].dmVarKey.szName,
                                VarUp[iOut].dmVarKey.dwID,
                                VarUp[iOut].dmValue.boolVal);
                            break;
                        }
                        case DM_VARTYPE_BYTE: //vt = 17 VT_UI1 = 17
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",

```

3.2 Data management functions (RT Professional)

```

                                iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                VarUp[iOut].dmValue.bVal);
                                break;
                                }
                                case DM_VARTYPE_SBYTE: //vt = 2 VT_I2 = 2
                                {
                                    sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                        iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                        VarUp[iOut].dmValue.iVal);
                                    break;
                                }
                                case DM_VARTYPE_WORD: //vt = 3 VT_I4 = 3
                                {
                                    sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                        iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                        VarUp[iOut].dmValue.lVal);
                                    break;
                                }
                                case DM_VARTYPE_SWORD: //vt = 2 VT_I2 = 2
                                {
                                    sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                        iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                        VarUp[iOut].dmValue.iVal);
                                    break;
                                }
                                case DM_VARTYPE_DWORD: //vt = 5 VT_R8 = 5
                                {
                                    sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                        iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                        VarUp[iOut].dmValue.dblVal);
                                    break;
                                }
                                case DM_VARTYPE_SDWORD: //vt = 3 VT_I4 = 3
                                {
                                    sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                        iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                        VarUp[iOut].dmValue.lVal);
                                    break;
                                }
                                case DM_VARTYPE_FLOAT: //vt = 4 VT_R4 = 4
                                {
                                    sprintf(szText, "Index=%d Name=%s ID=%d Value=%f",
                                        iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                        VarUp[iOut].dmValue.fltVal);
                                    break;
                                }
                                case DM_VARTYPE_DOUBLE: //vt = 5 VT_R8 = 5
                                {
                                    sprintf(szText, "Index=%d Name=%s ID=%d Value=%f",

```

```

        iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
        VarUp[iOut].dmValue.dblVal);
        break;
    }
    case DM_VARTYPE_TEXT_8: //vt = 8 VT_BSTR = 8
    {
        ret = WideCharToMultiByte( CP_ACP,
(DWORD)0,VarUp[iOut].dmValue.bstrVal,
                                -1, (LPSTR)&BstrValue[0],128,NULL,NULL);
        sprintf(szText, "Index=%d Name=%s ID=%d StrValue=%s",
            iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,BstrValue);
        break;
    }
    case DM_VARTYPE_TEXT_16: // vt = 8 VT_BSTR = 8
    {
        ret = WideCharToMultiByte( CP_ACP,
(DWORD)0,VarUp[iOut].dmValue.bstrVal,
                                -1, (LPSTR)&BstrValue[0],128,NULL,NULL);
        sprintf(szText, "Index=%d Name=%s ID=%d StrValue=%s",
            iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,BstrValue);
        break;
    }
    default:
        break;
    } // end switch case
    ODKTrace(szText);
    //printf("%s\r\n",szText);

    VariantClear( &VarUp[iOut].dmValue );
} //end for
}
else
{
    sprintf(szText, "Error in DMGetValue: E1= 0x%08lx ; E2= 0x%08lx ; %s",
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    //printf("%s\r\n",szText);
}
}
else
{
    sprintf(szText, "Error in DMGetRuntimeProject: E1= 0x%08lx ; E2= 0x%08lx ; %s",
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    //printf("%s\r\n",szText);
}
}
else
{
    sprintf(szText, "Error in MyDMGetConnectionState: E1= 0x%08lx ; E2= 0x%08lx ; %s",
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    //printf("%s\r\n",szText);
}
}

```

```
    }  
}  
// { {ODK_EXAMPLE} (END) }
```

See also

[DMGetRuntimeProject \(Page 1658\)](#)

[DMGetValue \(Page 1687\)](#)

3.2.11.35 Write tag (RT Professional)

Example

```
//{{ODK_EXAMPLE}Write tag (MCP)}
//{{FUNCTION}DMGetRuntimeProject (MCP)}
//{{FUNCTION}DMSetValue (MCP)}
//{{FUNCTION}(END)}
// =====
// Function: MyDMSetValue(void) ODK DM CS
// =====
// short : Write tag
// =====
void MyDMSetValue(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwSize = _MAX_PATH;
    TCHAR szText[255];
    TCHAR szProjFile[_MAX_PATH + 1];
    TCHAR szTextValue[255]; // for TEXT8 Values
    const short int nNum = 11; // tagcount
    const short int nTextMax = 20; // textlength
    TCHAR TagNames[nNum][nTextMax] =
    {
        "VAR_1_BOOL", // boolVal; // VT_BOOL.
        "VAR_2_BYTE", // bVal; // VT_UI1
        "VAR_3_SBYTE", // iVal; // VT_I2
        "VAR_4_WORD", // lVal // VT_I4
        "VAR_5_SWORD", // iVal // VT_I2
        "VAR_6_DWORD", // dblVal // VT_R8
        "VAR_7_SDWORD", // lVal // VT_I4
        "VAR_8_FLOAT", // fltVal // VT_R4
        "VAR_9_DOUBLE", // dblVal // VT_R8
        "VAR_A_TEXT8", // bstrVal // VT_BSTR
        "VAR_B_TEXT16" // bstrVal // VT_BSTR
    };
    DM_VARKEY VarKey[nNum];
    DM_VAR_UPDATE_STRUCT VarUp[nNum];
    VARIANT VarVal[nNum];
    DWORD VarSta[nNum];
    static short int Tmp[20];
    BSTR bstrText1, bstrText2;
    memset(&Error, 0, sizeof(CMN_ERROR));
    memset(&VarUp, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    memset(&VarKey, 0, sizeof(DM_VARKEY) * nNum);
    ret = MyDMGetConnectionState();
    if(FALSE != ret)
    {
        // RunTime project
        ret = DMGetRuntimeProject(szProjFile, dwSize, &Error);
        if(FALSE != ret)
        {
            // read tag
```

3.2 Data management functions (RT Professional)

```

for(int i = 0; i < nNum; i++)
{
    VarKey[i].dwKeyType = DM_VARKEY_NAME;
    VarKey[i].dwID = 0;
    _tcsncpy_s(VarKey[i].szName, _countof(VarKey[i].szName), &TagNames[i][0],
_TRUNCATE);
    VarKey[i].lpvUserData = (VOID *) i;
}
// init tag
// "VAR_1_BOOL",
VarVal[0].vt = VT_BOOL;
VarVal[0].boolVal = TRUE;
// "VAR_2_BYTE"
VarVal[1].vt = VT_UI1;
VarVal[1].bVal = 23;
// "VAR_3_SBYTE",
VarVal[2].vt = VT_I2;
VarVal[2].iVal = -23;
// "VAR_4_WORD",
VarVal[3].vt = VT_I4;
VarVal[3].lVal = 89;
// "VAR_5_SWORD",
VarVal[4].vt = VT_I2;
VarVal[4].iVal = -89;
// "VAR_6_DWORD",
VarVal[5].vt = VT_R8;
VarVal[5].dblVal = 1200;
// "VAR_7_SDWORD",
VarVal[6].vt = VT_I4;
VarVal[6].lVal = -1200;
// "VAR_8_FLOAT",
VarVal[7].vt = VT_R4;
VarVal[7].fltVal = (float)3.789;
// "VAR_9_DOUBLE",
VarVal[8].vt = VT_R8;
VarVal[8].dblVal = (double)34.789;
// "VAR_A_TEXT8",
_tcsncpy_s(szTextValue, _countof(szTextValue), _T("VAR_A_TEXT8 Value"),
_TRUNCATE);
ret = MultiByteToWideChar(CP_ACP, (DWORD)0, (LPCSTR) &szTextValue, -1,
(LPWSTR) &Tmp[0], 30);
// SysFreeString(pBSTR);
bstrText1 = SysAllocString((OLECHAR FAR*)&Tmp[0]);
VarVal[9].vt = VT_BSTR;
VarVal[9].bstrVal = bstrText1;
// "VAR_B_TEXT16"
_tcsncpy_s(szTextValue, _countof(szTextValue), _T("VAR_B_TEXT16 Value"),
_TRUNCATE);
ret = MultiByteToWideChar(CP_ACP, (DWORD)0, (LPCSTR) &szTextValue, -1,
(LPWSTR) &Tmp[0], 30);
// SysFreeString(pB2);
bstrText2 = SysAllocString((OLECHAR FAR*)&Tmp[0]);
VarVal[10].vt = VT_BSTR;
VarVal[10].bstrVal = bstrText2;
// set tag
memset(&Error, 0, sizeof(CMN_ERROR));

```

```

ret = DMSetValue(&VarKey[0], nNum, &VarVal[0], &VarSta[0], &Error);
if (FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMSetValue:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
    Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("write Value to WinCC"));
    ODKTrace(szText);
}
SysFreeString(bstrText1);
SysFreeString(bstrText2);
}
}
}
//{{ODK_EXAMPLE}(END)}

```

See also

DMGetRuntimeProject (Page 1658)

DMSetValue (Page 1730)

3.3 Functions of the graphic system (RT Professional)**3.3.1 Basics (RT Professional)****3.3.1.1 Overview of the functions (RT Professional)****Overview**

PDLRTShowApp (Page 1912)	Display graphic system
PDLRT_CALLBACK (Page 1913)	Callback function of the runtime functions
PDLRTCclosePicture (Page 1900)	Close screen
PDLRTDisableClosePicture (Page 1902)	Disable close screen function
PDLRTEnableClosePicture (Page 1904)	Enable close screen function
PDLRTGetCursorKeys (Page 1915)	Query cursor control keys

3.3 Functions of the graphic system (RT Professional)

PDLRTGetDefPropEx (Page 1923)	Query default values of object properties
PDLRTGetPropEx (Page 1925)	Query default values of object properties
PDLRTGetFocus (Page 1917)	Query input focus
PDLRTGetLink (Page 1933)	Query link between object property and tag
PDLRTGotoPicture (Page 1905)	Query screen memory
PDLRTInquireFreeArea (Page 1907)	Determine free area of a window
PDLRTOpenPicture (Page 1908)	Root screen switch in Runtime
PDLRTPictureNavigation (Page 1911)	Enable and disable screen navigation
PDLRTSetCursorKeys (Page 1919)	Define cursor control keys
PDLRTSetFocus (Page 1921)	Define input focus
PDLRTSetLink (Page 1935)	Define link between object property and tag
PDLRTSetMultiLink (Page 1937)	Define link between object property and tag (multiple tags)
PDLRTSetPropEx (Page 1928)	Define object property

3.3.1.2 Overview of structures (RT Professional)

Overview

FOCUSINFO (Page 1900)	Input focus data
LINKINFO (Page 1896)	Linking of object properties and tags
MULTILINK (Page 1898)	Linking of object properties and tags
MULTILINKINFO (Page 1899)	Linking of object properties and tags

3.3.1.3 Error messages (RT Professional)

Overview

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

Configuration

PDLCS_ELIASES_NO_ERROR	0	No error occurred.
PDLCS_ERROR_NO_DISPATCH	5	Ole Automation cannot be run
PDLCS_ERROR_NO_PROJECT	10	Project does not exist
PDLCS_ERROR_NO_PICTURE	15	Screen does not exist
PDLCS_ERROR_NO_OBJECT	20	Object does not exist
PDLCS_ERROR_NO_PROPERTY	25	Property does not exist

3.3 Functions of the graphic system (RT Professional)

PDLCS_ERROR_NO_OBJTYPE	30	Object type not present.
PDLCS_ERROR_NO_DYNAMIC	32	No dynamics available
PDLCS_ERROR_ALREADY_OPEN	35	Specified screen is already open
PDLCS_ERROR_WRITE_PROTECTED	40	Specified screen is read-only
PDLCS_ERROR_PICTURE_SAVE	45	Specified screen could not be saved
PDLCS_ERROR_PICTURE_SAVE_AS	50	Specified screen could not be saved
PDLCS_ERROR_PICTURE_CLOSE	55	Specified screen could not be closed
PDLCS_ERROR_PICTURE_CLOSE_ALL	60	Screens could not be closed
PDLCS_ERROR_PICTURE_IMPORT	65	Screens could not be imported
PDLCS_ERROR_PICTURE_CREATE	70	General error when creating the screen
PDLCS_ERROR_PICTURE_EXISTS	75	Screen already exists
PDLCS_ERROR_NEW_OBJECT_NAME	80	Object name already exists
PDLCS_ERROR_NEW_OBJECT	85	Object could not be created
PDLCS_ERROR_DELETE_OBJECT	90	Object could not be deleted
PDLCS_ERROR_SET_PROPERTY	95	Property could not be set
PDLCS_ERROR_GET_PROPERTY	96	Property could not be read
PDLCS_ERROR_ENUM_OBJECTTYPE	100	Error while enumerating object types
PDLCS_ERROR_ENUM_PROPERTY	105	Error while enumerating object properties
PDLCS_ERROR_CREATE_PROPERTY	110	Property could not be defined
PDLCS_ERROR_SET_LINK	115	Connection could not be established
PDLCS_ERROR_REMOVE_LINK	120	Connection could not be deleted
PDLCS_ERROR_SET_ACTION	125	Action could not be stored
PDLCS_ERROR_GET_ACTION	126	Action could not be fetched
PDLCS_ERROR_ENUM_DYNAMICS	127	Dynamics could not be enumerated
PDLCS_ERROR_SET_DIRECTCONNECT	128	Direct connection could not be stored
PDLCS_ERROR_AUTOMATION	130	Error during transfer
PDLCS_ERROR_MEMORY	200	No memory

Runtime

PDLRT_OK	0	No error occurred.
PDLRT_APP_NOT_RUNNING	1	PDL application not started
PDLRT_NO_TRANSFER	2	PDL application cannot be accessed
PDLRT_NO_PIC	3	Screen not selected
PDLRT_NO_OBJ	4	Cannot find object
PDLRT_NO_PROP	5	Cannot find property
PDLRT_NO_MET	6	Cannot find method for object
PDLRT_LINK_NOT_SET	7	Property could not be connected
PDLRT_WNAME_NOT_UNIQUE	8	Instance name of new window not unique
PDLRT_WCREATION_FAILED	9	Window could not be created
PDLRT_PICTURE_NOT_LOADED	10	Screen file could not be loaded
PDLRT_ILLEGAL_USERLEVEL	11	Insufficient operator authorization level
PDLRT_NO_POSITION	12	Rectangle cannot be placed

3.3 Functions of the graphic system (RT Professional)

PDLRT_POSITION_WITH_MODY	13	Rectangle was modified
PDLRT_E_ALREADY_CONNECTED	14	Another connection to the PDLRT already exists
PDLRT_PICTURE_ALREADY_OPEN	15	Screen is already open
PDLRT_BAD_OLE_CONVERSION	16	Error during conversion via OLE Automation
PDLRT_NO_LINK	17	Property has no dynamics
PDLRT_IND_LINK_READ_ERR	18	Error while reading the indirect tag
PDLRT_NO_IND_LINK	19	Property has no indirect dynamics
PDLRT_FAILURE_PARAM	20	Wrong parameter

3.3.1.4 Constants (RT Professional)

Addressing mode

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	1	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The PDLRT_AM_PICTABS and PDLRT_AM_OBJABS values can be ORed.

PDLRTInquireFreeArea

PDLRT_IQ_ONLY	0x01	Only inquiry with result: "fits/does not fit"
PDLRT_IQ_MODY_POSX	0x02	The x-position of the rectangle may be modified so that it can be placed
PDLRT_IQ_MODY_POSY	0x04	The y-position of the rectangle may be modified so that it can be placed
PDLRT_IQ_MODY_HEIGHT	0x08	The height of the rectangle may be modified so that it can be placed
PDLRT_IQ_MODY_WIDTH	0x10	The width of the rectangle may be modified so that it can be placed
PDLRTOpenPicture		
PDLRT_WS_STYLE_FROM_PIC	0	The display attributes are taken from the screen data
PDLRT_WS_BORDER	WS_BORDER	Window with frame
PDLRT_WS_CAPTION	WS_CAPTION	Window with title and frame
PDLRT_WS_THICKFRAME	WS_THICKFRAME	Window with thick frame. Size can be changed via mouse operation.
PDLRT_WS_HSCROLL	WS_HSCROLL	Window with horizontal slider
PDLRT_WS_VSCROLL	WS_VSCROLL	Window with vertical slider

Update cycles

Name	Index
"On change"	0

"250 ms"	1
"500 ms"	2
"1 s"	3
"2 s"	4
"5 s"	5
"10 s"	6
"1 min"	7
"5 min"	8
"10 min"	9
"1 h"	10
"User cycle 1"	11
"User cycle 2"	12
"User cycle 3"	13
"User cycle 4"	14
"User cycle 5"	15
"Window cycle"	235
"Screen cycle"	255

3.3.1.5 List of object properties (A-K) (Graphics Designer) (RT Professional)

Overview

Note

For many API functions of the Graphics Designer, you must initialize the `lpszPropName` parameter. The English name of an object property must be specified for this.

Properties of type `VT_USERDEFINED` must not be processed with the Get/Set Property functions.

`VT_DISPATCH` and other references must not be used; only the standard types and at most one `VT_VARIANT` for an array property of simple types may be used.

OLE Automation name	Data type		Property name	Attribute name	Object association
"ActualPointLeft"	VT_I4		Current Value X	Current Value X	Polygon
"ActualPointTop"	VT_I4		Current Value Y	Current Value Y	Polygon
"AdaptBorder"	VT_BOOLEAN		Adapt Border	Adapt Border	Button, static text, I/O field, text list, check box, radio box
"AdaptPicture"	VT_BOOLEAN		Adapt Picture	Adapt Picture	Screen window

3.3 Functions of the graphic system (RT Professional)

"AdaptSize"	VT_BOO L		Adapt Size	Adapt Size	Screen window
"AdaptWidth"					
"AlarmHigh"	VT_R8		Limit AH	Limit AH	Bar
"AlarmLow"	VT_R8		Limit AL	Limit AL	Bar
"Alignment"	VT_BOO L		Alignment	Alignment	Bar
"AlignmentLeft"	VT_I4		X Alignment	X Alignment	Button, static text, I/O field, group display, text list, check box, radio box
"AlignmentTop"	VT_I4		Y Alignment	Y Alignment	Button, static text, I/O field, group display, text list, check box, radio box
"Analog"	VT_BOO L	OCX	Analog	Analog	DAClockCtrl
"AngleAlpha"	VT_I4	BPC	Alpha	Alpha	3D bar
"AngleBeta"	VT_I4	BPC	Beta	Beta	3D bar
"AngleMax"	VT_R8	OCX	AngleMax	AngleMax	XGaugeCtrl
"AngleMin"	VT_R8	OCX	AngleMin	AngleMin	XGaugeCtrl
"Application"	VT_BST R		WindowCon- tent	WindowContent	Print job/Script diag- nostics
"ApplicationData- Path"					
"Assignments"	VT_BST R		Assignments	Assignments	Text list
"AssumeOnExit"	VT_BOO L		Apply on Exit	Apply on Exit	I/O field, text list
"AssumeOnFull"	VT_BOO L		Apply on Full	Apply on Full In- put	I/O field
"Autoscroll"	VT_BOO L	OCX	AutoScroll	AutoScroll	CCAlgWinCtrl
"AutoSize"	VT_I4	OCX	AutoSize	AutoSize	PButtonCtrl
"Average"	VT_BOO L		Average Value	Average Value	Bar
"Axe"	VT_I4				
"AxisSection"	VT_R8		Axis Section	Axis Section	Bar
"BackBorderWidth"	VT_I4		3D Border Weight	3D Border Weight	Button, group display, round button, slider ob- ject

3.3 Functions of the graphic system (RT Professional)

"BackColor"	VT_I4		Background Color	Background Color	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, 3D bar, I/O field, bar, graphic object, group display, text list, check box, radio box, round button, slider object
"BackColor"	VT_UI4		BackColor	BackColor	DAClockCtrl, XGaugeCtrl, PButtonCtrl, SliderCtrl
"BackColor2"	VT_I4		Bar Color	Bar Color	Bar
"BackColor3"	VT_I4		Bar Background Color	Bar Background Color	Bar
"BackColorBottom"	VT_I4		Low Limit Background Color	Low Limit Background Color	Slider object
"BackColorTop"	VT_I4		High Limit Background Color	High Limit Background Color	Slider object
"BackFlashColorOff"	VT_I4		Flashing Background Color Off	Flashing Background Color Off	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"BackFlashColorOn"	VT_I4		Flashing Background Color On	Flashing Background Color On	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"Background"	VT_BOOL		Background	Background	3D bar
"BackgroundPicture"	VT_USERDEFINED		Background-Picture	BackgroundPicture	XGaugeCtrl
"BarBackColor"	VT_UI4		BarBackColor	BarBackColor	SliderCtrl
"BarFillColor"	VT_UI4		BarFillColor	BarFillColor	SliderCtrl
"BarDepth"	VT_I4		Bar Depth	Bar Depth	3D bar
"BarHeight"	VT_I4		Bar Height	Bar Height	3D bar
"BarWidth"	VT_I4		Bar Width	Bar Width	3D bar
"BasePicReferenced"	VT_BOOL		Basic Picture Referenced	Basic Picture Referenced	Status display

3.3 Functions of the graphic system (RT Professional)

"BasePicTransColor"	VT_I4		Basic Picture Transparent Color	Basic Picture Transparent Color	Status display
"BasePicture"	VT_BSTR		Basic Picture	Basic Picture	Status display
"BasePicUseTransColor"	VT_BOOL		Basic Picture Transparent Color On	Basic Picture Transparent Color On	Status display
"BaseX"	VT_I4		Base X	Base X	3D bar
"BaseY"	VT_I4		Base Y	Base Y	3D bar
"Beveler"	VT_I4		Beveler	Beveler	SliderCtrl
"BevelColorDown"	VT_UI4		BevelColorDown	BevelColorDown	SliderCtrl
"BevelColorUp"	VT_UI4		BevelColorUp	BevelColorUp	SliderCtrl
"BevelOuter"	VT_I4		BevelOuter	BevelOuter	XGaugeCtrl
"BevelWidth"	VT_I4		BevelWidth	BevelWidth	XGaugeCtrl
"BitNumber"	VT_I4			Bit number	Status display
"BorderBackColor"	VT_I4		Border Background Color, Line Background Color	Border Background Color, Line Background Color	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"BorderColor"	VT_I4		Border Color, Line Background Color	Border Color, Line Background Color	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, I/O field, bar, graphic object, text list, status display, check box, radio box, round button, slider object
"BorderColorBottom"	VT_I4		3D Shadow Color	3D Shadow Color	Button, round button
"BorderColorTop"	VT_I4		3D Border Color	3D Border Color	Button, round button
"BorderEndStyle"	VT_UI4		Line End Style	Line End Style	Line, polyline

3.3 Functions of the graphic system (RT Professional)

"BorderFlashColor-Off"	VT_I4		Flashing Border Color Off, Flashing Line Color Off	Flashing Border Color Off, Flashing Line Color Off	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, slider object
"BorderFlashColor-On"	VT_I4		Flashing Border Color On, Flashing Line Color On	Flashing Border Color On, Flashing Line Color On	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"BorderStyle"	VT_UI4		Line Style	Line Style	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, I/O field, graphic object, text list, status display, check box, radio box, round button, slider object
"BorderWidth"	VT_I4		Line Weight	Line Weight	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, I/O field, bar, graphic object, text list, status display, check box, radio box, round button, slider object
"BorderWidth"	VT_I4	OCX	BorderWidth	BorderWidth	XGaugeCtrl
"BoxAlignment"	VT_BOOL		Box Alignment	Box Alignment	Check box, radio box
"BoxCount"	VT_I4		Number of Boxes	Number of Boxes	Check box, radio box
"BoxType"	VT_I4		Field Type	Field Type	I/O field, text list
"Button1Width"	VT_I4		Button 1 Width	Button 1 Width	Group display

3.3 Functions of the graphic system (RT Professional)

"Button2Width"	VT_I4		Button 2 Width (Error: Button 1 Width 2)	Button 2 Width	Group display
"Button3Width"	VT_I4		Button 3 Width (Error: Button 1 Width 3)	Button 3 Width	Group display
"Button4Width"	VT_I4		Button 4 Width	Button 4 Width	Group display
"ButtonColor"	VT_I4		Button Color	Button Color	Slider object
"ButtonCommand"	VT_I4	OCX	ButtonCommand	ButtonCommand	CCAlgWinCtrl
"Caption"	VT_BOOL		Title	Title	Print job/Script diagnostics, screen window
"Caption"	VT_BOOL	OCX	Caption	Caption	XGaugeCtrl, PButtonCtrl, SliderCtrl
"CaptionColor"	VT_UI4	OCX	CaptionColor	CaptionColor	XGaugeCtrl
"CaptionFont"	VT_USEDEFINED	OCX	CaptionFont	CaptionFont	XGaugeCtrl
"CaptionOffset"	VT_R8	OCX	CaptionOffset	CaptionOffset	XGaugeCtrl
"CellCut"	VT_BOOL	OCX	CellCut	CellCut	CCAlgWinCtrl
"CenterColor"	VT_UI4	OCX	CenterColor	CenterColor	XGaugeCtrl
"CenterScale"	VT_R4	OCX	CenterScale	CenterScale	XGaugeCtrl
"CheckAlarmHigh"	VT_BOOL		Monitoring AH	Monitoring AH	Bar
"CheckAlarmLow"	VT_BOOL		Monitoring AL	Monitoring AL	Bar
"CheckLimitHigh4"	VT_BOOL		Monitoring RH4	Monitoring RH4	Bar
"CheckLimitHigh5"	VT_BOOL		Monitoring RH5	Monitoring RH5	Bar
"CheckLimitLow4"	VT_BOOL		Monitoring RL4	Monitoring RL4	Bar
"CheckLimitLow5"	VT_BOOL		Monitoring RL5	Monitoring RL5	Bar
"CheckTolerance-High"	VT_BOOL		Monitoring TH	Monitoring TH	Bar
"CheckTolerance-Low"	VT_BOOL		Monitoring TL	Monitoring TL	Bar
"CheckWarningHigh"	VT_BOOL		Monitoring WH	Monitoring WH	Bar
"CheckWarningLow"	VT_BOOL		Monitoring WL	Monitoring WL	Bar
"ClearOnError"	VT_BOOL		Clear on Invalid Input	Clear on Invalid Input	I/O field
"ClearOnNew"	VT_BOOL		Clear on New Input	Clear on New Input	I/O field
"CloseButton"	VT_BOOL		Can be Closed	Can be Closed	Print job/Script diagnostics, screen window

3.3 Functions of the graphic system (RT Professional)

"CollectValue"	VT_UI4		Group Value	Group Value	Group display
"ColMove"	VT_BOOLEAN	OCX	ColMove	ColMove	CCAlgWinCtrl
"ColorAlarmHigh"	VT_I4		Bar Color AH	Bar Color AH	Bar
"ColorAlarmLow"	VT_I4		Bar Color AL	Bar Color AL	Bar
"ColorBottom"	VT_I4		Low Limit Color	Low Limit Color	Slider object
"ColorChangeType"	VT_BOOLEAN		Change Color	Change Color	Bar
"ColorLimitHigh4"	VT_I4		Bar Color RH4	Bar Color RH4	Bar
"ColorLimitHigh5"	VT_I4		Bar Color RH5	Bar Color RH5	Bar
"ColorLimitLow4"	VT_I4		Bar Color RL4	Bar Color RL4	Bar
"ColorLimitLow5"	VT_I4		Bar Color RL5	Bar Color RL5	Bar
"ColorTolerance-High"	VT_I4		Bar Color TH	Bar Color TH	Bar
"ColorTolerance-Low"	VT_I4		Bar Color TL	Bar Color TL	Bar
"ColorTop"	VT_I4		High Limit Color	High Limit Color	Slider object
"ColorWarningHigh"	VT_I4		Bar Color WH	Bar Color WH	Bar
"ColorWarningLow"	VT_I4		Bar Color WL	Bar Color WL	Bar
"ColTitle"	VT_BOOLEAN	OCX	ColTitle	ColTitle	CCAlgWinCtrl
"ColWidth"	VT_BOOLEAN	OCX	ColWidth	ColWidth	CCAlgWinCtrl
"CursorControl"	VT_BOOLEAN		Cursor Control	Cursor Control	I/O field, text list
"CursorMode"	VT_BOOLEAN				
"Danger"	VT_R4	OCX	Danger	Danger	XGaugeCtrl
"DangerColor"	VT_UI4	OCX	DangerColor	DangerColor	XGaugeCtrl
"DataFormat"	VT_I4		Data Format	Data Format	I/O field
"DataLanguage"	VT_I4		Configured Languages	Configured Languages	Button, static text, check box, radio box
"Delta"	VT_R4		Delta	Delta	XGaugeCtrl
"DesignMode"					
"DigitNumber"					
"Direction"	VT_I4		Bar Alignment	Bar Alignment	3D bar, slider object, bar
"EditAtOnce"	VT_BOOLEAN		Immediate Input	Immediate Input	I/O field, text list
"EnableTime"					
"EnableWriting"					
"EndAngle"	VT_I4		End Angle	End Angle	Elliptical arc, ellipse segment, circular arc, circle segment
"Exponent"	VT_BOOLEAN		Exponent Display	Exponent Display	Bar

3.3 Functions of the graphic system (RT Professional)

"ExtendedOperation"	VT_BOOLEAN		Extended Operation	Extended Operation	Slider object
"FillBackColor"					
"FillColor"	VT_I4		Fill Pattern Color	Fill Pattern Color	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"Filling"	VT_BOOLEAN		Dynamic Filling	Dynamic Filling	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, graphic object, check box, radio box, round button
"FillingIndex"	VT_I4		Fill Level	Fill Level	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, graphic object, check box, radio box, round button, slider object
"FillStyle"	VT_UI4		Fill Pattern	Fill Pattern	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"FillStyle2"	VT_UI4		Bar Pattern	Bar Pattern	Bar
"FlashBackColor"	VT_BOOLEAN		Flashing Background Active	Flashing Background Active	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object

3.3 Functions of the graphic system (RT Professional)

"FlashBorderColor"	VT_BOOL		Flashing Line Active	Flashing Line Active	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"FlashBorderColor-Ex"					
"FlashFlashPicture"	VT_BOOL		Flashing Flash Picture Active	Flashing Flash Picture Active	Status display
"FlashForeColor"	VT_BOOL		Flashing Text Active	Flashing Text Active	Button, static text, I/O field, text list, check box, radio box
"FlashPicReferenced"	VT_BOOL		Flash Picture Referenced	Flash Picture Referenced	Status display
"FlashPicTransparentColor"	VT_I4		Basic Picture Transparent Color	Basic Picture Transparent Color	Status display
"FlashPicture"	VT_BSTR		Flash Picture	Flash Picture	Status display
"FlashPicUseTransparentColor"			Flash Picture Transparent Color On	Flash Picture Transparent Color On	Status display
"FlashRate"	VT_I4		Flash Frequency	Flash Frequency	Group display
"FlashRateBackColor"	VT_I4		Background Flash Frequency	Background Flash Frequency	Button, ellipse, ellipse segment, circle, circle segment, polygon, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"FlashRateBorderColor"	VT_I4		Line Flash Frequency	Background Flash Frequency	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, I/O field, bar, graphic object, text list, check box, radio box, round button, slider object
"FlashRateFlashPic"	VT_I4		Flash Picture Flash Frequency	Flash Picture Flash Frequency	Status display

3.3 Functions of the graphic system (RT Professional)

"FlashRateForeCol- or"	VT_I4		Text Flash Fre- quency	Text Flash Fre- quency	Button, static text, I/O field, text list, check box, radio box
"Font"	VT_USE RDE- FINED	OCX	Font	Font	DAClockCtrl, PBut- tonCtrl, SliderCtrl
"FontBold"	VT_BOO L		Bold	Bold	Button, static text, I/O field, bar, group dis- play, text list, check box, radio box
"FontBold"	VT_BOO L	OCX	FontBold	FontBold	PButtonCtrl
"FontItalic"	VT_BOO L		Italic	Italic	Button, static text, I/O field, group display, text list, check box, ra- dio box
"FontItalic"	VT_BOO L		FontItalic	FontItalic	PButtonCtrl
"FontName"	VT_BST R		Font	Font	Button, static text, I/O field, group display, text list, check box, ra- dio box
"FontName"	VT_BST R	OCX	FontName	FontName	PButtonCtrl
"FontPosition"	VT_USE RDE- FINED	OCX	FontPosition	FontPosition	SliderCtrl
"FontSize"	VT_I4		Font Size	Font Size	Button, static text, I/O field, group display, text list, check box, ra- dio box
"FontSize"	VT_BST R	OCX	FontSize	Font Size	PButtonCtrl
"FontStrikeThru"	VT_BOO L	OCX	FontStrikeThru	FontStrikeThru	PButtonCtrl
"FontUnderline"	VT_BOO L		Underline	Underline	Button, static text, I/O field, group display, text list, check box, ra- dio box
"FontUnderline"	VT_BOO L	OCX	FontUnderline	FontUnderline	PButtonCtrl
"ForceInput"					
"ForeColor"	VT_I4		Font Color	Font Color	Button, static text, I/O field, group display, text list, check box, ra- dio box
"ForeColor"	VT_UI4	OCX	ForeColor	ForeColor	DAClockCtrl, PBut- tonCtrl, SliderCtrl
"ForeFlashColorOff"	VT_I4		Flashing Text Color Off	Flashing Text Color Off	Button, static text, I/O field, group display, text list, check box, ra- dio box

3.3 Functions of the graphic system (RT Professional)

"ForeFlashColorOn"	VT_I4		Flashing Text Color On	Flashing Text Color On	Button, static text, I/O field, group display, text list, check box, radio box
"FrameColor"	VT_UI4				
"FrameColorDown"	VT_UI4	OCX	FrameColor-Down	FrameColor-Down	PButtonCtrl
"FrameColorUp"	VT_UI4	OCX	FrameColorUp	FrameColorUp	PButtonCtrl
"FramePicture"	VT_USE RDE-FINED	OCX	FramePicture	FramePicture	XGaugeCtrl
"FrameScale"	VT_R4	OCX	FrameScale	FrameScale	XGaugeCtrl
"FrameSize"		OCX	FrameSize	FrameSize	PButtonCtrl
"FrameWidth"	VT_I4	OCX	FrameWidth	FrameWidth	PButtonCtrl
"Freeze"	VT_BOOLEAN		Freeze	Freeze	WTVctrlCtrl
"Grid"	VT_BOOLEAN				
"GridColor"	VT_I4				
"GridHeight"	VT_I4				
"GridLineHorz"	VT_BOOLEAN	OCX	GridLineHorz	GridLineHorz	CCAlgWinCtrl
"GridLineVert"	VT_BOOLEAN	OCX	GridLineVert	GridLineVert	CCAlgWinCtrl
"GridWidth"	VT_I4				
"Height"	VT_I4		Height	Height	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"Height"	VT_I4		Window Height	Window Height	Print job/Script diagnostics, screen window
"HiddenInput"	VT_BOOLEAN		Hidden Input	Hidden Input	I/O field
"Hotkey"	VT_UI4		Hot Key	Hot Key	Button
"HourNeedleHeight"	VT_I4		HourNeedle-Height	HourNeedle-Height	DAClockCtrl
"HourNeedleWidth"	VT_I4		HourNeedle-Width	HourNeedle-Width	DAClockCtrl

3.3 Functions of the graphic system (RT Professional)

"Hysteresis"	VT_BOOL		Hysteresis	Hysteresis	Bar
"HysteresisRange"	VT_R8		Hysteresis Range	Hysteresis Range	Bar
"Index"	VT_I4		Index	Index	Polygon, polyline, status display, check box, radio box
"InnerBevelOffset"	VT_I4	OCX	InnerBevelOffset	InnerBevelOffset	SliderCtrl
"InnerBevelStyle"	VT_I4	OCX	InnerBevelStyle	InnerBevelStyle	SliderCtrl
"InnerBevelWidth"	VT_I4	OCX	InnerBevelWidth	InnerBevelWidth	SliderCtrl
"ItemBorderBackColor"	VT_I4		Dividing Background Color	Dividing Background Color	Text list
"ItemBorderColor"	VT_I4		Dividing Line Color	Dividing Line Color	Text list
"ItemBorderStyle"	VT_UI4		Dividing Line Style	Dividing Line Style	Text list

3.3.1.6 List of object properties (L-Z) (Graphics Designer) (RT Professional)

Overview

Note

For many API functions of the Graphics Designer, you must initialize the `IpszPropName` parameter. The English name of an object property must be specified for this.

Properties of type `VT_USERDEFINED` must not be processed with the Get/Set Property functions.

`VT_DISPATCH` and other references must not be used; only the standard types and at most one `VT_VARIANT` for an array property of simple types may be used.

OLE Automation name	Data type		Property name	Attribute name	Object association
"LanguageSwitch"	VT_BOOL		Language Switch	Language switching	Text list
"LastChange"	VT_BSTR				

3.3 Functions of the graphic system (RT Professional)

"Layer"	VT_I4		Layer	Layer	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, print job/script diagnostics, screen window, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"Layer00Checked"	VT_BOOL	BPC	Monitoring 0	Monitoring 0	3D bar
"Layer01Checked"	VT_BOOL	BPC	Monitoring 1	Monitoring 1	3D bar
"Layer02Checked"	VT_BOOL	BPC	Monitoring 2	Monitoring 2	3D bar
"Layer03Checked"	VT_BOOL	BPC	Monitoring 3	Monitoring 3	3D bar
"Layer04Checked"	VT_BOOL	BPC	Monitoring 4	Monitoring 4	3D bar
"Layer05Checked"	VT_BOOL	BPC	Monitoring 5	Monitoring 5	3D bar
"Layer06Checked"	VT_BOOL	BPC	Monitoring 6	Monitoring 6	3D bar
"Layer07Checked"	VT_BOOL	BPC	Monitoring 7	Monitoring 7	3D bar
"Layer08Checked"	VT_BOOL	BPC	Monitoring 8	Monitoring 8	3D bar
"Layer09Checked"	VT_BOOL	BPC	Monitoring 9	Monitoring 9	3D bar
"Layer10Checked"	VT_BOOL	BPC	Monitoring 10	Monitoring 10	3D bar
"Layer00Color"	VT_I4	BPC	Bar Color 0	Bar Color 0	3D bar
"Layer01Color"	VT_I4	BPC	Bar Color 1	Bar Color 1	3D bar
"Layer02Color"	VT_I4	BPC	Bar Color 2	Bar Color 2	3D bar
"Layer03Color"	VT_I4	BPC	Bar Color 3	Bar Color 3	3D bar
"Layer04Color"	VT_I4	BPC	Bar Color 4	Bar Color 4	3D bar
"Layer05Color"	VT_I4	BPC	Bar Color 5	Bar Color 5	3D bar
"Layer06Color"	VT_I4	BPC	Bar Color 6	Bar Color 6	3D bar
"Layer07Color"	VT_I4	BPC	Bar Color 7	Bar Color 7	3D bar
"Layer08Color"	VT_I4	BPC	Bar Color 8	Bar Color 8	3D bar
"Layer09Color"	VT_I4	BPC	Bar Color 9	Bar Color 9	3D bar
"Layer10Color"	VT_I4	BPC	Bar Color 10	Bar Color 10	3D bar
"Layer00Value"	VT_R8	BPC	Limit 0	Limit 0	3D bar

3.3 Functions of the graphic system (RT Professional)

"Layer01Value"	VT_R8	BPC	Limit 1	Limit 1	3D bar
"Layer02Value"	VT_R8	BPC	Limit 2	Limit 2	3D bar
"Layer03Value"	VT_R8	BPC	Limit 3	Limit 3	3D bar
"Layer04Value"	VT_R8	BPC	Limit 4	Limit 4	3D bar
"Layer05Value"	VT_R8	BPC	Limit 5	Limit 5	3D bar
"Layer06Value"	VT_R8	BPC	Limit 6	Limit 6	3D bar
"Layer07Value"	VT_R8	BPC	Limit 7	Limit 7	3D bar
"Layer08Value"	VT_R8	BPC	Limit 8	Limit 8	3D bar
"Layer09Value"	VT_R8	BPC	Limit 9	Limit 9	3D bar
"Layer10Value"	VT_R8	BPC	Limit 10	Limit 10	3D bar
"Left"	VT_I4		X position	X position	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, print job/script diagnostics, screen window, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"LeftComma"	VT_I4		Digits to the Left of the Decimal Point	Digits to the Left of the Decimal Point	Bar
"LightEffect"	VT_BOOL		Light Effect	Light Effect	3D bar
"LimitHigh4"	VT_R8		Limit RH4	Limit RH4	Bar
"LimitHigh5"	VT_R8		Limit RH5	Limit RH5	Bar
"LimitLow4"	VT_R8		Limit RL4	Limit RL4	Bar
"LimitLow5"	VT_R8		Limit RL5	Limit RL5	Bar
"LimitMax"	VT_R8				
"LimitMin"	VT_R8				
"LineFont"	VT_BOOL	OCX	LineFont	LineFont	CCAlgWinCtrl
"LineHeight"	VT_BOOL	OCX	LineHeight	LineHeight	CCAlgWinCtrl
"LineTitle"	VT_BOOL	OCX	LineTitle	LineTitle	CCAlgWinCtrl
"ListType"	VT_I4		List Type	List Type	Text list
"Locale"					
"LockBackColor"	VT_I4		Lock Background Color	Lock Background Color	Group display
"LockStatus"	VT_BOOL		Lock Display	Lock Display	Group display
"LockText"	VT_BOOL		Lock Display Text	Lock Display Text	Group display
"LockTextColor"	VT_I4		Lock Font Color	Lock Font Color	Group display

3.3 Functions of the graphic system (RT Professional)

"LongStrokes-Bold"	VT_BOOL		Long Axis Section	Long Axis Section	Bar
"LongStrokesOnly"	VT_BOOL		Only Long Axis Section	Only Long Axis Section	Bar
"LongStrokes-Size"	VT_UI4		Length of Axis Section	Length of Axis Section	Bar
"LongStrokes-TextEach"	VT_UI4		Label Each	Label Each	Bar
"Marker"	VT_BOOL		Limit Marker	Limit Marker	Bar
"Max"	VT_R8		Maximum Value	Maximum Value	3D bar, slider object, bar
"MaximizeButton"	VT_BOOL		Can be Maximized	Can be Maximized	Print job/Script diagnostics, screen window
"MCGUBackColorOff"	VT_I4		Outgoing Unacknowledged- Background Color Off	Outgoing Unacknowledged- Background Color Off	Group display
"MCGUBackColorOn"	VT_I4		Outgoing Unacknowledged- Background Color On	Outgoing Unacknowledged- Background Color On	Group display
"MCGUBackFlash"	VT_BOOL		Outgoing Unacknowledged - Background Flashing	Outgoing Unacknowledged - Background Flashing	Group display
"MCGUTextColorOff"	VT_I4		Outgoing Unacknowledged- Text Color Off	Incoming - Background Color Off	Group display
"MCGUTextColorOn"	VT_I4		Outgoing Unacknowledged- Text Color On	Incoming - Background Color On	Group display
"MCGUTextFlash"	VT_BOOL		Outgoing Unacknowledged - Text Flashing	Outgoing Unacknowledged - Background Flashing	Group display
"MCKOBackColorOff"	VT_I4		Incoming - Background Color Off	Incoming - Background Color Off	Group display
"MCKOBackColorOn"	VT_I4		Incoming - Background Color On	Incoming - Background Color On	Group display
"MCKOBackFlash"	VT_BOOL		Incoming - Background Flashing	Incoming - Background Flashing	Group display
"MCKOTextColorOff"	VT_I4		Incoming - Text Color Off	Incoming - Text Color Off	Group display

3.3 Functions of the graphic system (RT Professional)

"MCKOTextColorOn"	VT_I4		Incoming - Text Color On	Incoming - Text Color On	Group display
"MCKOTextFlash"	VT_BOOL		Incoming - Text Flashing	Incoming - Text Flashing	Group display
"MCKQBackColorOff"	VT_I4		Incoming Acknowledge - Background Color Off	Incoming Acknowledge - Background Color Off	Group display
"MCKQBackColorOn"	VT_I4		Incoming Acknowledge - Background Color On	Incoming Acknowledge - Background Color On	Group display
"MCKQBackFlash"	VT_BOOL		Incoming Acknowledge - Background Flashing	Incoming Acknowledge - Background Flashing	Group display
"MCKQTextColorOff"	VT_I4		Incoming Acknowledge - Text Color Off	Incoming Acknowledge - Text Color Off	Group display
"MCKQTextColorOn"	VT_I4		Incoming Acknowledge - Text Color On	Incoming Acknowledge - Text Color On	Group display
"MCKQTextFlash"	VT_BOOL		Incoming Acknowledge - Text Flashing	Incoming Acknowledge - Text Flashing	Group display
"MCText"	VT_BOOL		Display Text	Display Text	Group display
"MessageClass"	VT_I4		Message Type	Message Type	Group display
"Min"	VT_R8		Minimum Value	Minimum Value	Group display
"MinuteNeedleHeight"	VT_I4	OCX	MinuteNeedleHeight	MinuteNeedleHeight	Group display
"MinuteNeedleWidth"	VT_I4	OCX	MinuteNeedleWidth	MinuteNeedleWidth	Group display
"MsgFilter"	VT_BSTR	OCX		MsgFilter	
"MsgFilter2"	VT_BSTR	OCX		MsgFilter2	
"Moveable"	VT_BOOL		Moveable	Moveable	Print job/Script diagnostics, screen window
"NeedleColor"	VT_UI4	OCX	NeedleColor	NeedleColor	XGaugeCtrl
"NormalColor"	VT_UI4	OCX	NormalColor	NormalColor	XGaugeCtrl
"NumberLines"	VT_I4		Number of visible Lines	Number of visible Lines	Text list

3.3 Functions of the graphic system (RT Professional)

"ObjectName"	VT_BSTR		Object Name	Object name	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, print job/script diagnostics, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"OffsetLeft"	VT_I4		Picture Offset X	Picture Offset X	Screen window
"OffsetTop"	VT_I4		Picture Offset Y	Picture Offset Y	Screen window
"OnTop"	VT_BOOL		Foreground	Foreground	Print job/Script diagnostics, screen window
"Operation"	VT_BOOL		Operator-Control Enable	Operator-Control Enable	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"OperationMessage"	VT_BOOL		Operator Input Message	Operator Input Message	I/O field, text list, check box, radio box, slider object
"OperationReport"	VT_BOOL		Operator Actions Report	Operator Actions Report	I/O field, text list, slider object
"OperationSteps"			Operation Steps	Operation Steps	Slider object
"Orientation"	VT_BOOL		Text Orientation	Text Orientation	Static text, I/O field, text list, check box, radio box
"OuterBevelOffset"	VT_I4	OCX	OuterBevelOffset	OuterBevelOffset	SliderCtrl
"OuterBevelStyle"	VT_I4	OCX	OuterBevelStyle	OuterBevelStyle	SliderCtrl
"OuterBevelWidth"	VT_I4	OCX	OuterBevelWidth	OuterBevelWidth	SliderCtrl
"Outline"	VT_BOOL	OCX	Outline	Outline	PButtonCtrl
"OutputFormat"	VT_BSTR		Output Format	Output Format	I/O field

3.3 Functions of the graphic system (RT Professional)

"OutputValue"	VT_R8		Output Value	Output Value	I/O field, text list
"PasswordLevel"	VT_UI4		User Level	Password	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object
"PicDeactReferenced"	VT_BOOL		Picture Deact. Referenced	Picture Deact. Referenced	Round button
"PicDeactTransparent"	VT_I4		Picture Deact. Transparent Color	Picture Deact. Transparent Color	Round button
"PicDeactUseTransparentColor"	VT_BOOL		Picture Deact. Transparent Color On	Picture Deact. Transparent Color On	Round button
"PicDownReferenced"	VT_BOOL		Picture On Referenced	Picture On Referenced	Round button
"PicDownTransparent"	VT_I4		Picture On Transparent Color	Picture On Transparent Color	Round button
"PicDownUseTransparentColor"	VT_BOOL		Picture On Transparent Color On	Picture On Transparent Color On	Round button
"PicReferenced"	VT_BOOL		Picture Referenced	Picture Referenced	Graphic object
"PicTransparentColor"	VT_I4		Picture Transparent Color	Picture Transparent Color	Graphic object
"Picture"	VT_USER DEFINED	OCX	Picture	Picture	DAClockCtrl
"PictureBack"	VT_USER DEFINED	OCX	PictureBack	PictureBack	SliderCtrl
"PictureDeactivated"	VT_BSTR		Picture Status Deactivated	Picture Status Deactivated	Round button
"PictureDown"	VT_BSTR		Picture Status On	Picture Status On	Button, round button
"PictureName"	VT_BSTR		Picture Name	Screen name	Screen window, graphic object
"PictureSelected"	VT_USER DEFINED	OCX	PictureSelected	PictureSelected	PButtonCtrl
"PictureUnselected"	VT_USER DEFINED	OCX	Picture Unselected	Picture Unselected	PButtonCtrl
"PictureObject-Path"					
"PictureThumb"	VT_USER DEFINED		PictureThumb	PictureThumb	SliderCtrl

3.3 Functions of the graphic system (RT Professional)

"PictureUp"	VT_BSTR		Picture Status Off	Picture Status Off	Button, round button
"PicUpReferenced"	VT_BOOL		Picture Off Referenced	Picture Off Referenced	Round button
"PicUpTransparent"	VT_I4		Picture Off Transparent Color	Picture Off Transparent Color	Round button
"PicUpUseTransparentColor"	VT_BOOL		Picture Off Transparent Color On	Picture Off Transparent Color On	Round button
"PicUseTransparentColor"	VT_BOOL		Picture Transparent Color On	Transparent Color On	Graphic object
"PointCount"	VT_I4		Number of Corners	Number of Corners	Polygon, polyline
"Position"	VT_I4	OCX	Positions	Positions	SliderCtrl
"PredefinedAngles"	VT_I4	BPC	Angle Settings	Angle Settings	3D bar
"Pressed"	VT_BOOL		Pressed	Pressed	Round button
"Process"	VT_R8		Process Driver Connection	Process Driver Connection	3D bar, slider object, bar
"Process"	VT_R8		Selected Boxes	Selected Boxes	Check box, radio box
"ProfileName"					
"ProjectName"					
"ProjectPath"	VT_BSTR	OCX	ProjectPath	ProjectPath	CCAlgWinCtrl
"Radius"	VT_I4		Radius	Radius	Circle, circular arc, circle segment, round button
"RadiusHeight"	VT_I4		Y radius	Y radius	Ellipse, Elliptical arc, ellipse segment
"RadiusWidth"	VT_I4		X radius	X radius	Ellipse, Elliptical arc, ellipse segment
"RangeMax"	VT_I4	OCX	RangeMax	RangeMax	SliderCtrl
"RangeMin"	VT_I4	OCX	RangeMin	RangeMin	SliderCtrl
"ReferenceMoveLeft"					
"ReferenceMoveTop"					
"ReferenceRotationLeft"	VT_I4		Rotation Reference X	Rotation Reference X	Line, polygon, polyline
"ReferenceRotationTop"	VT_I4		Rotation Reference Y	Rotation Reference Y	Line, polygon, polyline
"RefreshTimerPeriod"	VT_I4	OCX	RefreshTimerPeriod	RefreshTimerPeriod	WTVctrlCtrl
"Relevant"	VT_BOOL		Group Relevant	Group Relevant	Group display
"RightComma"	VT_I4		Digits to the Right of the Decimal Point	Digits to the Right of the Decimal Point	Bar

3.3 Functions of the graphic system (RT Professional)

"RotationAngle"	VT_I4		Rotation Angle	Rotation Angle	Line, polygon, polyline
"RoundCorner-Height"	VT_I4		Corner Radius Y	Corner Radius Y	Rounded rectangle
"RoundCorner-Width"	VT_I4		Corner Radius X	Corner Radius X	Rounded rectangle
"SameSize"	VT_BOOL		Same Size	Same Size	Group display
"ScaleColor"	VT_I4		Scale Color	Scale Color	Bar
"ScaleTicks"	VT_I4		Scale Marks	Scale Marks	Bar
"Scaling"	VT_BOOL		Scale	Scale	Bar
"ScalingType"	VT_I4		Bar Scaling	Bar Scaling	Bar
"ScrollBars"	VT_BOOL		Scroll Bar	Scroll Bar	Screen window
"SecondNeedle-Height"	VT_I4	OCX	SecondNeedleHeight	SecondNeedleHeight	DAClockCtrl
"SecondNeedle-Width"	VT_I4	OCX	SecondNeedleWidth	SecondNeedleWidth	DAClockCtrl
"SelBGColor"	VT_I4		Selection Background Color	Selection Background Color	Text list
"SelectionMode"	VT_I2	OCX	SelectionMode	SelectionMode	CCAlgWinCtrl
"SelTextColor"	VT_I4		Selection Font Color	Selection Font Color	Text list
"ServerName"	VT_BSTR		Server Name	Server Name	DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"Sign"					
"Significant-Mask"	VT_UI4		Bit Pattern Group Display	Bit Pattern Group Display	Group display
"Sizeable"	VT_BOOL		Sizeable	Sizeable	Print job/Script diagnostics, screen window
"ShowBar"	VT_BOOL	OCX	ShowBar	ShowBar	SliderCtrl
"ShowDanger"	VT_BOOL	OCX	ShowDanger	ShowDanger	XGaugeCtrl
"ShowDecimal-Point"	VT_BOOL	OCX	ShowDecimal-Point	ShowDecimal-Point	XGaugeCtrl
"ShowNormal"	VT_BOOL	OCX	ShowNormal	ShowNormal	XGaugeCtrl
"ShowPeak"	VT_BOOL	OCX	ShowPeak	ShowPeak	XGaugeCtrl
"ShowPosition"	VT_BOOL	OCX	ShowPosition	ShowPosition	SliderCtrl
"ShowThumb"	VT_BOOL	OCX	ShowThumb	ShowThumb	SliderCtrl
"SmallWarning"	VT_I4			SmallWarning	Slider object
"StartAngle"	VT_I4		Start Angle	Start Angle	Elliptical arc, ellipse segment, circular arc, circle segment
"TabOrderAlpha"					
"TabOrderSwitch"					
"Template"	VT_BSTR		Template	Template	Print job/Script diagnostics
"TemplateName"	VT_BSTR	OCX	Template-Name	Template-Name	CCAlgWinCtrl

3.3 Functions of the graphic system (RT Professional)

"Text"	VT_BSTR		Text	Text	Button, static text, check box, radio box
"ThumbBackColor"	VT_UI4	OCX	ThumbBack-Color	ThumbBack-Color	SliderCtrl
"TicColor"	VT_UI4	OCX	TicColor	TicColor	XGaugeCtrl
"TicFont"	VT_USER_DEFINED	OCX	TicFont	TicFont	XGaugeCtrl
"Ticks"	VT_BOOL	OCX	Ticks	Ticks	DAClockCtrl
"TicksColor"	VT_UI4	OCX	TicksColor	TicksColor	DAClockCtrl
"TickStyle"	VT_I4	OCX	TickStyle	TickStyle	SliderCtrl
"TicOffset"	VT_R8	OCX	TicOffset	TicOffset	XGaugeCtrl
"TicTextColor"	VT_UI4	OCX	TicTextColor	TicTextColor	XGaugeCtrl
"TicTextOffset"	VT_R8	OCX	TicTextOffset	TicTextOffset	XGaugeCtrl
"TicWidth"	VT_R8	OCX	TicWidth	TicWidth	XGaugeCtrl
"TitleCut"	VT_BOOL	OCX	TitleCut	TitleCut	CCAlgWinCtrl
"Toggle"	VT_BOOL		Latch Down	Latch Down	Round button
"ToleranceHigh"	VT_R8		Limit TH	Limit TH	Bar
"ToleranceLow"	VT_R8		Limit TL	Limit TL	Bar
"ToolTipText"	VT_BSTR		Tooltip Text	Tooltip Text	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"Top"	VT_I4		Y position	Y position	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, print job/script diagnostics, screen window, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"Trend"	VT_BOOL		Trend	Trend	Bar
"TrendColor"	VT_I4		Trend Color	Trend Color	Bar

3.3 Functions of the graphic system (RT Professional)

"TypeAlarmHigh"	VT_BOOL		Type AH	Type AH	Bar
"TypeAlarmLow"	VT_BOOL		Type AL	Type AL	Bar
"TypeLimitHigh4"	VT_BOOL		Type RH4	Type RH4	Bar
"TypeLimitHigh5"	VT_BOOL		Type RH5	Type RH5	Bar
"TypeLimitLow4"	VT_BOOL		Type RL4	Type RL4	Bar
"TypeLimitLow5"	VT_BOOL		Type RL5	Type RL5	Bar
"TypeTolerance-High"	VT_BOOL		Type TH	Type TH	Bar
"TypeTolerance-Low"	VT_BOOL		Type TL	Type TL	Bar
"TypeWarning-High"	VT_BOOL		Type WH	Type WH	Bar
"TypeWarningLow"	VT_UI4		Type WL	Type WL	Bar
"UnitColor"	VT_USER DEFINED	OCX	UnitColor	UnitColor	XGaugeCtrl
"UnitFont"	VT_BSTR	OCX	UnitFont	UnitFont	XGaugeCtrl
"UnitText"	VT_BSTR	OCX	UnitText	UnitText	XGaugeCtrl
"UnitOffset"	VT_R8	OCX	UnitOffset	UnitOffset	XGaugeCtrl
"UnselBGColor"	VT_I4		List Back- ground Color	List Back- ground Color	Text list
"UnselTextColor"	VT_I4		List Font Color	List Font Color	Text list
"UpdateCycle"	VT_I4		Update Cycle	Update Cycle	Screen window
"UseRefreshTimer"	VT_BOOL	OCX	UseRefresh- Timer	UseRefresh- Timer	WTVctrlCtrl
"UserValue1"	VT_UI4		User Value 1	User Value 1	Group display
"UserValue2"	VT_UI4		User Value 2	User Value 2	Group display
"UserValue3"	VT_UI4		User Value 3	User Value 3	Group display
"UserValue4"	VT_UI4		User Value 4	User Value 4	Group display
"Value"	VT_R4	OCX	Value	Value	XGaugeCtrl
"ValueMax"	VT_R4	OCX	ValueMax	ValueMax	XGaugeCtrl
"ValueMin"	VT_R4	OCX	ValueMin	ValueMin	XGaugeCtrl
"VideoEnabled"	VT_BOOL	OCX	VideoEnabled	VideoEnabled	WTVctrlCtrl
"VideoSource"	VT_I2	OCX	VideoSource	VideoSource	WTVctrlCtrl

3.3 Functions of the graphic system (RT Professional)

"Visible"	VT_BOOL		Display	Display	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, print job/script diagnostics, screen window, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"Warning"	VT_R4	OCX	Warning	Warning	XGaugeCtrl
"WarningColor"	VT_UI4	OCX	WarningColor	WarningColor	XGaugeCtrl
"WarningHigh"	VT_R8		Limit WH	Limit WH	Bar
"WarningLow"	VT_R8		Limit WL	Limit WL	Bar
"Width"	VT_I4		Width	Width	Button, ellipse, elliptical arc, ellipse segment, circle, circular arc, circle segment, line, polygon, polyline, rectangle, rounded rectangle, static text, 3-D bar, print job/script diagnostics, screen window, I/O field, bar, graphic object, group display, text list, status display, check box, radio box, round button, slider object, DAClockCtrl, XGaugeCtrl, CCAlgWinCtrl, PButtonCtrl, SliderCtrl, WTVctrlCtrl
"WindowBorder"	VT_BOOL		Border	Border	Print job/Script diagnostics, screen window
"WindowsStyle"	VT_BOOL		Windows Style	Windows Style	Button, Slider object
"WindowType"	VT_I2	OCX	WindowType	WindowType	CCAlgWinCtrl
"WinTVInformation"	VT_BSTR	OCX		WinTVInformation	WTVctrlCtrl
"WithAxes"	VT_BOOL	OCX	WithAxes	WithAxes	SliderCtrl
"WithLabels"	VT_BOOL	OCX	WithLabels	WithLabels	SliderCtrl
"ZeroLeading"					
"ZeroPoint"	VT_UI4		Zero Point	Zero Point	Bar
"ZeroPointValue"	VT_R8		Zero Point Value	Zero Point Value	3-D bar, bar
Zoom	VT_I4		Scaling Factor	Scaling Factor	Screen window

3.3.1.7 API calls to OCXs (RT Professional)

Overview

API was developed for uses in separate applications. However, problems may arise – including possible deadlocks – when API calls are made to OCXs that are integrated into graphics screens.

If use of an OCX in a graphics screen is absolutely necessary, always check to see if the task could be also be handled with IndustrialX. This option is designed for purposes like this.

If it is determined that API must be used for the task, note the following:

The Runtime API functions of alarm logging and tag logging are especially affected by this.

One reason is that API requests data from WinCC programs during runtime via Window Message and awaits the reply. If the OCX runs in the context of just such a WinCC program, it cannot process the request because it hangs in the API call.

Solution:

When this happens, it is often sufficient to place the API call in a different thread.

However, this solution should only be used when a problem actually exists. For example, some other calls experience problems when they are not run in the main thread.

3.3.2 Structures (RT Professional)

3.3.2.1 LINKINFO (RT Professional)

Declaration

```
typedef struct {
    LINKTYPE    LinkType;
    DWORD       dwCycle;
    TCHAR       szLinkName[256];
}
LINKINFO;
```

Members

LinkType

Values for enum LinkType defined in the "trigger.h" file should be used for the LINKTYPE type.

enum LinkType	Currently defined value	Link
BUBRT_LT_NOLINK	0 (defined start value)	No link
BUBRT_LT_VARIABLE_DIRECT	1 (sequence)	Direct tag
BUBRT_LT_VARIABLE_INDIRECT	2 (sequence)	Indirect tag

enum LinkType	Currently defined value	Link
BUBRT_LT_ACTION	3 (sequence)	C action
BUBRT_LT_ACTION_WIZARD	4 (sequence)	Dynamic dialog
BUBRT_LT_DIRECT_CONNECTION	5 (sequence)	Direct connection
BUBRT_LT_ACTION_WIZARD_INPROC	6 (sequence)	Dynamic dialog

When PDLRTSetLink is used, only BUBRT_LT_VARIABLE_DIRECT and BUBRT_LT_VARIABLE_INDIRECT may be used. All types of links can be used for PDLRTGetLink.

Note

For subsequent extensions of the enum LinkType definition in trigger.h, new constants must always be added at the end, otherwise the above listed values will be changed. Therefore, you should always use the definition for enumerations and not the value.

dwCycle

Cycle time for updating. dwCycle identifies the order within the list of update cycles.

"On change"	Index: 0
"250ms"	Index: 1
:	:
"User cycle 5"	Index: 15
"Window cycle"	Index: 235
"Screen cycle"	Index: 255

szLinkName

Name of the tag connection

Required files

pdlrtapi.h

trigger.h

API functions

PDLRTGetLink (Page 1933)	Query link between object property and tag
PDLRTSetLink (Page 1935)	Define link between object property and tag

See also

PDLRTGetLink (Page 1933)

PDLRTSetLink (Page 1935)

3.3.2.2 MULTILINK (RT Professional)

Declaration

```
typedef struct {
    DWORD          wArraySize;
    LPMULTILINKINFO pLinkArray;
}
MULTILINK;
```

Members

- wArraySize**
Number of links (number of structures of the MULTILINKINFO type).
- pLinkArray**
Pointer to the first wArraySize structure of the MULTILINKINFO (Page 1899) type with information about the link between an object property and a tag.

Required files

pdlrtapi.h

API functions

PDLRTSetMultiLink (Page 1937)	Define link between object property and tag (multiple tags)

See also

- PDLRTSetMultiLink (Page 1937)
- MULTILINKINFO (Page 1899)

3.3.2.3 MULTILINKINFO (RT Professional)

Declaration

```
typedef struct {
    char*      pszObjectName,
    char*      pszPropertyName,
    LINKTYPE   LinkType;
    DWORD      dwCycle;
    char*      pszLinkName;
}
MULTILINKINFO;
```

Members

lpzObjectName

Pointer to the name of the object.

lpzPropertyName

Pointer to the name of the object property to be linked.

LinkType

Identifies the type of connection.

LT_VARIABLE_DIRECT	Connect object property and tag directly
LT_VARIABLE_INDIRECT	Connect object property and tag indirectly

dwCycle

Cycle time for updating. dwCycle identifies the order within the list of update cycles.

"On change"	Index: 0
"250ms"	Index: 1
:	:
"User cycle 5"	Index: 15

pszLinkName

Pointer to name of the connection

Comments

The MULTILINKINFO structure is part of the MULTILINK (Page 1896) structure.

Required files

pdIrtapi.h

See also

MULTILINK (Page 1896)

3.3.2.4 FOCUSINFO (RT Professional)

Description

The screen object that has the runtime cursor is stored in the structure.

Declaration

```
typedef struct {
    WCHAR    szPicture[256];
    WCHAR    szObject[256];
}
FOCUSINFO;
```

Members

- szPicture**
Screen name
- szObject**
Object name

API functions

PDLRTGetFocus (Page 1917)	Query input focus

See also

PDLRTGetFocus (Page 1917)

3.3.3 General functions (RT Professional)

3.3.3.1 PDLRTClosPicture (RT Professional)

Use

Closes a screen in runtime mode.

Declaration

```

BOOL PDLRTClosePicture(
    ADRMODE          adrMode,
    LPCSTR           lpszPictureName,
    LPCSTR           lpszWName,
    PDLRT_CALLBACK   pfn,
    LPVOID           pvUser,
    PCMN_ERROR       pError );

```

Parameters

adrMode

The parameter is reserved for future development and must be preset to 0.

lpszPictureName

Pointer to the configured name of the screen, without path or extension.

The entry is case-sensitive.

lpszWName

Name of the window object to be closed.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If `pfn = NULL` is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Screen closed.

FALSE

Error.

Required files

pdLrtapi.h

pdLrt_s.lib

pdLrtapi.dll

Related functions

PDLRT_CALLBACK (Page 1913)	Callback function of the runtime functions

See also

PDLRT_CALLBACK (Page 1913)

Overview of the functions (Page 1867)

3.3.3.2 PDLRTDisableClosePicture (RT Professional)**Use**

Blocks a screen from being closed in runtime mode. The function can be called multiple times. The screen can only be closed again when the same number of calls to the PDLRTEnableClosePicture function have been made.

Declaration

```

BOOL PDLRTDisableClosePicture(
    ADRMODE      adrMode,
    LPCSTR       lpszPictureName,
    PDLRT_CALLBACK pfn,
    LPVOID       pvUser,
    PCMN_ERROR    pError );

```

Parameters**adrMode**

The parameter is reserved for future development and must be preset to 0.

lpszPictureName

Pointer to the configured name of the screen, without path or extension.

The entry is case-sensitive.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If `pfn = NULL` is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. If an error occurs, the system writes error information into this structure.

Return value**TRUE**

Function successfully completed.

FALSE

Error.

Required files

`pdlrtapi.h`

`pdlrt_s.lib`

`pdlrtapi.dll`

Related functions

PDLRT_CALLBACK (Page 1913)	Callback function of the runtime functions

See also

PDLRT_CALLBACK (Page 1913)

Overview of the functions (Page 1867)

3.3.3.3 PDLRTEnableClosePicture (RT Professional)

Use

Enables a screen to be closed in runtime mode.

The screen can only be closed, however, when the PDLRTEnableClosePicture function has been called the same number of times as the PDLRTDisableClosePicture function.

Any PDLRTCclosePicture made prior to this is noted and then performed later after the last PDLRTEnableClosePicture.

Declaration

```
BOOL PDLRTEnableClosePicture(  
    ADRMODE          adrMode,  
    LPCSTR            lpszPictureName,  
    PDLRT_CALLBACK    pfn,  
    LPVOID            pvUser,  
    PCMN_ERROR        pError );
```

Parameters

adrMode

The parameter is reserved for future development and must be preset to 0.

lpszPictureName

Pointer to the configured name of the screen, without path or extension.

The entry is case-sensitive.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Function successfully completed.

FALSE

Error.

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRT_CALLBACK (Page 1913)	Callback function of the runtime functions

See also

PDLRT_CALLBACK (Page 1913)

Overview of the functions (Page 1867)

3.3.3.4 PDLRTGotoPicture (RT Professional)**Use**

Loads the specified screen.

Declaration

```

BOOL PDLRTGotoPicture (
    PDLRTGotoPict  nextPict,
    PCMN_ERROR     pError );

```

Parameters

nextPict

The function of this method is controlled with this parameter. The following values are possible:

PDLRTPictureHome	Loads the start screen.
PDLRTPictureNext	Loads the next screen in screen memory.
PDLRTPicturePrev	Loads the previous screen in screen memory.
PDLRTPictureStored	Loads the stored screen in screen memory.
PDLRTStorePicture	Stores the current screen in screen memory.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Screen loaded.

FALSE

Error.

Error messages

PDLRT_NO_PIC	Screen not selected

Required files

pdLrtapi.h

pdLrt_s.lib

pdLrtapi.dll

Related functions

PDLRTPictureNavigation (Page 1911)	Enable and disable screen navigation

See also

PDLRTPictureNavigation (Page 1911)

Overview of the functions (Page 1867)

3.3.3.5 PDLRTInquireFreeArea (RT Professional)

Use

Determines whether a given rectangle can be placed on the screen without covering a "protected area". "Protected areas", for example, are other screens that are in the foreground.

Declaration

```

BOOL PDLRTInquireFreeArea (
    LPRECT          lpScreenRect,
    DWORD           dwModus,
    PDLRT_CALLBACK  pfn,
    LPVOID          pvUser,
    PCMN_ERROR      pError );

```

Parameters

lpScreenRect

Pointer to a Windows-specific RECT type structure.

dwModus

Inquire mode: Several inquiry options are provided that can be ORed as desired:

PDLRT_IQ_ONLY	(value: 0x01)	Upon demand only with the result "matches / does not match"
PDLRT_IQ_MODY_POSX	(value: 0x02)	If the given rectangle cannot be placed, the X position of the rectangle can be modified so that it can be placed.
PDLRT_IQ_MODY_POSY	(value: 0x04)	If the given rectangle cannot be placed, the Y position of the rectangle can be modified so that it can be placed.
PDLRT_IQ_MODY_HEIGHT	(value: 0x08)	If the given rectangle cannot be placed, the height of the rectangle can be modified so that it can be placed.
PDLRT_IQ_MODY_WIDTH	(value: 0x10)	If the given rectangle cannot be placed, the width of the rectangle can be modified so that it can be placed.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Window area is free.

FALSE

Error.

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRT_CALLBACK (Page 1913)	Callback function of the runtime functions

See also

PDLRT_CALLBACK (Page 1913)

Overview of the functions (Page 1867)

3.3.3.6 PDLRTOpenPicture (RT Professional)

Use

The function performs a start screen change in the runtime.

Declaration

```

BOOL PDLRTOpenPicture (
    ADRMODE          adrMode,
    LPCSTR           lpszPictureName,
    LPCSTR           lpszWName,
    LPCSTR           lpszPictureFileName,
    DWORD            dwWinStyle,
    LONG             lxPos,
    LONG             lyPos,
    LONG             lWidth,
    LONG             lHeight,
    PDLRT_CALLBACK   pfn,
    LPVOID           pvUser,
    PCMN_ERROR       pError );

```

Parameters

adrMode

Pointer to the configured name of the screen, without path or extension.

The entry is case-sensitive.

lpszWName

This parameter must be currently supplied with the screen name.

lpszPictureFileName

This parameter must be currently supplied with the screen name.

dwWinStyle

The parameter is reserved for future development and must be preset to 0.

lxPos

The parameter is reserved for future development and must be preset to 0.

lyPos

The parameter is reserved for future development and must be preset to 0.

lWidth

The parameter is reserved for future development and must be preset to 0.

lHeight

The parameter is reserved for future development and must be preset to 0.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

3.3 Functions of the graphic system (RT Professional)

If `pfn = NULL` is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Start screen change performed.

FALSE

Error.

Error messages

PDLRT_PICTURE_ALREADY_OPEN	Screen is already open
PDLRT_NO_PIC	Screen not selected
PDLRT_BAD_OLE_CONVERSION	Error in conversion through OLE automation

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRT_CALLBACK (Page 1913)	Callback function of the runtime functions

See also

PDLRT_CALLBACK (Page 1913)

Overview of the functions (Page 1867)

3.3.3.7 PDLRTPictureNavigation (RT Professional)

Use

Switches the screen navigation on and off. When switched off, no more screens are added to the stack with a screen change; the old state is frozen.

Declaration

```
BOOL PDLRTPictureNavigation (  
    PDLRT_PNFLAGS    flags,  
    PCMN_ERROR        pError );
```

Parameters

flags

The function is controlled with this parameter. The following values are possible:

PDLRT_PNF_ENABLE	Switches the screen navigation on.
PDLRT_PNF_DISABLE	Switches the screen navigation off.
PDLRT_PNF_ENABLE_LOAD_LAST_PICTURE	Switches the screen navigation on and loads the last screen in the screen memory.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Function successfully completed.

FALSE

Error.

Cause: Screen navigation was already switched on or off.

Required files

pdlrtapi.h
pdlrt_s.lib
pdlrtapi.dll

Related functions

PDLRTGotoPicture (Page 1903)	Load specified screen

See also

PDLRTGotoPicture (Page 1903)

Overview of the functions (Page 1867)

3.3.3.8 PDLRTShowApp (RT Professional)**Use**

This function brings a runtime window that may be in the background to the foreground of the desktop.

Declaration

```

BOOL PDLRTShowApp (
    PDLRT_CALLBACK    pfn,
    LPVOID             pvUser,
    PCMN_ERROR         pError );

```

Parameters**pfn**

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If `pfn = NULL` is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Runtime window in foreground.

FALSE

Error.

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRT_CALLBACK (Page 1913)	Callback function

See also

PDLRT_CALLBACK (Page 1913)

Overview of the functions (Page 1867)

3.3.3.9 PDLRT_CALLBACK (RT Professional)**Description**

If your application is to be informed asynchronously about the execution of API functions in runtime, you must provide callback functions of the PDLRT_CALLBACK type.

This function is used by all of the PDLRT functions as a callback function.

Declaration

```
void ( * PDLRT_CALLBACK ) (
    LPVOID      pvUser,
    PCMN_ERROR   pError );
```

Parameters**pvUser**

Pointer to application-specific data. This pointer is made available as a parameter of the API function and returned again here unchanged.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

This callback has no return value.

Comments

If the API function is called without callback, i.e. with pfn = NULL, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores. Without synchronization, there is also the risk that the asynchronous function will continue to write data via that data pointer that is no longer valid, when the calling process has already finished or left the range of validity, which will lead to exception handling.

It makes sense to use a separate callback for each API function.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

In rare cases, the callback may already be delivered before the function call has returned.

Required files

pdlrtapi.h

See also

PDLRTCclosePicture (Page 1898)

PDLRTDisableClosePicture (Page 1900)

PDLRTEnableClosePicture (Page 1902)

PDLRTInquireFreeArea (Page 1905)

PDLRTOpenPicture (Page 1906)

PDLRTShowApp (Page 1910)

PDLRTSetLink (Page 1935)

PDLRTSetMultiLink (Page 1937)

PDLRTGetLink (Page 1933)
 PDLRTSetPropEx (Page 1928)
 PDLRTGetPropEx (Page 1925)
 PDLRTGetDefPropEx (Page 1923)
 PDLRTSetFocus (Page 1921)
 PDLRTSetCursorKeys (Page 1919)
 PDLRTGetFocus (Page 1917)
 PDLRTGetCursorKeys (Page 1915)
 Overview of the functions (Page 1867)

3.3.4 Functions for affecting the runtime cursor (RT Professional)

3.3.4.1 PDLRTGetCursorKeys (RT Professional)

Use

This function queries the keys for cursor control that have been set with PDLRTSetCursorKeys.

Declaration

```

BOOL PDLRTGetCursorKeys (
    long*           pKeyUp,
    long*           pKeyDown,
    long*           pKeyLeft,
    long*           pKeyRight,
    long*           pKeyState,
    long*           pTabMode,
    PDLRT_CALLBACK pfn,
    LPVOID          pvUser,
    PCMN_ERROR      pError );
  
```

Parameters

KeyUp

This pointer provides the virtual key code (VK_...) of the key that moves the cursor down.

pKeyDown

This pointer provides the virtual key code (VK_...) of the key that moves the cursor up.

pKeyLeft

This pointer provides the virtual key code (VK_...) of the key that moves the cursor to the left.

pKeyRight

This pointer provides the virtual key code (VK_...) of the key that moves the cursor to the right.

pKeyState

This pointer provides the keyboard status:

HOTKEYF_SHIFT	(value: 0x01)	"SHIFT" key pressed
HOTKEYF_CONTROL	(value: 0x02)	"CTRL" key pressed
HOTKEYF_ALT	(value: 0x04)	"ALT" key pressed

The values can be ORed as desired.

pTabMode

This pointer provides the TabMode, the object that gets the focus with the next activation of a cursor control key:

O:	xxScreenxx
1:	xxScreenxx
10:	xxScreenxx
	xxScreenxx

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Cursor control keys determined.

FALSE

Error.

Required files

pdlrtapi.h
 pdlrt_s.lib
 pdlrtapi.dll

Related functions

PDLRTSetCursorKeys (Page 1919)	Set cursor control keys
PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRTSetCursorKeys (Page 1919)
 PDLRT_CALLBACK (Page 1911)
 Overview of the functions (Page 1867)

3.3.4.2 PDLRTGetFocus (RT Professional)**Use**

The function returns the screen and the object name of the object that has the input focus.

Declaration

```

BOOL PDLRTGetFocus (
    ADRMODE          adrMode,
    LPFOCUSINFO      pFocusInfo,
    PDLRT_CALLBACK   pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );

```

Parameters**adrMode**

The adrMode parameter is used to set the addressing mode of the screen to be edited.

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	1	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The PDLRT_AM_PICTABS and PDLRT_AM_OBJABS value can be ORed.

pFocusInfo

Pointer to the FOCUSINFO (Page 1898) structure in which the results are stored.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If `pfn = NULL` is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Input focus determined.

FALSE

Error.

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRTSetFocus (Page 1921)	Set input focus
PDLRT_CALLBACK (Page 1911)	Callback function

See also

FOCUSINFO (Page 1898)

PDLRTSetFocus (Page 1921)

[PDLRT_CALLBACK \(Page 1911\)](#)

[Overview of the functions \(Page 1867\)](#)

3.3.4.3 PDLRTSetCursorKeys (RT Professional)

Use

This function keys sets for the keys for cursor control.

Declaration

```

BOOL PDLRTSetCursorKeys (
    long      KeyUp,
    long      KeyDown,
    long      KeyLeft,
    long      KeyRight,
    long      KeyState,
    long      TabMode,
    PDLRT_CALLBACK pfn,
    LPVOID      pvUser,
    PCMN_ERROR  pError );

```

Parameters

KeyUp

Virtual key code (VK_...) of the key that moves the cursor down.

KeyDown

Virtual key code (VK_...) of the key that moves the cursor up.

KeyLeft

Virtual key code (VK_...) of the key that moves the cursor to the left.

KeyRight

Virtual key code (VK_...) of the key that moves the cursor to the right.

KeyState

KeyState provides the keyboard status:

HOTKEYF_SHIFT	(value: 0x01)	"SHIFT" key pressed
HOTKEYF_CONTROL	(value: 0x02)	"CTRL" key pressed
HOTKEYF_ALT	(value: 0x04)	"ALT" key pressed

The values can be ORed as desired.

TabMode

TabMode specifies the object that gets the focus with the next activation of a cursor control key:

0:	xxScreenxx
1:	xxScreenxx
10:	xxScreenxx
	xxScreenxx

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Cursor control keys defined.

FALSE

Error.

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRTGetCursorKeys (Page 1913)	Query cursor control keys
PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRTGetCursorKeys (Page 1913)

PDLRT_CALLBACK (Page 1911)

Overview of the functions (Page 1867)

3.3.4.4 PDLRTSetFocus (RT Professional)**Use**

The function is used to set the input focus. The object is determined by `lpszPictureName` and `lpszObjectName`.

Declaration

```

BOOL PDLRTSetFocus (
    ADRMODE          adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );

```

Parameters**adrMode**

The `adrMode` parameter is used to set the addressing mode of the screen to be edited.

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	1	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The `PDLRT_AM_PICTABS` and `PDLRT_AM_OBJABS` value can be ORed.

lpszPictureName

Pointer to the configured name of the screen according to addressing mode defined by `adrMode` without extension. The entry is case-sensitive.

lpszObjectName

Pointer to the configured object name according to addressing mode defined by `adrMode`.

3.3 Functions of the graphic system (RT Professional)

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Input focus set

FALSE

Error

Error messages

PDLRT_NO_PIC	Screen not selected
PDLRT_NO_OBJ	Cannot find object

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRTGetFocus (Page 1915)	Query input focus
PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRTGetFocus (Page 1915)

PDLRT_CALLBACK (Page 1911)

Overview of the functions (Page 1867)

3.3.5 Functions for processing object properties (RT Professional)**3.3.5.1 PDLRTGetDefPropEx (RT Professional)****Use**

Request a default property value. If the requested data type does not match the data type of the property, the value is changed during the transfer into the pvProp user buffer in accordance with the requested format, if this is possible.

Declaration

```

BOOL PDLRTGetDefPropEx (
    ADRMODE          adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    VARTYPE           vt,
    LPVOID            pvProp,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    DWORD             dwFlags,
    LPVOID            pData,
    PCMN_ERROR        pError );

```

Parameters**adrMode**

The adrMode parameter is used to set the addressing mode of the screen to be edited.

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	0	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The PDLRT_AM_PICTABS and PDLRT_AM_OBJABS value can be ORed.

lpszPictureName

Pointer to the configured name of the screen according to addressing mode defined by adrMode. The entry is case-sensitive.

lpszObjectName

Pointer to the configured object name according to addressing mode defined by `via adrMode`.

lpszPropName

Pointer on the configured name of the object property.

vt

Data type of a value passed under `pvProp`. The allowed types are defined in the include `oidl.h` and `"wtypes.h"` files of the compiler. You can use the WinCC include file, `APLIB\Defines.h`, for scripts.

pvProp

Pointer to a tag in which the property value is stored. The data type of the value is determined by `vt`:

vt	pvProp
VT_BSTR	BSTR*
VT_LPSTR	LPSTR*
VT_LPWSTR	LPWSTR*
VT_UI4	LONG*
VT_XXXX	pvProp must point to a buffer

For types with a buffer (for example, BSTR), the buffer is allocated by the function and must be released again by the calling application.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If `pfn = NULL` is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

dwFlags

The parameter is reserved for future development and must be preset to 0.

pData

The parameter is reserved for future development and must be preset to NULL.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Default values of the object property determined.

FALSE

Error.

Comments

To release allocated memory areas, use the VT_BSTR Windows API function for SysFreeString(...) and the VT_LPSTR Windows API function for VT_LPWSTR, LocalFree(...).

Error messages

PDLRT_NO_PROP	Cannot find property
PDLRT_BAD_OLE_CONVERSION	Error in conversion through OLE automation

Required files

pdlrtapi.h
pdlrt_s.lib
pdlrtapi.dll

Related functions

PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRT_CALLBACK (Page 1911)
Overview of the functions (Page 1867)

3.3.5.2 PDLRTGetPropEx (RT Professional)**Use**

Request a current property value. The source is determined by lpszPictureName, lpszObjectName and lpszPropName.

If the requested data type does not match the data type of the property, the value is changed during the transfer into the pvProp user buffer in accordance with the requested format, if this is possible.

Declaration

```

BOOL PDLRTGetPropEx (
    ADRMODE      adrMode,
    LPCSTR       lpszPictureName,
    LPCSTR       lpszObjectName,
    LPCSTR       lpszPropName,
    VARTYPE      vt,
    LPVOID       pvProp,
    PDLRT_CALLBACK pfn,
    LPVOID       pvUser,
    DWORD        dwFlags,
    LPVOID       pData,
    PCMN_ERROR   pError );

```

Parameters

adrMode

The adrMode parameter is used to set the addressing mode of the screen to be edited.

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	0	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The PDLRT_AM_PICTABS and PDLRT_AM_OBJABS value can be ORed.

lpszPictureName

Pointer to the configured name of the screen according to addressing mode defined by adrMode. The entry is case-sensitive.

lpszObjectName

Pointer to the configured name of the object according to addressing mode defined by adrMode.

lpszPropName

Pointer on the configured name of the object property.

vt

Data type of a value passed under pvProp. The allowed types are defined in the include oaidl.h" and "wtypes.h" files of the compiler. You can use the WinCC include file, APLIB\Defines.h , for scripts.

pvProp

Pointer to a tag in which the property value is stored. The data type of the value is determined by vt:

vt	pvProp
VT_BSTR	BSTR*
VT_LPSTR	LPSTR*
VT_LPWSTR	LPWSTR*
VT_UI4	LONG*
VT_XXXX	pvProp must point to a buffer

For types with a buffer (for example, BSTR), the buffer is allocated by the function and must be released again by the calling application.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

dwFlags

The parameter is reserved for future development and must be preset to 0.

pData

The parameter is reserved for future development and must be preset to NULL.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Object properties determined.

FALSE

Error.

Comments

To release allocated memory areas, use the, use the SysFreeString(...) Windows API function for VT_BSTR and the LocalFree(...) Windows API function for VT_LPSTR, VT_LPWSTR .

No VT_DISPATCH or other reference may be used; only the normal types and a VT_VARIANT at the most for an array property of a simple type.

Error messages

PDLRT_NO_PIC	Screen not selected
PDLRT_BAD_OLE_CONVERSION	Error in conversion through OLE automation
PDLRT_NO_PROP	Cannot find property

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRTSetPropEx (Page 1928)	Set property
PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRT_CALLBACK (Page 1911)

PDLRTSetPropEx (Page 1928)

Overview of the functions (Page 1867)

3.3.5.3 PDLRTSetPropEx (RT Professional)**Use**

Set a property. The target is determined by lpszPictureName, lpszObjectName and lpszPropName. When the passed data type does not match the data type of the property, the value is changed during the transfer to the property, if this is possible.

Only those properties that can also be made dynamic can be set.

Declaration

```

BOOL PDLRTSetPropEx (
    ADRMODE          adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    VARTYPE           vt,
    LPVOID            pvProp,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    DWORD             dwFlags,
    LPVOID            pData,
    PCMN_ERROR        pError );

```

Parameters

adrMode

The adrMode parameter is used to set the addressing mode of the screen to be edited.

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	0	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The PDLRT_AM_PICTABS and PDLRT_AM_OBJABS value can be ORed.

lpszPictureName

Pointer to the configured name of the screen according to addressing mode defined by adrMode. The entry is case-sensitive.

lpszObjectName

Pointer to the configured object name according to addressing mode defined by adrMode.

lpszPropName

Pointer on the configured name of the object property.

vt

Data type of a value passed under pvProp. The allowed types are defined in the include oaidl.h" and "wtypes.h" files of the compiler. You can use the WinCC include file, APLIB\Defines.h , for scripts.

pvProp

Pointer to a buffer in which the property value is stored. The data type of the value is determined by vt:

vt	pvProp
VT_BSTR	BSTR*
VT_LPSTR	LPSTR*

3.3 Functions of the graphic system (RT Professional)

VT_LPWSTR	LPWSTR*
VT_UI4	LONG*
VT_xxxx	pvProp must point to a buffer

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function, but is made available again in the callback function.

dwFlags

The parameter is reserved for future development and must be preset to 0.

pData

The parameter is reserved for future development and must be preset to ZERO.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Object properties set.

FALSE

Error.

Comments

No VT_DISPATCH or other reference may be used; only the normal types and a VT_VARIANT at the most for an array property of a simple type.

Error messages

PDLRT_NO_PIC	Screen not selected
PDLRT_PICTURE_NOT_LOADED	Cannot load screen file

PDLRT_FAILURE_PARAM	Wrong parameter
PDLRT_NO_OBJ	Cannot find object
PDLRT_NO_PROP	Cannot find property
PDLRT_BAD_OLE_CONVERSION	Error in conversion through OLE automation

Required files

pdLrtapi.h
pdLrt_s.lib
pdLrtapi.dll

Related functions

PDLRTGetPropEx (Page 1923)	Query property
PDLRT_CALLBACK (Page 1911)	Callback function

Example

```
#pragma code("pdlrtapi.dll")

#include "apdefap.h"

#define CP_ACP 0
#define MB_PRECOMPOSED 1

void ButtonRT(char* text)
{
    DM_DIRECTORY_INFO dirInfo;
    CMN_ERROR error;
    BOOL bRet;
    char szProjectFile[222] = "";
    WCHAR wstring[99];
    BSTR bstrval;
    WORD *pw = NULL;
    int i;

    printf("Function ButtonRT() with text '%s'\r\n", text);

    bRet = DMGetRuntimeProject
(&szProjectFile[0], sizeof(szProjectFile), &error);
    printf("DMGetRuntimeProject: %s\r\n", szProjectFile);

    bRet = PDLRTOpenPicture(0, "Start.PDL", NULL, NULL, 0, 0, 0, 0, 0, NULL, NULL,
&error);
    printf("PDLRTOpenPicture: %s,%ld, %s\r\n", bRet?"TRUE":"FALSE",
error.dwError1, error.szErrorText);

    bRet = PDLRTSetPropEx(0, "Start.PDL", "Button4", "Text", VT_LPSTR,
&text, NULL, NULL, 0, NULL, &error);
    printf("PDLRTSetPropEx: %s,%ld, %s\r\n", bRet?"TRUE":"FALSE",
error.dwError1, error.szErrorText);

    SysFreeString( bstrval );
    printf("SysFreeString done\r\n");
}
```

See also[PDLRT_CALLBACK \(Page 1911\)](#)[PDLRTGetPropEx \(Page 1923\)](#)[Overview of the functions \(Page 1867\)](#)

3.3.6 Functions for processing dynamics (RT Professional)

3.3.6.1 PDLRTGetLink (RT Professional)

Use

Query a link from a property. If the property has an indirect tag connection, this is returned. If the current active direct connection is desired for an indirect tag connection, this can be requested by specifying `LinkType = BUBRT_LT_VARIABLE_DIRECT` in the `LINKINFO` structure.

Declaration

```

BOOL PDLRTGetLink (
    ADRMODE          adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    LPLINKINFO        pLink,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );

```

Parameters

adrMode

The `adrMode` parameter is used to set the addressing mode of the screen to be edited.

<code>PDLRT_AM_DEFAULT</code>	0	The addressing of screen and object is relative
<code>PDLRT_AM_PICTABS</code>	1	The addressing of the screen is absolute
<code>PDLRT_AM_OBJABS</code>	2	The addressing of the object is absolute

The `PDLRT_AM_PICTABS` and `PDLRT_AM_OBJABS` value can be ORed.

lpszPictureName

Pointer to the configured name of the screen according to addressing mode defined by `adrMode` without extension. The entry is case-sensitive.

lpszObjectName

Pointer to the configured object name according to addressing mode defined by `adrMode`.

lpszPropName

Pointer on the name of the object property.

pLink

Pointer to the `LINKINFO` (Page 1894) structure in which the link information is stored.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Link determined.

FALSE

Error.

Error messages

PDLRT_NO_LINK	Property has no dynamics

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRTSetLink (Page 1935)	Define link between object property and tag
PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRTSetLink (Page 1935)

PDLRT_CALLBACK (Page 1911)

LINKINFO (Page 1894)

Overview of the functions (Page 1867)

3.3.6.2 PDLRTSetLink (RT Professional)**Use**

Set a link between a property and a tag.

Declaration

```

BOOL PDLRTSetLink (
    ADRMODE          adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    LPLINKINFO        pLink,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );

```

Parameters**adrMode**

The adrMode parameter is used to set the addressing mode of the screen to be edited.

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	1	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The PDLRT_AM_PICTABS and PDLRT_AM_OBJABS value can be ORed.

lpszPictureName

Pointer to the configured name of the screen according to addressing mode defined by adrMode without extension. The entry is case-sensitive.

lpszObjectName

Pointer to the configured object name according to addressing mode defined by adrMode.

lpszPropName

Pointer on the name of the object property.

pLink

Pointer to the LINKINFO (Page 1894) structure in which the link information is stored.

Only BUBRT_LT_VARIABLE_DIRECT and BUBRT_LT_VARIABLE_INDIRECT can be used in LinkType, since only a link with tags is allowed here.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Link set

FALSE

Error

Error messages

PDLRT_FAILURE_PARAM	Wrong parameter
PDLRT_NO_PIC	Screen not selected
PDLRT_NO_OBJ	Cannot find object
PDLRT_LINK_NOT_SET	Cannot link property

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRTGetLink (Page 1931)	Determine link between object property and tag
PDLRTSetMultiLink (Page 1937)	Define link between object property and tag (multiple tags)
PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRTGetLink (Page 1931)
 PDLRTSetMultiLink (Page 1937)
 PDLRT_CALLBACK (Page 1911)
 LINKINFO (Page 1894)
 Overview of the functions (Page 1867)

3.3.6.3 PDLRTSetMultiLink (RT Professional)**Use**

Set multiple links between a property and a tag.

Declaration

```

BOOL PDLRTSetMultiLink (
    ADRMODE          adrMode,
    LPCSTR            lpszPictureName,
    LPMULTILINK       pMultiLink,
    PDLRT_CALLBACK    pfn,
    LPVOID             pvUser,
    PCMN_ERROR        pError );
  
```

Parameters**adrMode**

The adrMode parameter is used to set the addressing mode of the screen to be edited.

PDLRT_AM_DEFAULT	0	The addressing of screen and object is relative
PDLRT_AM_PICTABS	1	The addressing of the screen is absolute
PDLRT_AM_OBJABS	2	The addressing of the object is absolute

The PDLRT_AM_PICTABS and PDLRT_AM_OBJABS value can be ORed.

lpszPictureName

Pointer to the configured name of the screen according to addressing mode defined by adrMode without extension. The entry is case-sensitive.

pMultiLink

Pointer to the MULTILINK (Page 1896) structure in which a field with link information is stored.

pfn

Pointer to your callback function. This is called to notify you about the success or failure of the request.

If pfn = NULL is passed, the call is synchronous. The calling application is set as waiting until PDLRT reports failure or success. This type of call should be preferentially used for mutually dependent call sequences.

If subsequent API calls require this function to be completed and this function is used asynchronously with callback, the API calls should be synchronized with semaphores.

It makes sense to use a separate callback for each API function.

pvUser

Pointer to application-specific data that is passed to the callback function. This pointer is not evaluated by the function but provided again in the callback function.

Return value**TRUE**

The TRUE return value is only returned if all links were successfully set. Even if only one link could not be set, FALSE is returned.

FALSE

Error.

Error messages

PDLRT_FAILURE_PARAM	Wrong parameter
PDLRT_NO_PIC	Screen not selected
PDLRT_NO_OBJ	Cannot find object
PDLRT_LINK_NOT_SET	Cannot link property

Required files

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

Related functions

PDLRTSetLink (Page 1933)	Define link between object property and tag
PDLRT_CALLBACK (Page 1911)	Callback function

See also

PDLRTSetLink (Page 1933)
PDLRT_CALLBACK (Page 1911)
MULTILINK (Page 1896)
Overview of the functions (Page 1867)

3.4 Scripting functions (RT Professional)

3.4.1 Basics (RT Professional)

3.4.1.1 Overview of the functions (RT Professional)

Overview

AP_RT_PROC (Page 1954)	Notification of runtime functions
APActive (Page 1966)	Activate action
APCompile (Page 1957)	Compile source code
APCompileEx (Page 1959)	Compile source code
APConnect (Page 1949)	Logon application
APDisconnect (Page 1951)	Logoff application
APEndAct (Page 1968)	Logon action in the application control
APFreeResultStruct (Page 1969)	Free action result memory
APInactive (Page 1970)	Deactivate actions
APSetLanguage (Page 1953)	Set language for error texts
APStart (Page 1972)	Start action
APTransact (Page 1975)	Logon actions for processing
GSCGenCompile (Page 1962)	Compile action
GSCGenCompileUserFunctions (Page 1963)	Compile user functions
GSCGenGetActionStream (Page 1965)	Determine action stream

3.4.1.2 Overview of structures (RT Professional)

Overview

AP_ACT_KEY (Page 1944)	Action identification
AP_ACT_RESULT_STRUCT (Page 1946)	Action result

CREATE_USER_HEADER_FILE (Page 1947)	Create header file of user functions
GENERATE_COMPILE (Page 1948)	Generate action
GET_ACTION_STREAM (Page 1948)	Determine action stream

3.4.1.3 Error messages (RT Professional)

Overview

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

General error messages

AP_ALREADY_CONNECTED	2	Application is already connected to the action control
AP_NO_CONNECTION	3	Application cannot establish a connection to the action control.
AP_ERROR_IPC_SEND	4	Error in communication with the action control. The job could not be sent.
AP_FAILURE_UNKNOWN	5	Error not described in detail
AP_FAILURE_PARAM	6	Incorrect parameter assignment
AP_NO_ACT_PROGRAM	7	Action control could not be started
AP_TIMEOUT	8	Timeout, not yet implemented at present.
AP_ACT_QUIT	9	Action control has ended
AP_INSTALL_SERV_ERROR	10	Service channel could not be installed
AP_ENDACT_UNKNOWN_ORDER	11	An unknown job number was used for EndAct
AP_ACTION_FAILED	12	An action could not be executed without error. The return results are invalid.
AP_FAILURE_IN_SERVER	13	Server is signaling an error.
AP_TO_MANY_CLIENTS	14	The maximum number of connections is reached. No new connections possible.
AP_TRANSACTID_UNKNOWN	15	Invalid transaction ID. Error occurs when calling AP_EndAct if an attempt is made to end a transaction that was not logged on beforehand.
AP_NO_MEMORY	16	There is no longer enough memory available for this operation
AP_TRANSACT_ERROR	17	An error occurred within the transaction; action-related error in dwerror of AP_ACT_KEY
AP_RESULT_TRANS_ERROR	18	Error occurred in the results.
AP_RESULT_START_ERROR	19	Error occurred in the results.
AP_NO_UPDATE_WRONG_FORMAT	50	Update option not available for the data format. Action contains no interpreter code.
AP_ERR_WRONG_FORMAT	202	Data format of action is incorrect.
AP_NO_VALID_FUNCTION_VALUE	203	Function return value cannot be converted to VARIANT data type.

Errors

Errors that may occur when performing an action

AP_CISS_ERR_EXIT_OVERFLOW	1001	A stack overflow occurred in the action interpreter during execution. Further execution of action is cancelled.
AP_CISS_ERR_EXIT_DIVIDE0	1002	Division by 0 occurred while executing an action. Canceling action.
AP_CISS_ERR_EXIT_UNRESOLVED	1003	While executing the action, a symbol was referenced that does not exist.
AP_CISS_ERR_EXIT_GPF	1004	While executing the action, memory was accessed that is not defined.
AP_CISS_ERR_EXIT_BREAKPOINT	1005	The action interpreter stopped due to a breakpoint.
AP_CISS_ERR_EXIT_STEP	1006	The action interpreter was resumed in the next processing step.

Alarms

Alarms of the action interpreter

AP_CISS_ERR_CREATE_PCH_FROM_PCH	8001	A precompiled header file cannot be generated from a precompiled header file.
AP_CISS_ERR_MODULE_IN_USE	8002	The action cannot be accessed. Module is currently in use.
AP_CISS_ERR_INVALID_PROGRAM	8003	The program is invalid.
AP_CISS_ERR_INVALID_MODULE	8004	The action is invalid.
AP_CISS_ERR_CANNOT_CREATE_FILE	8005	File could not be created by the action control.
AP_CISS_ERR_CANNOT_NO_MEMORY	8006	The action interpreter no longer has enough memory.
AP_CISS_ERR_INVALID_FILE_FORMAT	8007	File format is invalid for the action control.
AP_CISS_ERR_CANNOT_OPEN_FILE	8008	File could not be opened by the action control.
AP_CISS_ERR_PROGRAM_IS_LOCKED	8009	Program is locked by the action control at present.
AP_CISS_ERR_MODULE_ALREADY_INSERTED	8010	Action was already provided to the action control for processing.
AP_CISS_ERR_CONFLICT_WITH_OTHER	8011	During the action, a conflict occurred with another action.
AP_CISS_ERR_MODULE_NOT_FOUND	8013	Action was not found by the action control.
AP_CISS_ERR_FUNCTION_NOT_FOUND	8014	Function was not found by the action control.
AP_CISS_ERR_INVALID_LINE	8015	Specified line information is invalid.
AP_CISS_ERR_INVALID_SCOPE	8016	Specified symbol is outside the valid range.

3.4 Scripting functions (RT Professional)

AP_CISS_ERR_BUFFER_TOO_SMALL	8017	Transferred memory is too small for the action interpreter.
AP_CISS_ERR_INVALID_TYPE	8018	The specified type is unknown to the action interpreter.
AP_CISS_ERR_SYMBOL_NOT_FOUND	8019	The specified symbol was not found.

Notify codes

AP_NOTIFY_ERROR	0	An error occurred while performing the function.
AP_NOTIFY_DATA	1	No error occurred while performing the function.
AP_NOTIFY_CODE_TRANSACT	1	Confirmation of an AP_TransAct call
AP_NOTIFY_CODE_START	2	Confirmation of an AP_Start call
AP_NOTIFY_CODE_RESULT	3	Result of an action
AP_NOTIFY_CODE_DISCONNECT	6	
AP_NOTIFY_CODE_ENDACT	7	Confirmation of an AP_EndAct call
AP_NOTIFY_CODE_ACTIVE	8	Confirmation of an AP_Active call
AP_NOTIFY_CODE_INACTIVE	9	Confirmation of an AP_Inactive call
AP_NOTIFY_ERROR_SERVER_QUITT	10	Notification that the action control has been closed.
AP_NOTIFY_CODE_TRANSRESULT	11	Notification of an erroneous result. Call of an AP-TransAct
AP_NOTIFY_CODE_STARTRESULT	12	Notification of an erroneous result. Call of an AP-Start.
AP_NOTIFY_CODE_RESULT_RT	13	Notification of cyclic results of an optimized call

3.4.1.4 Constants (RT Professional)

Function/Action IDs

GSC_AP_SFCT	0x00000011	Standard function
GSC_AP_PFCT	0x00000012	Project function
GSC_AP_GSC	0x00000014	GSC action

Limit values

AP_MAX_TRIG_NAME	21	Maximum length of the name of a trigger

Header types

CMHF_APDEFAP	0x00000001	Header of a project definition
CMHF_AP_PBIB	0x00000002	Header of a project function
CMHF_AP_GLOB	0x00000004	Header of a standard function
CMHF_AP_ICF	0x00000008	Header of an internal function

CMHF_AP_ALL	CMHF_APDEFAP CMHF_AP_PBIB CMHF_AP_GLOB CMHF_AP_ICF	
CMHF_AP_USER	0x00000010	Header of a user function

Trigger types

AP_TRIG_UNDEFINED	0	Trigger is not yet initialized.
AP_TRIG_VAR	1	Trigger is a tag.
AP_TRIG_TIMER	2	Trigger is a timer.
AP_TRIG_UNKNOWN	3	Trigger is unknown.
AP_TRIG_TRANSACT	4	Trigger is specified via a TransAct call.

Cycle types

AP_TRIG_CYCLE	1	Cyclic
AP_TRIG_NCYCLE	2	Acyclic

Cycle times

AP_TRIG_CYCLE_01	1	Reflects cycle time of 250 ms in the Control Center
AP_TRIG_CYCLE_02	2	Reflects cycle time of 500 ms in the Control Center
AP_TRIG_CYCLE_03	3	Reflects cycle time of 1 s in the Control Center
AP_TRIG_CYCLE_04	4	Reflects cycle time of 2 s in the Control Center
AP_TRIG_CYCLE_05	5	Reflects cycle time of s in the Control Center
AP_TRIG_CYCLE_06	6	Reflects cycle time of 10 s in the Control Center
AP_TRIG_CYCLE_07	7	Reflects cycle time of 1 min in the Control Center
AP_TRIG_CYCLE_08	8	Reflects cycle time of 5 min in the Control Center
AP_TRIG_CYCLE_09	9	Reflects cycle time of 10 min in the Control Center
AP_TRIG_CYCLE_10	10	Reflects cycle time of 1 h in the Control Center
AP_TRIG_CYCLE_11	11	Reflects cycle time of "User defined 1" in the Control Center
AP_TRIG_CYCLE_12	12	Reflects cycle time of "User defined 2" in the Control Center
AP_TRIG_CYCLE_13	13	Reflects cycle time of "User defined 3" in the Control Center
AP_TRIG_CYCLE_14	14	Reflects cycle time of "User defined 4" in the Control Center
AP_TRIG_CYCLE_15	15	Reflects cycle time of "User defined 5" in the Control Center

3.4.2 Structures (RT Professional)

3.4.2.1 AP_ACT_KEY (RT Professional)

Declaration

```
typedef struct {
    DWORD      dwKeyType;
    DWORD      dwID;
    CHAR       szActionName[AP_MAX_ACTION_NAME + 1];
    DWORD      dwCycle;
    VARIANT    *pVariant;
    DWORD      dwVariantItem;
    DWORD      dwerror;
    LPVOID     lpvUser;
} AP_ACT_KEY ;
```

Members

dwKeyType

The key type determines whether the action is addressed by a name (szActionName) or by an ID (dwID).

AP_ID_TYPE	0	Identification of the action by its ID
AP_NAME_TYPE	1	Identification of the action by its name

Project and standard functions which already exist in in the WinCC project are identified by the name. User-defined functions that are first logged on with the APTransAct function are identified by the ID.

dwID

In connection with dwKeyType = AP_ID_TYPE the action is started with dwID.

The ID of the action is assigned on calling the APTransAct function. On calling the APTransAct function, dwID must be defaulted with 0.

szActionName

In connection with dwKeyType = AP_NAME_TYPE the action is started by szActionName.

Can be used when starting an action with APStart.

dwCycle

Cyclic starting of the action when addressing is by name. The upgrade cycle is defined by the index of the entries in the list of update cycles.

pVariant

Pointer to field of VARIANT data type which describes the parameters of the action. The following data types are supported as VARTYPES:

unsigned char	VT_UI1
unsigned integer 2 Bytes	VT_UI2 (WORD)
unsigned integer 4 Bytes	VT_UI4 (DWORD)
short	VT_I2
long	VT_I4
float	VT_R4
double	VT_R8
VARIANT_BOOL	VT_BOOL
SCODE	VT_ERROR
CY	VT_CY
DATE	VT_DATE
BSTR	VT_BSTR
BLOB	VT_BLOB

The parameter descriptions of the actions and the return result are only possible in the form of the VARIANT data type.

Transfer as a reference is not permitted.

dwVariantItem

Number of VARIANT data types which the action receives as parameters.

dwerror

Error number which can occur in the APTransAct.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

Description

The AP_ACT_KEY structure clearly identifies the action. This key is assigned as an ID with the API call APTransAct. The action is started with this key.

Required files

ap_def.h

API functions

APActive (Page 1966)	Activate action
APInactive (Page 1970)	Deactivate actions

APStart (Page 1972)	Start action
APTransact (Page 1975)	Logon actions for processing

See also

APActive (Page 1966)

APInactive (Page 1970)

APStart (Page 1972)

APTransact (Page 1975)

3.4.2.2 AP_ACT_RESULT_STRUCT (RT Professional)**Declaration**

```
typedef struct {  
    VARIANT      *ap_result;  
    AP_ACT_KEY    apActKey;  
    CMN_ERROR     error;  
    DWORD         dwreserved;  
} AP_ACT_RESULT_STRUCT ;
```

Members***ap_result**

Return result of the action in the form of a VARIANT data type.

apActKey

Key of the action for identification. The AP_ACT_KEY corresponds to the key with which the action was started.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

dwreserved

The parameter is reserved for future development and must be preset to 0.

Required files

ap_def.h

API functions

APFreeResultStruct (Page 1969)	Free action result memory
APStart (Page 1972)	Start action

See also

APFreeResultStruct (Page 1969)

APStart (Page 1972)

3.4.2.3 CREATE_USER_HEADER_FILE (RT Professional)**Declaration**

```
typedef struct {
    char*      pszStartDir;
    char*      pszHeaderFileName;
    BOOL       bShowDlg;
    char*      pszWindowText;
} CREATE_USER_HEADER_FILE ;
```

Members**pszStartDir**

Pointer to the start directory at which the search for user functions begins with terminating "\"

pszHeaderFileName

Pointer to the name of the header file

bShowDlg

Show dialog with progress display

pszWindowText

Pointer to the dialog label.Default:BROWSER

Required files

capigsc.h

API functions

GSCGenCompileUserFunctions (Page 1963)	Compile user functions

See also

GSCGenCompileUserFunctions (Page 1963)

3.4.2.4 GENERATE_COMPILE (RT Professional)

Declaration

```
typedef struct {
    char*      pszProjectName;
    LPACTION   pAction;
} GENERATE_COMPILE ;
```

Members

- pszProjectName**
Pointer to the project name
- pAction**
Pointer to valid action stream with source code

Required files

capigsc.h

Related functions

GSCGenCompile (Page 1962)	Compile action

See also

GSCGenCompile (Page 1962)

3.4.2.5 GET_ACTION_STREAM (RT Professional)

Declaration

```
typedef struct {
    char*      pszPathName;
    DWORD      dwType;
} GET_ACTION_STREAM, *LPGET_ACTION_STREAM ;
```

Members**pszPathName**

Full filename of the function or action

dwType

Permissible types:

GSC_AP_SFCT	Default function
GSC_AP_PFCT	Project function
GSC_AP_GSC	Action

Required files

capigsc.h

API functions

GSCGenGetActionStream (Page 1965)	Determine action stream

See also

GSCGenGetActionStream (Page 1965)

3.4.3 General functions (RT Professional)**3.4.3.1 APConnect (RT Professional)****Description**

The function logs on in an application in the action control. If a callback function is specified in fpAppBack, the function is performed asynchronously, in fpAppBack == NULL action is carried out synchronously.

Declaration

```

BOOL APConnect (
    LPCSTR      lpszAppName,
    AP_RT_PROC  fpAppBack,
    PDWORD      pdwOrderId,
    LPCVOID     lpvUser,
    PCMN_ERROR  pError )

```

Parameters

lpszAppName

Name of the application with which the application logged onto DMConnect. A DMConnect is therefore necessary first. APConnect is only performed synchronously.

If functions of the script programming are called in actions, the "AktSteu" value (note exact way of writing) must be specified for lpszAppName because the WinCC task has performed the DMConnect with this name.

fpAppBack

Your callback function which receives messages. The message that the action control has ended is reported by this function for example.

If a program logs on a Notify routine, it must empty a message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

pdwOrderId

Job number which is assigned when calling the APTransAct function. In synchronous calling pdwOrderId is not important, in an asynchronous job the job number is supplied in the callback function.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

If lpvUser is not used, it must be assigned with NULL.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Application logged on

FALSE

Error

Error messages

AP_NO_ERROR	No error occurred
AP_NO_CONNECTION	Application cannot establish a connection with the action control.
AP_ERROR_IPC_SEND	Error in communication with the action control. The job could not be sent.
AP_TIMEOUT	Time-out not implemented yet at present.
AP_ALREADY_CONNECTED	Application is already connected with the action control

AP_FAILURE_PARAM	Faulty parameter supply
AP_TO_MANY_CLIENTS	The maximum number of connections is reached. No new connections possible.
AP_FAILURE_UNKNOWN	Error not described in detail
AP_ACT_QUIT	Action control has ended

Required files

ap_def.h
apcli_S.lib
apclient.dll

Related functions

DMConnect (Page 1626)	Set up connection to the data manager
APDisconnect (Page 1951)	Disconnect connection to action control
APTransact (Page 1975)	Logon action for processing
AP_RT_PROC (Page 1954)	Callback function

Examples

Establish connection to script programming (Page 1977) "AP01.cpp"

See also

DMConnect (Page 1626)
APTransact (Page 1975)
AP_RT_PROC (Page 1954)
APDisconnect (Page 1951)
Establish connection to script programming (Page 1977)
Overview of the functions (Page 1937)

3.4.3.2 APDisconnect (RT Professional)**Description**

The function logs off an application in the action control.

Declaration

```
BOOL APDisconnect (
    AP_RT_PROC    fpAppBack,
    PDWORD        pdwOrderId,
    LPCVOID        lpvUser,
    PCMN_ERROR     pError )
```

Parameters

fpAppBack

Your callback function which receives messages. The job is performed asynchronously when using a callback function. With fpAppBack == NULL it takes place synchronously.

If a program logs on a Notify routine, it must empty a message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

pdwOrderId

Job number which is assigned when calling the APTransAct function. In synchronous calling pdwOrderId is not important, in an asynchronous job the job number is supplied in the callback function.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Application logged off

FALSE

Error

Comment

Note

The call must not be used in the destructor of an application (EXE, DLL, OCX, etc.). This may lead to jamming of the call and thus the program due to Microsoft-specific mechanisms.

Error messages

AP_NO_ERROR	No error occurred
AP_NO_CONNECTION	Application cannot establish a connection with the action control.
AP_ERROR_IPC_SEND	Error in communication with the action control. The job could not be sent.
AP_TIMEOUT	Time-out not implemented yet at present.
AP_ALREADY_CONNECTED	Application is already connected with the action control
AP_FAILURE_PARAM	Faulty parameter supply
AP_TO_MANY_CLIENTS	The maximum number of connections is reached. No new connections possible.
AP_FAILURE_UNKNOWN	Error not described in detail
AP_ACT_QUIT	Action control has ended

Required files

ap_def.h
apcli_S.lib
apclient.dll

Related functions

APConnect (Page 1947)	Set up connection to action control
APTransact (Page 1975)	Logon action for processing
AP_RT_PROC (Page 1954)	Callback function

Examples

Establish connection to script programming (Page 1977) "AP01.cpp"

See also

APTransact (Page 1975)
APConnect (Page 1947)
AP_RT_PROC (Page 1954)
Establish connection to script programming (Page 1977)
Overview of the functions (Page 1937)

3.4.3.3 APSetLanguage (RT Professional)

Description

The language in which the error texts are to be output is set with this function.

Declaration

```
BOOL APSetLanguage (
    const DWORD    dwLanguageID )
```

Parameters

dwLanguageID

ID of the language according to the Windows language setting. The error texts are returned according to the new set language. A default language is used if the language is not supported.

Return value

TRUE

Language of the error texts changed

FALSE

Error

Required files

ap_def.h
apcli_S.lib
apclient.dll

See also

Overview of the functions (Page 1937)

3.4.3.4 AP_RT_PROC (RT Professional)

Description

If your application is to be informed asynchronously about the execution of API functions in runtime, you must provide callback functions of the AP_RT_PROC type.

This function is used by all functions of the action programming as a callback function.

Declaration

```

BOOL ( * AP_RT_PROC) (
    DWORD      dwAP_Notify,
    WORD       dwAP_NotifyCode,
    DWORD      dwError,
    LPVOID     lpvData,
    DWORD      dwItems,
    DWORD      dwOrderId,
    LPVOID     lpvUser );

```

Parameters

dwAP_Notify

Describes the type of callback function. Possible values are AP_NOTIFY_ERROR and AP_NOTIFY_DATA.

dwAP_NotifyCode

If dwAP_Notify == AP_NOTIFY_ERROR, lpvData points to a structure of the CMN_ERROR type with the error description and dwAP_NotifyCode is NULL.

If dwAP_Notify == AP_NOTIFY_DATA, then dwAP_NotifyCode contains a more exact specification of the callback function:

AP_NOTIFY_CODE_TRANSACT	Confirmation of a call by APTransAct
AP_NOTIFY_CODE_START	Confirmation of a call by APStart
AP_NOTIFY_CODE_RESULT	Result of an action

dwError

Error number

lpvData

Pointer to provided data. The data structure depends on dwAP_Notify and dwAP_NotifyCode.

dwItems

Number of entries in lpvData.

dwOrderId

Job number which was assigned when calling the API function.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

Return value**TRUE**

Function successful

FALSE

Error

Comment

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

In rare cases, the callback may already be delivered before the function call has returned.

Required files

ap_def.h

Related functions

APActive (Page 1966)	Activate action
APConnect (Page 1947)	Logon application
APDisconnect (Page 1949)	Logoff application
APEndAct (Page 1968)	Logon action in the application control
APInactive (Page 1970)	Deactivate actions
APStart (Page 1972)	Start action
APTransact (Page 1975)	Logon actions for processing

Examples

Establish connection to script programming (Page 1977) "AP01.cpp"

See also

APConnect (Page 1947)
 APDisconnect (Page 1949)
 APActive (Page 1966)
 APEndAct (Page 1968)
 APInactive (Page 1970)
 APStart (Page 1972)
 APTransact (Page 1975)
 Establish connection to script programming (Page 1977)
 Overview of the functions (Page 1937)

3.4.4 Functions for processing Source Code (RT Professional)**3.4.4.1 APCompile (RT Professional)****Description**

A source code saved in `lpvScore` is compiled and saved as a P-code in `lpvPcode`. The application must provide the memory for the P-code.

Declaration

```

BOOL APCompile (
    LPCSTR      szProjectName,
    PDWORD      pdwOrderId,
    LPCSTR      lpvScore,
    const DWORD dwScoreSize,
    LPVOID      *lpvPcode,
    PDWORD      dwPcodeSize,
    const HWND  hwndLog,
    const DWORD dwDebugFlag,
    PDWORD      nErrors,
    PDWORD      nWarnings,
    LPCVOID     lpvUser,
    PCMN_ERROR  pError )
  
```

Parameters**szProjectName**

Character string which contains a valid project path. The project path decides which Include and Precompiled Header are used for compiling.

pdwOrderId

Job number which is assigned when calling the APTransAct function. The job number must be provided by the caller.

lpvScode

Pointer to the source code to be compiled.

dwScodeSize

Size of the source code in bytes.

lpvPcode

Address of a pointer which contains the P-code in successful compiling. The memory is created by the APCompile function and must be freed by the caller again with the APFreePCode function.

dwPcodeSize

Contains the size of the P-code in bytes after compiling.

hwndLog

Window handle with which the error messages can be output. The window handle may also be NULL. When using a window for the output, a message is sent by the Windows function WM_COPYDATA.

dwDebugFlag

This flag determines whether the P-code should also supply debug information (bit1 = 1) or whether no debug information is to be supplied (bit1 = 0). "0" should normally be selected based on the performance.

Bit2 = 1: Use server settings for Includes

Bit2 = 0: Use local settings for Includes

nErrors

Number of errors occurred. No P-code is returned for nErrors > 0.

nWarnings

Number of warnings occurred. The P-code is also generated for nWarnings > 0.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the Callback-Funktion.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Source code compiled successfully

FALSE

Error

Required files

ap_def.h

apcli_S.lib

apclient.dll

Related functions

APCompileEx (Page 1959)	Compile source code
APFreePCode	Free source code memory

See also

APCompileEx (Page 1959)

Overview of the functions (Page 1937)

3.4.4.2 APCompileEx (RT Professional)

Description

Unlike the APCompile function, the source code to be compiled is divided into subsections.

In the source code void function {instructions} is divided into

void function {	Start of the instruction
Instructions	Base
}	End of the instruction

A source code saved in the subsections lpvScodeProlog, lpvScodeBase and lpvScodeEpilog is compiled and saved as a P-code in lpvPcode. The application must provide the memory for the P-code.

Declaration

```

BOOL APCompileEx (
    LPCSTR      szProjectName,
    PDWORD      pdwOrderId,
    LPCSTR      lpvScodeProlog,
    const DWORD dwScodePrologSize,
    LPCSTR      lpvScodeBase,
    const DWORD dwScodeBaseSize,
    LPCSTR      lpvScodeEpilog,
    const DWORD dwScodeEpilogSize,
    LPVOID      *lpvPcode,
    PDWORD      dwPcodeSize,
    const HWND  hwndLog,
    const DWORD dwDebugFlag,
    PDWORD      nErrors,
    PDWORD      nWarnings,
    LPCVOID     lpvUser,
    PCMN_ERROR  pError )

```

Parameters

szProjectName

Character string which contains a valid project path. The project path decides which Include and Precompiled Header are used for compiling.

pdwOrderId

Job number which is assigned when calling the APTransAct function. The job number must be provided by the caller.

szProjectName

Character string which contains a valid project path. The project path decides which Include and Precompiled Header are used for compiling.

pdwOrderId

Job number which is assigned when calling the APTransAct function. The job number must be provided by the caller.

lpvScodeProlog

Pointer to the prolog of the source code to be compiled.

dwScodePrologSize

Size of the prolog in bytes.

lpvScodeBase

Pointer to the base of the source code to be compiled.

dwScodeBaseSize

Size of the base in bytes.

IpvScodeEpilog

Pointer to the epilog of the source code to be compiled.

dwScodeEpilogSize

Size of the epilog in bytes.

IpvPcode

Address of a pointer which contains the P-code in successful compiling. The memory is created by the APCompile function and must be freed by the caller again with the APFreePCode function.

dwPcodeSize

Contains the size of the P-code in bytes after compiling.

hwndLog

Window handle with which the error messages can be output. The window handle may also be NULL. When using a window for the output, a message is sent by the Windows function WM_COPYDATA.

dwDebugFlag

This flag determines whether the P-code should also supply debug information (bit1 = 1) or whether no debug information is to be supplied (bit1 = 0). "0" should normally be selected based on the performance.

Bit2 = 1: Use server settings for Includes

Bit2 = 0: Use local settings for Includes

nErrors

Number of errors occurred. No P-code is returned for nErrors > 0.

nWarnings

Number of warnings occurred. The P-code is also generated for nWarnings > 0.

IpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the Callback-Funktion.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Source code compiled successfully

FALSE

Error

Required files

- ap_def.h
- apcli_S.lib
- apclient.dll

Related functions

APCompile (Page 1955)	Compile source code
APFreePCode	Free source code memory

See also

- APCompile (Page 1955)
- Overview of the functions (Page 1937)

3.4.4.3 GSCGenCompile (RT Professional)

Description

The function compiles an action.

Declaration

```
LPACTION GSCGenCompile(  
    LPGENERATE_COMPILE lpGenCompile,  
    HWND                hWndParent,  
    unsigned long*       plErrors,  
    unsigned long*       plWarnings,  
    AllocAppMem          lpfnAllocAppMem,  
    LPCMN_ERROR          lpdmError)
```

Parameters

lpGenCompile

Pointer to the GENERATE_COMPILE (Page 1946) structure.

hWndParent

Handle to the window in which the status outputs are to be made.

plErrors

Pointer to variable for return of number of errors

plWarnings

Pointer to variable for return of number of warnings

lpfnAllocAppMem

Pointer to a function with which memory is to be allocated in the calling application

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

Pointer to the compiled action stream. A new action stream is generated even if the compilation is faulty. The success of the call must be checked based on the number of errors and warnings.

Comment

The compiled action is not available until the next time the runtime starts.

Required files

capigsc.h

gscgr_s.lib

gscgen.dll

See also

GENERATE_COMPILE (Page 1946)

Overview of the functions (Page 1937)

3.4.4.4 GSCGenCompileUserFunctions (RT Professional)**Description**

The function recompiles all special user functions.

Declaration

```
BOOL GSCGenCompileUserFunctions (  
    LPCREATE_USER_HEADER_FILE    pGenCUHF,  
    HWND                          hWndParent,  
    LPCMN_ERROR                   lpdmError)
```

Parameters

pGenCUHF

Pointer to the CREATE_USER_HEADER_FILE (Page 1945) structure.

hWndParent

Handle to the window in which the status outputs are to be made.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

User functions compiled

FALSE

Error

Comment

For internal reasons this function always returns FALSE without an error message although the functions are generated.

The compiled user functions are not available until the next time the runtime is started.

Required files

capigsc.h

gscgr_s.lib

gscgen.dll

See also

CREATE_USER_HEADER_FILE (Page 1945)

Overview of the functions (Page 1937)

3.4.5 Functions for processing actions (RT Professional)

3.4.5.1 GSCGenGetActionStream (RT Professional)

Description

The function determines the action stream for a project function, a standard function or an action.

Declaration

```
LPACTION GSCGenGetActionStream(  
    LPGET_ACTION_STREAM    pGenGAS,  
    AllocAppMem            lpfnAllocAppMem,  
    LPCMN_ERROR            lpdmError)
```

Parameters

pGenGAS

Pointer to structure of the GET_ACTION_STREAM (Page 1946) type with which the function or action is specified.

lpfnAllocAppMem

Pointer to a function with which memory is to be allocated in the calling application

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

Pointer to the action stream.

Required files

capigsc.h
gscgr_s.lib
gscgen.dll

See also

GET_ACTION_STREAM (Page 1946)
Overview of the functions (Page 1937)

3.4.6 Functions of the action programming (RT Professional)

3.4.6.1 AActive (RT Professional)

Description

Actions which were deactivated with APInactive can only be reactivated with this function, i.e. the appropriate triggers are monitored again or can be restarted with APStart.

Declaration

```

BOOL AActive (
    PAP_ACT_KEY    lpapActKey,
    const DWORD    dwItems,
    AP_RT_PROC     fpAppBack,
    PDWORD         pdwOrderId,
    LPCVOID        lpvUser,
    PCMN_ERROR     pError )

```

Parameters

lpapActKey

Pointer to a data structure with the following structure:

AP_ACT_KEY 1
AP_ACT_KEY 2
:
AP_ACT_KEY n
Actionstream 1
:
Actionstream 1
Actionstream n

The data structure must be provided by the calling application.

The actions must be logged on with APTransAct first. The ID in AP_ACT_KEY decides whether the action is to be started by the ID or the name.

dwItems

Number of actions, i.e. structures AP_ACT_KEY (Page 1942).

fpAppBack

Your callback function for the asynchronous call. The call is performed synchronously when using NULL.

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

pdwOrderId

Job number which is assigned when calling the APTransAct function. In synchronous calling pdwOrderId is not important, in an asynchronous job the job number is supplied in the callback function.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Action activated

FALSE

Error.

Required files

ap_def.h

apcli_S.lib

apclient.dll

Related functions

APInactive (Page 1970)	Deactivate actions
APTransact (Page 1975)	Logon action for processing
AP_RT_PROC (Page 1952)	Callback function

See also

AP_RT_PROC (Page 1952)

AP_ACT_KEY (Page 1942)

APInactive (Page 1970)

APTransact (Page 1975)

Overview of the functions (Page 1937)

3.4.6.2 APEndAct (RT Professional)

Description

The action logged on for processing is logged off in the action control.

Declaration

```
BOOL APEndAct (
    AP_RT_PROC      fpAppBack,
    PDWORD          pdwOrderId,
    const PDWORD    pdwOrderEnd,
    LPCVOID         lpvUser,
    PCMN_ERROR      pError )
```

Parameters

fpAppBack

Your callback function for the asynchronous call. The call is performed synchronously when using NULL.

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

pdwOrderId

Job number which is assigned when calling the APTransAct function. In synchronous calling pdwOrderId is not important, in an asynchronous job the job number is supplied in the callback function.

pdwOrderEnd

Job number of the transaction which is to be ended.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Action logged off

FALSE

Error.

Required files

ap_def.h

apcli_S.lib

apclient.dll

Related functions

APTransact (Page 1975)	Logon action for processing
AP_RT_PROC (Page 1952)	Callback function

See also

AP_RT_PROC (Page 1952)

APTransact (Page 1975)

Overview of the functions (Page 1937)

3.4.6.3 APFreeResultStruct (RT Professional)**Description**

In synchronous calling of the APStart function the action results are allocated in the form of an array of structures of the AP_ACT_RESULT_STRUCT type. The memory allocated by APStart must be freed again with the APFreeResultStruct function.

Declaration

```

BOOL APFreeResultStruct (
    PAP_ACT_RESULT_STRUCT    *lpapars,
    const DWORD               dwItems )

```

Parameters**lpvPcode**

Pointer to the first dwItems structures of the AP_ACT_RESULT_STRUCT (Page 1944) type.

dwItems

Number of structures AP_ACT_RESULT_STRUCT.

Return value

TRUE
Free memory.

FALSE
Error.

Required files

ap_def.h
apcli_S.lib
apclient.dll

Related functions

APStart (Page 1972)	Processing of action started

See also

APStart (Page 1972)
AP_ACT_RESULT_STRUCT (Page 1944)
Overview of the functions (Page 1937)

3.4.6.4 **APInactive (RT Professional)**

Description

Actions which have been logged on with APTransAct can be deactivated with APInactive. The appropriate triggers are no longer monitored and the actions cannot be started with APStart. The actions can be reactivated with AActive.

Declaration

```
BOOL APInactive (
    PAP_ACT_KEY    lpActKey,
    const DWORD    dwItems,
    AP_RT_PROC      fpAppBack,
    PDWORD          pdwOrderId,
    LPCVOID         lpvUser,
    PCMN_ERROR      pError )
```


Parameters

lpapActKey

Pointer to a data structure with the following structure:

AP_ACT_KEY 1
AP_ACT_KEY 2
:
AP_ACT_KEY n
Actionstream 1
:
Actionstream 1
Actionstream n

The data structure must be provided by the calling application.

The actions must be logged on with APTransAct first. The ID in AP_ACT_KEY decides whether the action is to be started by the ID or the name.

dwItems

Number of actions, i.e. structures AP_ACT_KEY (Page 1942).

fpAppBack

Your callback function for the asynchronous call. The call is performed synchronously when using NULL.

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

pdwOrderId

Job number which is assigned when calling the APTransAct function. In synchronous calling pdwOrderId is not important, in an asynchronous job the job number is supplied in the callback function.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Action deactivated

FALSE

Error.

Required files

- ap_def.h
- apcli_S.lib
- apclient.dll

Related functions

APActive (Page 1964)	Activate actions
APTransact (Page 1975)	Logon action for processing
AP_RT_PROC (Page 1952)	Callback function

See also

- AP_RT_PROC (Page 1952)
- APActive (Page 1964)
- AP_ACT_KEY (Page 1942)
- APTransact (Page 1975)
- Overview of the functions (Page 1937)

3.4.6.5 APStart (RT Professional)

Description

The actions are started for processing. Simultaneous processing of several actions is possible. The actions are described in the AP_ACT_KEY structure.

Declaration

```

BOOL APStart (
    AP_ACT_KEY          lpapActKey,
    const DWORD         dwItems,
    AP_RT_PROC          fpAppBack,
    AP_RT_PROC          fpAppResult,
    PDWORD              pdwOrderId,
    PAP_ACT_RESULT_STRUCT *lpapars,
    LPCVOID             lpUser,
    PCMN_ERROR          pError )

```

Parameters

lpapActKey

Pointer to a data structure with the following structure:

AP_ACT_KEY 1
AP_ACT_KEY 2
:
AP_ACT_KEY n
Actionstream 1
:
Actionstream 1
Actionstream n

The data structure must be provided by the calling application.

The actions must be logged on with APTransAct first. The ID in AP_ACT_KEY decides whether the action is to be started by the ID or the name.

dwItems

Number of actions, i.e. structures AP_ACT_KEY (Page 1942).

fpAppBack

Your callback function for the asynchronous call. The call is performed synchronously when using NULL.

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

fAppResult

Your callback function for cyclic action results. The structures of the AP_ACT_RESULT_STRUCT (Page 1944) type are supplied in the callback function. In synchronous call lpapars is assigned with the result.

pdwOrderId

Job number which is assigned when calling the APTransAct function. In synchronous calling pdwOrderId is not important, in an asynchronous job the job number is supplied in the callback function.

lpapars

Pointer to the action result in synchronous call. An array of structures of the AP_ACT_RESULT_STRUCT (Page 1944) type is returned according to the number of started actions. The memory allocated by the function must be freed again with APFreeResultStruct.

In asynchronous use islpapars = NULL

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

pError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Action started

FALSE

Error.

Comment

With APStart only actions which were not previously deactivated with APInactive can be started.

Required files

ap_def.h

apcli_S.lib

apclient.dll

Related functions

APFreeResultStruct (Page 1967)	Free memory
AP_RT_PROC (Page 1952)	Callback function

See also

AP_RT_PROC (Page 1952)

APFreeResultStruct (Page 1967)

AP_ACT_KEY (Page 1942)

AP_ACT_RESULT_STRUCT (Page 1944)

Overview of the functions (Page 1937)

3.4.6.6 APTransact (RT Professional)

Application

Declaration

Parameter

xxx

xxx

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. The system writes error information into this structure in the case of an error.

Return value

TRUE

.

FALSE

Error

Comment

Required files

ap_def.h

apcli_S.lib

apclient.dll

Related functions

Examples

Auto-Hotspot "DM01.cpp"

See also

APConnect (Page 1947)
APDisconnect (Page 1949)
AP_RT_PROC (Page 1952)
APActive (Page 1964)
APEndAct (Page 1966)
APInactive (Page 1968)
AP_ACT_KEY (Page 1942)
Overview of the functions (Page 1937)

3.4.7 Examples (RT Professional)

3.4.7.1 Establish connection to script programming (RT Professional)

Overview

3.4 Scripting functions (RT Professional)

```
// =====
// Filename:..... ap01.c
// =====
// : Modul with examples of AP_API
// *****
// Copyright (C) 1995/96 SIEMENS AG, AUT 913 All rights reserved
// *****
#include "stdafx.h"
#include "ap01.h" // if console application

//{{ODK_EXAMPLE}Establish connection to script programming (AP)}
//{{FUNCTION}APConnect (AP)}
//{{FUNCTION}APDisconnect (AP)}
//{{FUNCTION}AP_RT_PROC (AP)}
//{{FUNCTION}(END)}
// Establish connection to script programming (AP)
// =====
// Function: AprConnect(void) ODK AP CS
// =====
// short : Establish connection to script programming
// :
// =====
BOOL MyAPRTCallback(DWORD dwAP_Notify, DWORD dwAP_NotifyCode, DWORD dwError,
                    LPVOID lpvData, DWORD dwItems, DWORD dwOrderID, LPVOID lpvUser)
{
    lpvUser;
    dwOrderID;
    lpvData;
    TCHAR szText[255];
    _sntprintf_s( szText, _countof(szText), _TRUNCATE, _T("AprNotCon:: AP= %d ;
                    dwAP_Notify, dwAP_NotifyCode, dwError, dwItems);
    ODKTrace(szText);
    return(TRUE );
}

void MyApConnect(void)
{
    TCHAR szText[255];
    CMN_ERROR Error;
    BOOL ret= FALSE;
    DWORD dwOrderID = 0;
    TCHAR szApp[255];
    VOID* pUser = NULL;
    _tcsncpy_s(szApp, _countof(szApp), _T("MyODKApp_23"), _TRUNCATE); // must be the same
    AppName as by DMConnect
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = APConnect(szApp, MyAPRTCallback, &dwOrderID, pUser, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in APConnect: E1= 0x
        %08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {

```



```

    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("APConnect"));
}
ODKTrace(szText);
//printf("%s\r\n"szText);
void MyAPDisconnect()
VOID* pUser = NULL;
CMN_ERROR Error;
TCHAR szText[255];
BOOL ret = FALSE;
DWORD dwOrderID = 0;
memset(&Error, 0, sizeof(CMN_ERROR));
ret = APDisconnect(NULL, &dwOrderID, pUser, &Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in APDisconnect: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("APDisconnect"));
}
ODKTrace(szText);
}
// -----
//{{ODK_EXAMPLE}(END)}

```

See also

[APConnect \(Page 1947\)](#)
[APDisconnect \(Page 1949\)](#)
[AP_RT_PROC \(Page 1952\)](#)

3.5 Functions of the user administration (RT Professional)

3.5.1 Basics (RT Professional)

3.5.1.1 Overview of the functions (RT Professional)

Overview

PWGEN_ENUM_GROUPS_CALLBACK (Page 2001)	List user groups (callback)
PWGEN_ENUM_LEVELS_CALLBACK (Page 2008)	List authorization levels (callback)

3.5 Functions of the user administration (RT Professional)

PWGEN_ENUM_USERS_CALLBACK (Page 1997)	List users (callback)
PWGENAddGroup (Page 1998)	Create user group
PWGENAddPermLevel (Page 2002)	Create authorization level
PWGENAddUser (Page 1987)	Create user
PWGENAddUserEx (Page 1989)	Create user
PWGENChangePassword (Page 1991)	Change password
PWGENCheckPermission (Page 2004)	Verify user authorizations of a user
PWGENCheckUser (Page 1992)	Check user
PWGENConnect (Page 1984)	Establish connection to the database.
PWGENDeletePermLevel (Page 2005)	Delete authorization level
PWGENDeleteUser (Page 1994)	Delete user/delete user group
PWGENDisconnect (Page 1986)	Terminate connection to the database
PWGENEnumGroups (Page 2000)	List user groups
PWGENEnumPermLevels (Page 2006)	List authorization levels
PWGENEnumUsers (Page 1995)	List users
PWGENReadUserPerm (Page 2009)	Determine user authorizations
PWRTCheckPermission (Page 2010)	Check area authorization
PWRTCheckPermissionOnPicture (Page 2011)	Check authorization level
PWRTGetCurrentUser (Page 2016)	Determine login name
PWRTGetLoginPriority (Page 2017)	GetLoginPriority
PWRTIsLoggedInByCard (Page 2018)	Login via chip card
PWRTLogin (Page 2019)	Login via dialog
PWRTLogout (Page 2020)	Logout
PWRTLogoutEx (Page 2021)	Logout
PWRTPermissionLevelDialog (Page 2012)	Select authorization level
PWRTPermissionLevelDialogEx (Page 2013)	Select authorization level
PWRTPermissionToString (Page 2014)	Determine authorization level for text description
PWRTSilentLogin (Page 2022)	Login
PWRTSilentLoginEx (Page 2023)	Login

3.5.1.2 Overview of structures (RT Professional)

Overview

PWGEN_GROUPINFO (Page 1982)	User group information
PWGEN_LEVELINFO (Page 1983)	Authorization level information
PWGEN_USERINFO (Page 1984)	User information

3.5.1.3 Error messages (RT Professional)

Overview

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

Error messages for RT:

PWRT_CAP_SHMERROR	1	Shared memory error
PWRT_CAP_NOLOGIN	2	No user is logged on
PWRT_CAP_NOPERM	3	User does not have authorization.
PWRT_CAP_NOAREA	4	The area could not be found.
PWRT_GPA_SHMERROR	1	Shared memory error
PWRT_GPA_NOLOGIN	2	No user is logged on

Error messages for CS V5.0 and higher:

PWGEN_API_ERR_SUPPLY	0x10000001L	An unspecified error has occurred.
PWGEN_API_NO_MEMORY	0x10000001L	Not enough memory.
PWGEN_API_NOT_SUPPORTED	0x10000003L	This function is not supported
PWGEN_API_INVALID_PARAM	0x10000004L	A function parameter is incorrect or missing.
PWGEN_API_NO_INTERFACE	0x10000005L	No connection to the UserAdminASO-Interface.
PWGEN_API_I_ENUM	0x10000006L	Error while enumerating
PWGEN_API_I_READ	0x10000007L	Error while reading
PWGEN_API_I_CREATE	0x10000008L	Error while creating new data
PWGEN_API_I_MODIFY	0x10000009L	Error while modifying data
PWGEN_API_I_DELETE	0x1000000AL	Error while deleting data
PWGEN_API_TERMINATED	0x1000000FL	Processing was cancelled prematurely.
PWGEN_API_NO_CONNECTION	0x10000010L	PWGENConnect not yet executed.
PWGEN_API_IS_CONNECTED	0x10000011L	PWGENConnect has already been executed.
PWGEN_API_NO_USER	0x10000020L	User not found
PWGEN_API_MAX_USER	0x10000021L	The maximum number of users has been reached (~ 128).
PWGEN_API_EXIST_USER	0x10000022L	The selected user already exists.
PWGEN_API_NO_GROUP	0x10000030L	User group was not found.
PWGEN_API_MAX_GROUP	0x10000031L	The maximum number of user groups has been reached (~ 10).
PWGEN_API_EXIST_GROUP	0x10000032L	The selected user group already exists.
PWGEN_API_NO_LEVEL	0x10000040L	Authorization level not found.
PWGEN_API_MAX_LEVEL	0x10000041L	The maximum number of authorization levels has been reached (~ 1000).

3.5 Functions of the user administration (RT Professional)

PWGEN_API_EXIST_LEVEL	0x10000042L	The selected authorization level already exists.
PWGEN_API_NO_PERM	0x10000050L	No authorization assigned.
PWGEN_API_MAX_PERM	0x10000051L	
PWGEN_API_EXIST_PERM	0x10000052L	The authorization already exists.
PWGEN_API_NO_AREA	0x10000060L	No range was assigned.
PWGEN_API_MAX_AREA	0x10000061L	The maximum number of areas has been reached (~ 32).
PWGEN_API_EXIST_AREA	0x10000062L	The area already exists.
PWGEN_API_NO_PASSWORD	0x10000070L	No password, or password is invalid.
PWGEN_API_MAX_PASSWORD	0x10000071L	
PWGEN_API_EXIST_PASSWORD	0x10000072L	

3.5.1.4 Constants (RT Professional)

String and array sizes

MAX_LOGIN	25
MAX_PASS	25
MAX_LEVEL	70

3.5.2 Structures (RT Professional)

3.5.2.1 PWGEN_GROUPINFO (RT Professional)

Declaration

```
typedef struct {
    TCHAR    name[MAX_LOGIN];
    int      expiration_time;
}
PWGEN_GROUPINFO;
```

Members

name

Name of the user group

expiration_time

automatic logout time in minutes

Required files

usegenap.h

API functions

PWGEN_ENUM_GROUPS_CALLBACK (Page 2001)	List user groups

See also

PWGEN_ENUM_GROUPS_CALLBACK (Page 2001)

3.5.2.2 PWGEN_LEVELINFO (RT Professional)**Declaration**

```
typedef struct {
    int      levelNumber;
    DWORD    dwTextID;
}
PWGEN_LEVELINFO;
```

Members**levelNumber**

Number of the authorization

dwTextID

ID which identifies the description of authorization.

Required files

usegenap.h

API functions

PWGEN_ENUM_LEVELS_CALLBACK (Page 2008)	List authorization levels

See also

PWGEN_ENUM_LEVELS_CALLBACK (Page 2008)

3.5.2.3 PWGEN_USERINFO (RT Professional)

Declaration

```
typedef struct {
    TCHAR    login[MAX_LOGIN];
    TCHAR    group[MAX_LOGIN];
    int      expiration_time;
}
PWGEN_USERINFO;
```

Members

- login**
Logon name of the user
- group**
Name of the group to which the user belongs
- expiration_time**
automatic logout time in minutes

Required files

usegenap.h

API functions

PWGEN_ENUM_USERS_CALLBACK (Page 1997)	List users

See also

PWGEN_ENUM_USERS_CALLBACK (Page 1997)

3.5.3 General functions (RT Professional)

3.5.3.1 PWGENConnect (RT Professional)

Description

The function connects to the database of the currently open project.

Declaration

```

BOOL PWGENConnect (
    LPCTSTR      DSNName,
    LPCMN_ERROR  err)

```

Parameters

DSNName

DataSource name of the project. The DSNName parameter is not evaluated. An empty string can be passed but not ZERO. The database of the currently open project is always used.

err

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection established

FALSE

Error

Comment

Each application can perform only one Connect. With a repeated Connect, the PWGEN_API_IS_CONNECTED error message is returned.

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_IS_CONNECTED	PWGENConnect has already been executed.

Required files

usegenap.h

usegen.lib

usegen.dll

Related functions

PWGENDisconnect (Page 1986)	Terminate connection to the database

See also

PWGENDisconnect (Page 1986)

Overview of the functions (Page 1977)

3.5.3.2 PWGENDisconnect (RT Professional)**Description**

The connection to the database of the currently open project is terminated.

Declaration

```
BOOL PWGENDisconnect (  
    LPCMN_ERROR    err)
```

Parameters**err**

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Connection terminated

FALSE

Error

Comment

A PWGENDisconnect before the application is closed, otherwise the internally used UserAdminASO interfaces will no longer be enabled.

Note

The call may not be used in the destructor of an application (EXE, DLL, OCX, etc.). Due to Microsoft-specific mechanisms, this may cause the call to freeze and thereby crash the program.

Error messages

PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.

Required files

usegenap.h

usegen.lib

usegen.dll

Related functions

PWGENConnect (Page 1982)	Make connection to the database.

See also

PWGENConnect (Page 1982)

Overview of the functions (Page 1977)

3.5.4 Functions for processing users (RT Professional)**3.5.4.1 PWGENAddUser (RT Professional)****Description**

The function creates a new user, including an empty user authorization matrix. You can use the PWGENAddUserEx function to transfer the authorizations of the group to the users to be created.

Declaration

```

BOOL PWGENAddUser (
    LPCTSTR      username,
    LPCTSTR      password,
    LPCTSTR      group,
    int          expiration_time,
    LPCMN_ERROR  error )

```

Parameters

username

Logon name of the user

password

Password of the user

group

Group to which the user should be added

expiration_time

Automatic logout time in minutes. If the selected expiration_time is outside the permitted range, it is set to 0.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Users created

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_NO_MEMORY	Not enough memory.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_EXIST_USER	The selected user already exists.

PWGEN_API_MAX_USER	The maximum number of users has been reached (~ 128).
PWGEN_API_NO_GROUP	Cannot find user group.
PWGEN_API_I_CREATE	Error creating new data.

Required files

usegenap.h
 usegen.lib
 usegen.dll

Related functions

PWGENAddUserEx (Page 1989)	Creating users

See also

PWGENAddUserEx (Page 1989)
 Overview of the functions (Page 1977)

3.5.4.2 PWGENAddUserEx (RT Professional)**Description**

The function creates a new user. Depending on the copy_group_protection parameter, you can transfer the authorizations of the group to the users to be created.

Declaration

```

BOOL PWGENAddUserEx (
    LPCTSTR    username,
    LPCTSTR    password,
    LPCTSTR    group,
    int        expiration_time,
    BOOL       copy_group_permissions,
    LPCMN_ERROR error )

```

Parameters**username**

Logon name of the user

password

Password of the user

group

Group to which the user should be added

expiration_time

Automatic logout time in minutes. If the selected expiration_time is outside the permitted range, it is set to 0.

copy_group_permissions

If copy_group_permission = TRUE, the user authorizations of the group to the newly created user will be transferred.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Users created

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_NO_MEMORY	Not enough memory.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_EXIST_USER	The selected user already exists.
PWGEN_API_MAX_USER	The maximum number of users has been reached (~ 128).
PWGEN_API_NO_GROUP	Cannot find user group.
PWGEN_API_I_CREATE	Error creating new data.

Required files

usegenap.h

usegen.lib

usegen.dll

Related functions

PWGENAddUser (Page 1985)	Creating users

See also

PWGENAddUser (Page 1985)

Overview of the functions (Page 1977)

3.5.4.3 PWGENChangePassword (RT Professional)**Description**

This function can be used to change the password of the user specified by the user name.

Declaration

```

BOOL PWGENChangePassword (
    LPCTSTR      username,
    LPCTSTR      oldpassword,
    LPCTSTR      newpassword,
    LPCMN_ERROR  error )

```

Parameters**username**

Logon name of the user

oldpassword

Old password of the user

newpassword

New password of the user

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Password changed

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_NO_PASSWORD	No or no valid password.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_NO_MEMORY	Not enough memory.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_USER	Cannot find user.
PWGEN_API_I_READ	Error reading.
PWGEN_API_I_MODIFY	Error changing data.

Required files

usegenap.h

usegen.lib

usegen.dll

See also

Overview of the functions (Page 1977)

3.5.4.4 PWGENCheckUser (RT Professional)**Description**

This checks whether the specified user exists in the current WinCC project and whether the given password is correct.

Declaration

```

BOOL PWGENCheckUser (
    LPCTSTR      username,
    LPCTSTR      password,
    LPCMN_ERROR  error )

```

Parameters

username

Name of the user.

password

The password that belongs to the specified user.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

User exists and the password is correct.

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_USER	Cannot find user.
PWGEN_API_NO_PASSWORD	No or no valid password.

Required files

usegenap.h

usegen.lib

usegen.dll

See also

Overview of the functions (Page 1977)

3.5.4.5 PWGENDeleteUser (RT Professional)

Description

The specified user or user group is deleted, including the authorization data.

Declaration

```
BOOL PWGENDeleteUser (
    LPCTSTR      username,
    BOOL         is_user,
    LPCMN_ERROR  error )
```

Parameters

username

Number of the user to be deleted.

is_user

The is_user parameter is used to distinguish whether a user or a user group is to be deleted.

TRUE	A user should be deleted.
FALSE	A user group should be deleted.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

User or user groups deleted

FALSE

Error

Comment

If you attempt to delete the administrator, the function returns the PWGEN_API_EXIST_USER error.

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_I_DELETE	Error deleting data.
PWGEN_API_NO_USER	Cannot find user.
PWGEN_API_NO_GROUP	Cannot find user group.
PWGEN_API_EXIST_USER	The selected user already exists.
PWGEN_API_NO_MEMORY	Not enough memory.

Required files

usegenap.h
 usegen.lib
 usegen.dll

See also

Overview of the functions (Page 1977)

3.5.4.6 PWGENEnumUsers (RT Professional)

Description

The function reads the configured users, calls the callback function for each user and returns the number of users in dwCount.

Declaration

```

BOOL PWGENEnumUsers (
    LPDWORD                dwCount,
    PWGEN_ENUM_USERS_CALLBACK cfn,
    PVOID                  userdata,
    LPCMN_ERROR            error )

```

Parameters

dwCount

Pointer to the location where the number of users should be stored.

cfn

Your callback function that receives the information. If cfn == NULL, only users are counted.

userdata

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Users listed

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_MEMORY	Not enough memory.

Required files

usegenap.h

usegen.lib

usegen.dll

Related functions

PWGEN_ENUM_USERS_CALLBACK (Page 1997)	List users (callback)

See also

PWGEN_ENUM_USERS_CALLBACK (Page 1997)

Overview of the functions (Page 1977)

3.5.4.7 PWGEN_ENUM_USERS_CALLBACK (RT Professional)

Description

In order to evaluate the user information listed by the system, you must provide a PWGEN_ENUM_USERS_CALLBACK type callback function.

Declaration

```
BOOL ( * PWGEN_ENUM_USERS_CALLBACK) (
    LPWGEN_USERINFO    lpUserInfo,
    PVOID               lpUser);
```

Parameters

lpUserInfo

Pointer to a PWGEN_USERINFO (Page 1982) type structure with the data of a user.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

Users listed

FALSE

Error

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

usegenap.h

Related functions

PWGENEnumUsers (Page 1993)	List users

See also

PWGENEnumUsers (Page 1993)

PWGEN_USERINFO (Page 1982)

Overview of the functions (Page 1977)

3.5.5 Functions for processing user groups (RT Professional)**3.5.5.1 PWGENAddGroup (RT Professional)****Description**

The function creates a new user group, including an empty user authorization matrix.

Declaration

```
BOOL PWGENAddGroup (
    LPCTSTR      username,
    int           expiration_time,
    LPCMN_ERROR   error )
```

Parameters**username**

Name of the group to be created

expiration_time

Automatic logout time. If the selected expiration_time is outside the permitted range, it is set to 0.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

User group created

FALSE

User group already exists

Comment

Use the PWGENDeleteUser function to delete an existing user group.

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_EXIST_GROUP	The selected user group already exists.
PWGEN_API_MAX_GROUP	The maximum number of user groups has been reached (~ 10).
PWGEN_API_NO_MEMORY	Not enough memory.
PWGEN_API_I_CREATE	Error creating new data.

Required files

usegenap.h

usegen.lib

usegen.dll

Related functions

PWGENDeleteUser (Page 1992)	Creating users

See also

PWGENDeleteUser (Page 1992)

Overview of the functions (Page 1977)

3.5.5.2 PWGENEnumGroups (RT Professional)

Description

The function reads the configured user groups, calls the callback function for each group and returns the number of groups in dwCount.

Declaration

```
BOOL PWGENEnumGroups (
    LPDWORD                dwCount,
    PWGEN_ENUM_GROUPS_CALLBACK cfn,
    PVOID                  userdata,
    LPCMN_ERROR             error )
```

Parameters

dwCount

Pointer to the location where the number of user groups should be stored.

cfn

Your callback function that receives the information. If cfn == NULL, only user groups are counted.

userdata

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

User groups listed

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.

PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_MEMORY	Not enough memory.

Required files

usegenap.h
 usegen.lib
 usegen.dll

Related functions

PWGEN_ENUM_GROUPS_CALLBACK (Page 2001)	List user groups (callback)

See also

PWGEN_ENUM_GROUPS_CALLBACK (Page 2001)
 Overview of the functions (Page 1977)

3.5.5.3 PWGEN_ENUM_GROUPS_CALLBACK (RT Professional)**Description**

In order to evaluate the information of user groups listed by the system, you must provide a PWGEN_ENUM_GROUPS_CALLBACK type callback function.

Declaration

```

BOOL ( * PWGEN_ENUM_GROUPS_CALLBACK) (
    LPWGEN_GROUPINFO    lpGroupInfo,
    PVOID                lpUser);

```

Parameters**lpUserInfo**

Pointer to a PWGEN_GROUPINFO (Page 1980) type structure with the data of a user group.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

User groups listed

FALSE

Error

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

usegenap.h

Related functions

PWGENEnumGroups (Page 1998)	List user groups

See also

PWGENEnumGroups (Page 1998)

PWGEN_GROUPINFO (Page 1980)

Overview of the functions (Page 1977)

3.5.6 Functions for processing authorizations (RT Professional)

3.5.6.1 PWGENAddPermLevel (RT Professional)

Description

Creates a new authorization for all users.

Declaration

```

BOOL PWGENAddPermLevel (
    DWORD          txtID,
    int             number,
    LPCMN_ERROR     error )

```

Parameters

txtID

Text ID of the text for the authorization.

number

Number of the authorization to be created

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Authorization level created

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_MEMORY	Not enough memory.
PWGEN_API_MAX_LEVEL	The maximum number of authorization levels has been reached (1000).
PWGEN_API_EXIST_LEVEL	The selected authorization level already exists.
PWGEN_API_I_CREATE	Error creating new data.

Required files

usegenap.h

usegen.lib

usegen.dll

See also

Overview of the functions (Page 1977)

3.5.6.2 PWGENCheckPermission (RT Professional)

Description

The function checks whether the user has the operator authorizations for the specified authorization level.

Declaration

```
BOOL PWGENCheckPermission(  
    LPCTSTR      username,  
    DWORD        permlevel,  
    LPCMN_ERROR  error)
```

Parameters

username

Name of the user

permlevel

Number of the authorization to be checked

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

The operator authorization for the authorization level of the user is available.

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_MEMORY	Not enough memory.
PWGEN_API_NO_LEVEL	Cannot find authorization level.
PWGEN_API_I_READ	Error reading.
PWGEN_API_NO_PERM	No authorization granted.

Required files

usegenap.h

usegen.lib

usegen.dll

See also

Overview of the functions (Page 1977)

3.5.6.3 PWGENDeletePermLevel (RT Professional)

Description

Deletes the authorization level specified by levelNumber from the user authorization matrix.

Declaration

```

BOOL PWGENDeletePermLevel (
    int          levelNumber,
    LPCMN_ERROR  error )

```

Parameters

levelNumber

Number of the authorization to be deleted

error

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Authorization level deleted

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_MEMORY	Not enough memory.
PWGEN_API_NO_LEVEL	Cannot find authorization level.
PWGEN_API_I_DELETE	Error deleting data.

Required files

usegenap.h

usegen.lib

usegen.dll

See also

Overview of the functions (Page 1977)

3.5.6.4 PWGENEnumPermLevels (RT Professional)**Description**

The function reads the configured authorization levels, calls the callback function for each level and returns the number of authorization levels in dwCount.

Declaration

```

BOOL PWGENEnumPermLevels (
    LPDWORD                dwCount,
    PWGEN_ENUM_LEVELS_CALLBACK cfn,
    PVOID                  userdata,
    LPCMN_ERROR             error )

```

Parameters

dwCount

Pointer to the location where the number of authorization levels should be stored.

cfm

Your callback function that receives the information. If `cfm == NULL`, only authorizations are counted.

userdata

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

error

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Authorization levels listed

FALSE

Error

Error messages

PWGEN_API_INVALID_PARAM	A function parameter is incorrect or missing.
PWGEN_API_ERR_SUPPLY	An unspecified error has occurred.
PWGEN_API_NO_INTERFACE	No connection to the UserAdminASO interface.
PWGEN_API_NO_CONNECTION	No PWGENConnect has been executed.
PWGEN_API_I_ENUM	Error in enumeration.
PWGEN_API_NO_MEMORY	Not enough memory.

Required files

usegenap.h

usegen.lib

usegen.dll

Related functions

PWGEN_ENUM_LEVELS_CALLBACK (Page 2008)	List authorizations (callback)

See also

PWGEN_ENUM_LEVELS_CALLBACK (Page 2008)

Overview of the functions (Page 1977)

3.5.6.5 PWGEN_ENUM_LEVELS_CALLBACK (RT Professional)

Description

In order to evaluate the authorization levels listed by the system, you must provide a PWGEN_ENUM_LEVELS_CALLBACK type callback function.

Declaration

```
BOOL ( * PWGEN_ENUM_LEVELS_CALLBACK) (
    LPWGEN_LEVELINFO    lpLevelInfo,
    PVOID                lpUser);
```

Parameters

lpLevelInfo

Pointer to a PWGEN_LEVELINFO (Page 1981) type structure with the data of the authorization level.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

Authorization levels listed

FALSE

Error

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

usegenap.h

Related functions

PWGENEnumPermLevels (Page 2004)	List authorization levels

See also

PWGENEnumPermLevels (Page 2004)

PWGEN_LEVELINFO (Page 1981)

Overview of the functions (Page 1977)

3.5.6.6 PWGENReadUserPerm (RT Professional)**Description**

You can use this function to determine the area for which a user has operator authorizations on a certain authorization level.

Declaration

```

BOOL PWGENReadUserPerm(
    LPCTSTR    username,
    BOOL       is_user,
    int        levelnumber,
    LPBYTE     freigabe,
    LPDWORD    areaperms )

```

Parameters**username**

Name of the user or user group

is_user

is_user indicates whether username refers to a user or a user group.

TRUE	User
FALSE	User group

levelnumber

Number of the authorization to be read

3.5 Functions of the user administration (RT Professional)

freigabe

If freigabe = 1, the user has operator authorization for all areas on this authorization level.

areaperms

The areas are characterized with the 32 bits of areaperm. If a user has authorization for an area, the corresponding bit is set. The least significant bit corresponds to the first area.

Return value

TRUE

Operator authorizations determined

FALSE

Error

Required files

usegenap.h

usegen.lib

usegen.dll

See also

Overview of the functions (Page 1977)

3.5.6.7 PWRTCheckPermission (RT Professional)

Description

Checks whether the logged-on user has operator authorization for the given authorization.

Declaration

```
BOOL PWRTCheckPermission (  
    DWORD    permlevel,  
    DWORD    suppress_messagebox )
```

Parameters

permlevel

Number of the authorization level to be checked

suppress_messagebox

No dialog is displayed with suppress_messagebox != 0.

Return value**TRUE**

Authorization granted

FALSE

Authorization not granted

Required files

pwr_api.h

pass_s.lib

useadmin.dll

Examples

PWRT check permission (Page 2026) "PWRTBunch.cpp"

See also

PWRT check permission (Page 2026)

Overview of the functions (Page 1977)

3.5.6.8 PWRTCheckPermissionOnPicture (RT Professional)**Description**

Checks whether the logged-on user has operator authorization for the given authorization.

Declaration

```
BOOL PWRTCheckPermissionOnPicture (
    DWORD          permlevel,
    LPCTSTR        picture_name,
    DWORD          suppress_messagebox,
    LPCMN_ERROR    lperr )
```

Parameters

permlevel

Number of the authorization to be checked.

picture_name

Name of the screen that contains the object to be tested.

suppress_messagebox

No dialog is displayed with `suppress_messagebox != 0`.

lperr

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Authorization granted

FALSE

Authorization not granted

Required files

`pwrtd_api.h`

`pass_s.lib`

`useadmin.dll`

Examples

Checks admission of a certain level for a screen (Page 2027) "PWRTBunch.cpp"

See also

Checks admission of a certain level for a screen (Page 2027)

Overview of the functions (Page 1977)

3.5.6.9 PWRTPermissionLevelDialog (RT Professional)

Description

You should open the dialog for selecting authorizations using the `PWRTPermissionLevelDialogEx` function, because then a redraw is performed when you move the dialog window.

Declaration

```
LONG PWRTPermissionLevelDialog (
)
```

Parameters

None

Related functions

PWRTPermissionLevelDialogEx (Page 2013)	Select authorization for via dialog

Examples

Authorization level query through a dialog with specifying a possible error (Page 2029)
 "PWRTBunch.cpp"

See also

Authorization level query through a dialog with specifying a possible error (Page 2029)
 PWRTPermissionLevelDialogEx (Page 2013)
 PWRTPermissionToString (Page 2014)
 Overview of the functions (Page 1977)

3.5.6.10 PWRTPermissionLevelDialogEx (RT Professional)**Description**

Protecting the usability of objects made in WinCC through dialogue to select a rating. PWRTPermissionLevelDialogEx displays a corresponding dialog that offers the available authorizations for selection.

Declaration

```
LONG PWRTPermissionLevelDialogEx (
    HWND      hParentWnd,
    CMN_ERROR *lpErr)
```

Parameters

hParentWnd

Handle to the parent of the dialog box.

lpErr

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

0-999 Authorization level

-1: Dialog closed with Cancel.

Required files

pwr_api.h

pass_s.lib

useadmin.dll

Example

Authorization level query through a dialog with specifying a possible error (Page 2029)
"PWRTBunch.cpp"

See also

PWRTPermissionLevelDialog (Page 2010)

Authorization level query through a dialog with specifying a possible error (Page 2029)

Overview of the functions (Page 1977)

3.5.6.11 PWRTPermissionToString (RT Professional)

Description

Determines the description associated with an authorization level.

Declaration

```
BOOL PWRTPermissionToString (  
    LONG      perm,  
    LPTSTR    string,  
    int       bufsize)
```

Parameters

perm

Authorization level whose description is to be determined.

string

Pointer to the buffer for recording the description.

bufsize

Size of the buffer

Return value

TRUE

Description successfully transferred.

FALSE

Description could not be determined.

Required files

pwrt_api.h

pass_s.lib

useadmin.dll

Related functions

PWRTPermissionLevelDialog (Page 2010)	Select authorization level

Examples

Gets a string associated with the permission number (Page 2028). "PWRTBunch.cpp"

See also

PWRTPermissionLevelDialog (Page 2010)

Gets a string associated with the permission number (Page 2028)

Overview of the functions (Page 1977)

3.5.7 Functions for logon, logoff (RT Professional)

3.5.7.1 PWRTGetCurrentUser (RT Professional)

Description

Determines the user name of the current logged-on user.

Declaration

```
BOOL PWRTGetCurrentUser (
    LPTSTR    dest,
    int       bufsize )
```

Parameters

dest

Data buffer for receiving the user name

bufsize

Size of the data buffer in bytes

Return value

TRUE

User name successfully transferred

FALSE

No user is logged on

Required files

pwrt_api.h

pass_s.lib

useadmin.dll

Examples

Return the name of the current user (Page 2030) "PWRTBunch.cpp"

See also

Return the name of the current user (Page 2030)

Overview of the functions (Page 1977)

3.5.7.2 PWRTGetLoginPriority (RT Professional)**Description**

Logon check without dialog.

Declaration

```
LONG PWRTGetLoginPriority (  
    )
```

Parameters

None

Return value

Priority level:

LOGIN_STANDARD	(value: 0)
LOGIN_CARD	(value: 1)
LOGIN_KEYSWITCH	(value: 2)

Required files

pwrt_api.h

pass_s.lib

useadmin.dll

Examples

Queries the current logon priority (Page 2031) "PWRTBunch.cpp"

See also

Queries the current logon priority (Page 2031)

Overview of the functions (Page 1977)

3.5.7.3 PWRTIsLoggedInByCard (RT Professional)

Description

Determines whether the user is logged on via smart card.

Declaration

```
BOOL PWRTIsLoggedInByCard (
    )
```

Parameters

None

Return value

TRUE

User logged on with smart card

FALSE

Users not logged on with smart card or not logged on

Required files

pwrt_api.h

pass_s.lib

useadmin.dll

Examples

Checks if the user has been logged on by card (Page 2031) "PWRTBunch.cpp"

See also

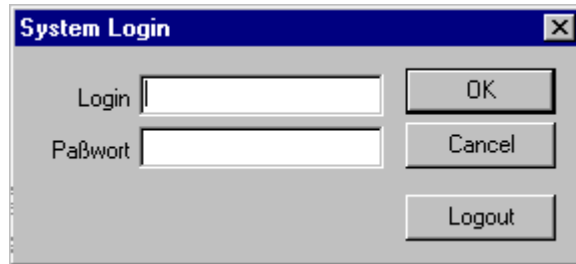
Checks if the user has been logged on by card (Page 2031)

Overview of the functions (Page 1977)

3.5.7.4 PWRTLogin (RT Professional)

Description

Displays the logon dialog and loads the user data into the shared memory when logon is successful.



Declaration

```
BOOL PWRTLogin (  
    TCHAR    monitor )
```

Parameters

monitor

Monitor on which the dialog is shown. The value is not numeric; it is specified in TCHAR format, i.e. '1' for Monitor 1.

Return value

TRUE

Dialog open. The function does not wait for the dialog to close.

FALSE

Dialog not open

Comment

In addition to manually entering the logon data, you can also log on with a smart card. The card reader is connected directly to the OS. If no valid card is connected when the function is executed, the logon dialog automatically disappears.

Required files

pwrt_api.h

pass_s.lib
useadmin.dll

Related functions

PWRTSilentLogin (Page 2022)	Logon

Examples

PWRT login - dialog provided by WinCC itself (Page 2033) "PWRTBunch.cpp"

See also

PWRTSilentLogin (Page 2022)
PWRT login - dialog provided by WinCC itself (Page 2033)
PWRTSilentLoginEx (Page 2023)
Overview of the functions (Page 1977)

3.5.7.5 PWRTLogout (RT Professional)

Description

The function causes a flag to be set at logoff in the shared memory.

Declaration

```
BOOL PWRTLogout (  
    )
```

Parameters

None

Return value

TRUE

Logoff successfully passed to PassDBRT.

FALSE

Logoff rejected (for example, PassDBRT not available, ServiceMode, etc.).

Required files

pwrt_api.h
pass_s.lib
useadmin.dll

Related functions

PWRTLogoutEx (Page 2021)	Log off user with priority level

Examples

PWRT logoff (Page 2034) "PWRTBunch.cpp"

See also

PWRT logoff (Page 2034)
PWRTLogoutEx (Page 2021)
Overview of the functions (Page 1977)

3.5.7.6 PWRTLogoutEx (RT Professional)**Description**

The function causes a flag to be set at logoff in the shared memory, unless the specified priority level is too low.

Declaration

```
BOOL PWRTLogoutEx (  
    int    nLevel  
)
```

Parameters

nLevel
Priority level of the user.

Return value

TRUE
Logoff successfully passed to PassDBRT.

FALSE

Logoff rejected (for example, PassDBRT not available, ServiceMode, etc.).

Required files

pwrt_api.h
pass_s.lib
useadmin.dll

Related functions

PWRTLogout (Page 2018)	User logoff

Examples

Silent logoff with priority level (Page 2035) "PWRTBunch.cpp"

See also

Silent logoff with priority level (Page 2035)
PWRTLogout (Page 2018)
Overview of the functions (Page 1977)

3.5.7.7 PWRTSilentLogin (RT Professional)**Description**

In contrast to PWRTLogin, logon is not performed in a dialog. The data for the user, user name and password are passed directly to the function.

Declaration

```
BOOL PWRTSilentLogin (  
    LPCTSTR    login,  
    LPCTSTR    password )
```

Parameters

login
Name of the user.

password

Password of the user

Return value**TRUE**

Logon successful

FALSE

Logon rejected

Comment

If no valid smart card is connected when the function is executed, the logon is rejected.

Required files

pwrt_api.h

pass_s.lib

useadmin.dll

Related functions

PWRTLogin (Page 2017)	Logon via dialog
PWRTSilentLoginEx (Page 2023)	Logon with priority level

Examples

Logon without using a dialog (Page 2036) "PWRTBunch.cpp"

See also

PWRTLogin (Page 2017)

PWRTSilentLoginEx (Page 2023)

Logon without using a dialog (Page 2036)

Overview of the functions (Page 1977)

3.5.7.8 PWRTSilentLoginEx (RT Professional)**Description**

In contrast to PWRTLogin, logon is not performed in a dialog. The data for the user (user name, password and priority level) are passed directly to the function.

Declaration

```
BOOL PWRTSilentLoginEx (  
    LPCTSTR    login,  
    LPCTSTR    password,  
    int        nLevel)
```

Parameters

login

Name of the user.

password

Password of the user

nLevel

Priority level of the user

Return value

TRUE

Logon successful

FALSE

Logon rejected

Comment

If no valid smart card is connected when the function is executed, the logon is rejected.

Required files

pwr_api.h

pass_s.lib

useadmin.dll

Related functions

PWRTLogin (Page 2017)	Logon via dialog
PWRTSilentLogin (Page 2020)	Logon without dialog

Examples

Silent logon with priority level (Page 2037) "PWRTBunch.cpp"

See also

[PWRTSilentLogin \(Page 2020\)](#)

[Silent logon with priority level \(Page 2037\)](#)

[PWRTLogin \(Page 2017\)](#)

[Overview of the functions \(Page 1977\)](#)

3.5.8 Examples (RT Professional)

3.5.8.1 PWRT check permission (RT Professional)

Example

```
//{{ODK_EXAMPLE}PWRT check permission. (USE)}
//{{FUNCTION}PWRTCheckPermission (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtcheckpermission()
{
    BOOL bRet;
    CString csOut;
    CString csPermLevName;
    CGetText l_PermissionLevel("Insert the permission level number:");
    if(l_PermissionLevel.DoModal()==IDOK)
    {
        //////////////////////////////////////
        //
        bRet = PWRTCheckPermission( l_PermissionLevel.m_lNumber, 0L ); // 0 = show the
message box
        //////////////////////////////////////
        //
        //////////////////////////////////////
        //
        bRet &= PWRTPermissionToString(l_PermissionLevel.m_lNumber,
csPermLevName.GetBuffer(1024), 1024);
        //////////////////////////////////////
        //
        if(!bRet)
        {
            m_pView->Print("ERROR: ", FSIZE_FUNCMARK);
            m_pView->Print("PWRTCheckPermission.\n", FSIZE_PARAMMARK, FALSE, TRUE);
            csOut.Format("Level = %ld (\"%s\") - Access denied.\n",
l_PermissionLevel.m_lNumber,
csPermLevName);
            m_pView->Print(csOut, FSIZE_SUBMARK);
            m_pView->Print("\n");
            return;
        }
        else
        {
            csOut.Format("Level = %ld (\"%s\")\n",
l_PermissionLevel.m_lNumber,
csPermLevName);
            m_pView->Print("PWRTCheckPermission\n", FSIZE_FUNCMARK, TRUE);
            m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
            m_pView->Print(csOut, FSIZE_SUBMARK);
            m_pView->Print("\n");
        }
    }
}
```



```
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTCheckPermission (Page 2008)

3.5.8.2 Checks admission of a certain level for a screen (RT Professional)

Example

```
//{{ODK_EXAMPLE}Checks admission of a certain level for a picture. (USE)}
//{{FUNCTION}PWRTCheckPermissionOnPicture (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtcheckpermissiononpicture()
{
    BOOL bRet;
    CMN_ERROR err;
    CString csOut;
    CGetText l_PermissionLevel("Enter the permission level:");
    CGetText l_PictureName("Enter the picture name:", FALSE);
    if(l_PermissionLevel.DoModal()==IDCANCEL)
    {
        return;
    }
    if(l_PictureName.DoModal()==IDCANCEL)
    {
        return;
    }
    //////////////////////////////////////
    //
    bRet=PWRTCheckPermissionOnPicture(l_PermissionLevel.m_lNumber, l_PictureName.m_csText,
    0, &err);
    //////////////////////////////////////
    //
    if(bRet)
    {
        csOut.Format("Picture: \"%s\"\nLevel = %ld\n",
        l_PictureName.m_csText,
        l_PermissionLevel.m_lNumber);
        m_pView->Print("PWRTCheckPermissionOnPicture\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
    else
    {
        m_pView->PrintError(&err, "PWRTCheckPermissionOnPicture");
        m_pView->PrintError("Access denied", "PWRTCheckPermissionOnPicture");
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTCheckPermissionOnPicture (Page 2009)

3.5.8.3 Gets a string associated with the permission number (RT Professional)**Overview**

```

//{{ODK_EXAMPLE}Gets a string associated with the permission number. (USE)}
//{{FUNCTION}PWRTPermissionToString (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtpermissiontostring()
{
    BOOL bRet;
    CString csOut;
    CString csPermLevName;
    CGetText l_PermissionLevel("Insert the permission level number:");
    if(l_PermissionLevel.DoModal()==IDOK)
    {
        //////////////////////////////////////////
        //
        bRet=PWRTPermissionToString(l_PermissionLevel.m_lNumber,
csPermLevName.GetBuffer(1024), 1024);
        //////////////////////////////////////////
        //
        if(!bRet)
        {
            m_pView->Print("ERROR: ", FSIZE_FUNCMARK);
            m_pView->Print("PWRTPermissionToString.\n", FSIZE_PARAMMARK, FALSE, TRUE);
            m_pView->Print("\n");
            return;
        }
        csOut.Format("PWRTPermissionToString( %ld, buffer )\n",
l_PermissionLevel.m_lNumber);
        m_pView->Print(csOut, FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Permission level name:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("buffer = \"%s\"\n", csPermLevName);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

PWRTPermissionToString (Page 2012)

3.5.8.4 Authorization level query through a dialog with specifying a possible error (RT Professional)

Example

```
//{{ODK_EXAMPLE}Permission level query through a dialog with specifying a possible error.
(USE)}
//{{FUNCTION}PWRTPermissionLevelDialogEx (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::PWRTxPermissionLevelDialogEx()
{
    long lLevel;
    CMN_ERROR err;
    CString csOut;
    //////////////////////////////////////
    //
    lLevel=PWRTPermissionLevelDialogEx(*m_pView, &err);
    //////////////////////////////////////
    //
    if(m_pView->PrintError(&err, "PWRTPermissionLevelDialogEx"))
    {
        m_pView->Print("lLevel = PWRTPermissionLevelDialogEx(...)\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Permission level number chosen:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("lLevel = %ld\n", lLevel);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTPermissionLevelDialog (Page 2010)

PWRTPermissionLevelDialogEx (Page 2011)

3.5.8.5 Return the name of the current user (RT Professional)**Example**

```

//{{ODK_EXAMPLE}Return the name of the current user. (USE)}
//{{FUNCTION}PWRTGetCurrentUser (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtgetcurrentuser()
{
    BOOL bRet;
    CString csUsername;
    CString csOut;
    //////////////////////////////////////
    //
    bRet=PWRTGetCurrentUser(csUsername.GetBuffer(1024), 1023);
    //////////////////////////////////////
    //
    if(!bRet)
    {
        m_pView->Print("ERROR: ", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("PWRTGetCurrentUser.\n", FSIZE_PARAMMARK, FALSE, TRUE);
        m_pView->Print("Failed to get current user's name.\n", FSIZE_SUBMARK);
        m_pView->Print("\n");
        return;
    }
    m_pView->Print("PWRTGetCurrentUser\n", FSIZE_FUNCMARK, TRUE);
    m_pView->Print("Current user's name:\n", FSIZE_PARAMMARK, FALSE, TRUE);
    csOut.Format("Name = \"%s\"\n", csUsername);
    m_pView->Print(csOut, FSIZE_SUBMARK);
    m_pView->Print("\n");
}
//{{ODK_EXAMPLE}(END)}

```

See also

PWRTGetCurrentUser (Page 2014)

3.5.8.6 Queries the current logon priority (RT Professional)

Example

```
//{{ODK_EXAMPLE}Queries the current login priority. (USE)}
//{{FUNCTION}PWRTGetLoginPriority (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtgetloginpriority()
{
    long lPriority;
    CString csOut;
    CMap<long, LONG, CString, CString> l_map(3);
    l_map[LOGIN_STANDARD]=CString("LOGIN_STANDARD");
    l_map[LOGIN_CARD]=CString("LOGIN_CARD");
    l_map[LOGIN_KEYSWITCH]=CString("LOGIN_KEYSWITCH");
    ///////////////////////////////////////////////////////////////////
//
    lPriority=PWRTGetLoginPriority();
    ///////////////////////////////////////////////////////////////////
//
    if(lPriority!=-1)
    {
        m_pView->Print("PWRTGetLoginPriority\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Priority returned:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("Priority level = %s\n", l_map[lPriority]);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTGetLoginPriority (Page 2015)

3.5.8.7 Checks if the user has been logged on by card (RT Professional)

Overview

3.5 Functions of the user administration (RT Professional)

```

//{{ODK_EXAMPLE}Checks for the user has been logged on by card. (USE)}
//{{FUNCTION}PWRTIsLoggedInByCard (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtisloggedinbycard()
{
    BOOL bRet;
    CString csOut;
    //////////////////////////////////////
    //
    bRet=PWRTIsLoggedInByCard();
    //////////////////////////////////////
    //
    if(bRet)
    {
        m_pView->Print("PWRTIsLoggedInByCard\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("An user show-up logged in by card.\n", FSIZE_PARAMMARK, FALSE, TRUE);
    }
    else
    {
        m_pView->PrintError(_T("Either no user logged-on-by-card or no user logged on at
all."),
        _T("PWRTIsLoggedInByCard()"));
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

PWRTIsLoggedInByCard (Page 2016)

3.5.8.8 PWRT login - dialog provided by WinCC itself (RT Professional)

Example

```
//{{ODK_EXAMPLE} PWRT login - dialog provided by WinCC intself. (USE)}
//{{FUNCTION}PWRTLogin (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::PWRTxLogin()
{
    BOOL bRet;
    ///////////////////////////////////////////////////////////////////
    //
    bRet=PWRTLogin(_T('1')); // it always appears on the first screen
    ///////////////////////////////////////////////////////////////////
    //
    if(!bRet)
    {
        m_pView->Print("ERROR: ", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("PWRTLogin.\n", FSIZE_PARAMMARK, FALSE, TRUE);
        m_pView->Print("General function failure.\n", FSIZE_SUBMARK);
        m_pView->Print("\n");
        return;
    }
    m_pView->Print("PWRTLogin\n", FSIZE_FUNCMARK, TRUE);
    m_pView->Print("Logging on...\n", FSIZE_PARAMMARK, FALSE, TRUE);
    m_pView->Print("\n");
}
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTLogin (Page 2017)

3.5.8.9 PWRT logoff (RT Professional)

Example

```
//{{ODK_EXAMPLE}PWRT logout. (USE)}  
//{{FUNCTION}PWRTLogout (USE)}  
//{{FUNCTION}(END)}  
void CPWRTBunch::Pwrtlogout()  
{  
    BOOL bRet;  
    ///////////////////////////////////////  
    //  
    bRet=PWRTLogout(); // it appears on the first screen  
    ///////////////////////////////////////  
    //  
    if(!bRet)  
    {  
        m_pView->Print("ERROR: ", FSIZE_FUNCMARK, TRUE);  
        m_pView->Print("PWRTLogout.\n", FSIZE_PARAMMARK, FALSE, TRUE);  
        m_pView->Print("General function failure.\n", FSIZE_SUBMARK);  
        m_pView->Print("\n");  
        return;  
    }  
    m_pView->Print("PWRTLogout\n", FSIZE_FUNCMARK, TRUE);  
    m_pView->Print("Logging out...\n", FSIZE_PARAMMARK, FALSE, TRUE);  
    m_pView->Print("\n");  
}  
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTLogout (Page 2018)

3.5.8.10 Silent logoff with priority level (RT Professional)

Example

```
//{{ODK_EXAMPLE}Silent logout with priority level. (USE)}
//{{FUNCTION}PWRTLogoutEx (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::PWRTxLogoutEx()
{
    BOOL bRet;
    CString csOut;
    CGetText l_PriorityLevel("The priority level:");
    if(l_PriorityLevel.DoModal()==IDCANCEL)
    {
        return;
    }
    ///////////////////////////////////////////////////////////////////
    // bRet=PWRTLogoutEx( l_PriorityLevel.m_lNumber);
    ///////////////////////////////////////////////////////////////////
    //
    if(!bRet)
    {
        m_pView->PrintError( _T("Cannot logout."),
            _T("PWRTLogoutEx"));
    }
    else
    {
        m_pView->Print("PWRTLogoutEx\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Logging out...\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("Priority level = %ld\n", l_PriorityLevel.m_lNumber);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTLogoutEx (Page 2019)

3.5.8.11 Logon without using a dialog (RT Professional)

Example

```
//{{ODK_EXAMPLE}Logs on without using a dialog. (USE)}
//{{FUNCTION}PWRTSilentLogin (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtsilentlogin()
{
    BOOL bRet;
    CPassdlg l_Password("Enter name and password:");
    if(l_Password.DoModal() == IDCANCEL)
    {
        return;
    }
    //////////////////////////////////////
    //
    bRet=PWRTSilentLogin(l_Password.m_csUsername,
    l_Password.m_csPassword);
    //////////////////////////////////////
    //
    if(!bRet)
    {
        m_pView->PrintError( _T("Access denied or no Runtime project open."),
        _T("PWRTSilentLogin"));
    }
    else
    {
        m_pView->Print("PWRTSilentLogin\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
    }
}
//{{ODK_EXAMPLE}(END)}
```

See also

PWRTSilentLogin (Page 2020)

3.5.8.12 Silent logon with priority level (RT Professional)

Example

```

//{{ODK_EXAMPLE}Silent login with priority level. (USE)}
//{{FUNCTION}PWRTSilentLoginEx (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtsilentloginex()
{
    BOOL bRet;
    CString csOut;
    CPassdlg l_Password("Enter name and password:");
    CGetText l_PriorityLevel("Enter priority level:");
    if(l_Password.DoModal()==IDCANCEL)
    {
        return;
    }
    if(l_PriorityLevel.DoModal()==IDCANCEL)
    {
        return;
    }
    //////////////////////////////////////
    //
    bRet=PWRTSilentLoginEx(l_Password.m_csUsername, l_Password.m_csPassword,
l_PriorityLevel.m_lNumber);
    //////////////////////////////////////
    //
    if(!bRet)
    {
        m_pView->PrintError( _T("Access denied or no Runtime project open."),
        _T("PWRTSilentLogin"));
    }
    else
    {
        m_pView->Print("PWRTSilentLogin\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("Priority level = %ld\n", l_PriorityLevel.m_lNumber);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

PWRTSilentLoginEx (Page 2021)

3.6 Functions of the text system (RT Professional)

3.6.1 Basics (RT Professional)

3.6.1.1 Overview of the functions (RT Professional)

Declaration

TXT_ENUM_INFOTEXTS_PROC (Page 2050)	List info texts (callback)
TXT_ENUM_LANGUAGES_PROC (Page 2057)	List configured languages (callback)
TXTCloseProject (Page 2041)	Close text library
TXTEnumInfoText (Page 2048)	List info texts
TXTEnumLanguages (Page 2056)	List configured languages
TXTGetFont (Page 2059)	Determine character set of a language
TXTGetMaxTextID (Page 2043)	Determine text ID (maximum)
TXTOpenProject (Page 2044)	Open text library
TXTRTConnect (Page 2045)	Keep connection to text server
TXTRTDisconnect (Page 2046)	Connection to the text server released
TXTRTGetInfoText (Page 2052)	Read info texts
TXTRTGetInfoTextMC (Page 2054)	Read info texts from text server
TXTRTGetLanguageID (Page 2062)	Determine activated language
TXTRTSetLanguage (Page 2063)	Language switch for error message texts
TXTShowLanguagesDialog (Page 2060)	Display languages (installed)
TXTUpdateRuntime (Page 2051)	Update info texts in runtime

3.6.1.2 Error messages (RT Professional)

Overview

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

TXT_SYS_ERROR	(value: 0x10000000)	If this bit is set in dwError1 of error structure CMN_ERROR, then dwError2 contains the system error code.
TXT_OK	(value: 0x00000000)	No error occurred.
TXT_NO_CONNECT	(value: 0x00000001)	The database is not yet open.
TXT_CONNECT	(value: 0x00000002)	Error while connecting to the database.
TXT_UPDATE	(value: 0x00000003)	Error while writing to the database.

3.6 Functions of the text system (RT Professional)

TXT_CREATE_KEY	(value: 0x00000004)	The TextID could not be created.
TXT_KEY_NOT_FOUND	(value: 0x00000005)	The specified TextID could not be found.
TXT_LANGUAGE_NOT_FOUND	(value: 0x00000006)	The specified language could not be found.
TXT_NO_NEW_RECORD	(value: 0x00000007)	A new record could not be created (AddRecord generated an error)
TXT_FONT_NOT_FOUND	(value: 0x00000008)	Font not found
TXT_TABLE_NOT_FOUND	(value: 0x00000009)	A database table was not found. The dwError2 field contains more detailed information:
TXTVERSIONINFO	(value: 0x00000001)	Version info table was not found.
TXTLANGUAGES	(value: 0x00000002)	Language table was not found.
TXTTABLE	(value: 0x00000003)	Text table was not found.
TXT_WRONG_PROJECT	(value: 0x00000010)	Wrong project name
TXT_OPEN	(value: 0x00000011)	Database could not be loaded.
TXT_PARAM	(value: 0x00000012)	Invalid parameter
TXT_DISCONNECT	(value: 0x00000013)	Error logging off of the database
TXT_CALLBACK	(value: 0x00000014)	Callback returned FALSE
TXT_ALREADY_FOUND	(value: 0x00000015)	Text already exists
TXT_CREATE	(value: 0x00000016)	Database tables could not be created.
TXT_UPDATE_RUNTIME	(value: 0x00000017)	Runtime not found, or SendMessage returned NULL.
TXT_ALREADY_EXIST	(value: 0x00000018)	Language already exists
TXT_NO_LANGUAGE	(value: 0x00000019)	Invalid language specified
TXT_TEXT_LIBRARY_RUNNING	(value: 0x00000020)	TEXTBIB.EXE already running; opening not allowed
TXT_TABLES_ALREADY_EXIST	(value: 0x00000021)	Database tables already exist. Creating no longer allowed.

Text Library RT

TXT_RT_OK	(value: 0x00000000)	No error
TXT_RT_OPEN_MMF	(value: 0x00000001)	Error while opening the memory mapped file
TXT_RT_CREATE_SERVICE_WINDOW	(value: 0x00000002)	Error while creating the service window
TXT_RT_KEY_NOT_FOUND	(value: 0x00000003)	TextID not found
TXT_RT_NO_READ_ACCESS	(value: 0x00000004)	Read access to the memory mapped file denied
TXT_RT_NO_LANGUAGE_FOUND	(value: 0x00000005)	No language found
TXT_RT_LANGUAGE_NOT_FOUND	(value: 0x00000006)	Language not found
TXT_RT_INTERNAL_ERROR	(value: 0x00000007)	Internal fault

3.6 Functions of the text system (RT Professional)

TXT_RT_INVALIDPARAM	(value: 0x00000008)	Parameter error
TXT_RT_NODEFAULTSERVER	(value: 0x00000011)	No default server is configured.
TXT_RT_NOLOCALSERVER	(value: 0x00000012)	No local server is available.
TXT_RT_NOSERVER	(value: 0x00000013)	No default server is configured, and no local server is available.
TXT_RT_NOMC	(value: 0x00000014)	This is not a multiclient project (not used here)
TXT_RT_NOMCDEFAULT-SERVER	(value: 0x00000015)	This is not a multiclient project, but "@default" was given as a server prefix.

3.6.1.3 Language codes (RT Professional)

Language codes of the text system

Only the SUBLANG_DEFAULT languages of Windows are supported in WinCC. As a result, the the following values can be assigned to the dwLocaleID parameter:

LANG_ARABIC	0x0401
LANG_AFRIKAANS	0x0436
LANG_ALBANIAN	0x041C
LANG_BASQUE	0x042D
LANG_BULGARIAN	0x0402
LANG_BELARUSIAN	0x0423
LANG_CATALAN	0x0403
LANG_CHINESE	0x0804
LANG_CROATIAN	0x041A
LANG_CZECH	0x0405
LANG_DANISH	0x0406
LANG_DUTCH	0x0413
LANG_ENGLISH	0x0409
LANG_ESTONIAN	0x0425
LANG_FAEROESE	0x0438
LANG_FARSI	0x0429
LANG_FINNISH	0x040B
LANG_FRENCH	0x040C
LANG_GERMAN	0x0407
LANG_GREEK	0x0408
LANG_HEBREW	0x040D
LANG_HUNGARIAN	0x040E
LANG_ICELANDIC	0x040F
LANG_INDONESIAN	0x0421
LANG_ITALIAN	0x0410
LANG_JAPANESE	0x0411
LANG_KOREAN	0x0412

LANG_LATVIAN	0x0426
LANG_LITHUANIAN	0x0427
LANG_NORWEGIAN	0x0414
LANG_POLISH	0x0415
LANG_PORTUGUESE	0x0416
LANG_ROMANIAN	0x0418
LANG_RUSSIAN	0x0419
LANG_SLOVAK	0x041B
LANG_SLOVENIAN	0x0424
LANG_SPANISH	0x040A
LANG_SWEDISH	0x041D
LANG_THAI	0x041E
LANG_TURKISH	0x041F
LANG_UKRAINIAN	0x0422

Here, the upper 4 bytes (0x04..) designate the language as SUBLANG_DEFAULT language

3.6.2 General functions (RT Professional)

3.6.2.1 TXTCloseProject (RT Professional)

Description

Closes the project database upon successful execution.

Declaration

```

BOOL TXTCloseProject (
    LPCTSTR      lpszProjectFile,
    LPCMN_ERROR  lpdmError );

```

Parameters

lpszProjectFile

Pointer to the name of the project file including path and extension.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Text library closed.

FALSE

Error

Comment

The TXTOpenProject function increments an internal reference counter with each call. This counter is decremented with call of TXTCloseProject. The text library is closed only when the reference counter returns to 0.

Therefore, the same number of TXTCloseProject must always be called, otherwise an exception error will occur when the application closes.

Note

The call may not be used in the destructor of an application (EXE, DLL, OCX, etc.). Due to Microsoft-specific mechanisms, this can cause the call to freeze and thereby crash the program.

Error messages

TXT_DISCONNECT	Error logging off of the database

Required files

text_cs.h

text_cs.lib

text_cs.dll

Related functions

TXTOpenProject (Page 2044)	Opens the text library

Examples

Get infotext (Page 2065) "TX01.cpp"

Enumerate infotexts (Page 2067) "TX01.cpp"

See also

[TXTOpenProject \(Page 2044\)](#)

[Get infotext \(Page 2065\)](#)

[Enumerate infotexts \(Page 2067\)](#)

[Overview of the functions \(Page 2036\)](#)

3.6.2.2 TXTGetMaxTextID (RT Professional)**Description**

Records the highest text ID that is kept in the DLL.

Declaration

```
BOOL TXTGetMaxTextID (  
    LPLONG          lpIMaxTextID,  
    LPCMN_ERROR     lpdmError );
```

Parameters**lpIMaxTextID**

Pointer to the buffer in which the text ID should be stored.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Maximum text ID determined

FALSE

Error

Required files

text_cs.h

text_cs.lib

text_cs.dll

See also

Overview of the functions (Page 2036)

3.6.2.3 TXTOpenProject (RT Professional)

Description

Opens the database upon successful execution.

Declaration

```
BOOL TXTOpenProject (
    LPCTSTR      lpszProjectFile,
    LPCTSTR      lpszDSNName,
    BOOL         fExclusive,
    LPCMN_ERROR  lpdmError );
```

Parameters

lpszProjectFile

Pointer to the name of the project file including path and extension.

lpszDSNName

Pointer on the name of the data source.

fExclusive

If TRUE, TEXTBIB.EXE cannot be started, because another program is accessing the text library, for example. If TEXTBIB.EXE has already started, the call is rejected.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Database open

FALSE

Error

Comment

If no text library tables are available, they are generated new.

Each time the function called a reference counter is incremented. The same number as TXTCloseProject must be present to close. Otherwise, an exception error can occur when the application closes.

Error messages

TXT_CONNECT	Error connecting to the database.
TXT_TABLE_NOT_FOUND	A database table was not found. The dwError2 field contains more detailed information:
TXT_TEXT_LIBRARY_RUNNING	TEXTBIB.EXE has already started
TXT_WRONG_PROJECT	Wrong project name

Required files

text_cs.h
text_cs.lib
text_cs.dll

Related functions

TXTCloseProject (Page 2039)	Closes the text library

Examples

Get infotext (Page 2065) "TX01.cpp"
Enumerate infotexts (Page 2067) "TX01.cpp"

See also

TXTCloseProject (Page 2039)
Get infotext (Page 2065)
Enumerate infotexts (Page 2067)
Overview of the functions (Page 2036)

3.6.2.4 TXTRTConnect (RT Professional)

Description

Establishes a connection to the text server. This function accelerates the execution of TXTRT functions. The connection to the text server is then not made for each call and closed again at the end.

Declaration

```
BOOL TXTRTConnect ( );
```

Parameters

None

Return value

- TRUE**
Connects to the text server established
- FALSE**
Error

Comment

Before closing the application, the connection must always be closed with TXTRTDisconnect to avoid subsequent malfunction in the system.

Required files

- text_rt.h
- text_rt.lib
- text_rt.dll

Related functions

TXTRTDisconnect (Page 2046)	Connection to the text server released

See also

- TXTRTDisconnect (Page 2046)
- Overview of the functions (Page 2036)

3.6.2.5 TXTRTDisconnect (RT Professional)

Description

Closes a connection to the text server that was made with TXTRTConnect.

Declaration

```
BOOL TXTRTDisconnect ( );
```

Parameters

None

Return value

TRUE

Connects to the text server established

Comment

This function stops the acceleration provided by the TXTRT functions and always returns TRUE.

Subsequently called TXTRT functions then establish a new connection to the text server for each call and then close it.

If TXTRTConnect is used and no TXTRTDisconnect is executed before closing the application, a malfunction may subsequently occur in the system.

Note

The call may not be used in the destructor of an application (EXE, DLL, OCX, etc.). Due to Microsoft-specific mechanisms, this can cause the call to freeze and thereby crash the program.

Required files

text_rt.h

text_rt.lib

text_rt.dll

Related functions

TXTRTConnect (Page 2043)	Keep connection to text server

See also

TXTRTConnect (Page 2043)

Overview of the functions (Page 2036)

3.6.3 Functions for processing infotext (RT Professional)

3.6.3.1 TXTEnumInfoText (RT Professional)

Description

Lists all infotext that meets the filter criterion in lpszFilter.

Declaration

```
BOOL TXTEnumInfoText (
    LPCTSTR                lpszProjectFile,
    DWORD                  dwLocale,
    LPDWORD                 lpdwItems,
    LPCTSTR                 lpszFilter,
    TXT_ENUM_INFOTEXTS_PROC lpfnEnum,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError );
```

Parameters

lpszProjectFile

Pointer to the name of the project file including path and extension.

dwLocale

Language code of the language, the infotext of which should be enumerated.

lpdwItems

Pointer to a double word of the application, which contains the number of enumerated infotexts after the call.

lpszFilter

Pointer to the condition of an SQL statement for the LIKE operator.

lpfnEnum

Your callback function that receives the infotext.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Infotext listed

FALSE

Error

Comment

If IpszFilter == NULL or an empty string is passed, all infotexts of the language are enumerated. Empty, unassigned info texts can also be enumerated and included using lpdwltems.

Error messages

TXT_CALLBACK	Callback returns FALSE
TXT_LANGUAGE_NOT_FOUND	The specified language could not be found.
TXT_NO_CONNECT	The database is not yet open.
TXT_NO_LANGUAGE	Invalid language specified
TXT_WRONG_PROJECT	Wrong project name

Required files

text_cs.h

text_cs.lib

text_cs.dll

Related functions

TXT_ENUM_INFOTEXTS_PROC (Page 2050)	List infotext (callback)

Examples

Enumerate infotexts (Page 2067) "TX01.cpp"

See also

TXT_ENUM_INFOTEXTS_PROC (Page 2050)

Enumerate infotexts (Page 2067)

Overview of the functions (Page 2036)

3.6.3.2 TXT_ENUM_INFOTEXTS_PROC (RT Professional)

Description

In order to evaluate the listed tool tips, you must provide a TXT_ENUM_INFOTEXTS_PROC type callback function.

Declaration

```
BOOL ( * TXT_ENUM_INFOTEXTS_PROC) (
    DWORD      dwTextID,
    LPCTSTR    lpszInfoText,
    LPVOID      lpvUser );
```

Parameters

dwTextID

ID of the text to which lpszInfotext refers.

lpszInfoText

Pointer to the tool tip that was transferred by the calling function.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued

FALSE

The enumeration is cancelled

Comment

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

text_cs.h
text_cs.lib
text_cs.dll

Related functions

TXTEnumInfoText (Page 2046)	List infotext

Examples

Enumerate infotexts (Page 2067) "TX01.cpp"

See also

TXTEnumInfoText (Page 2046)
Enumerate infotexts (Page 2067)
Overview of the functions (Page 2036)

3.6.3.3 TXTUpdateRuntime (RT Professional)**Description**

The text from the text library are reloaded in runtime.

Declaration

```

BOOL TXTUpdateRuntime (
    LPCTSTR      lpszProjectFile,
    HWND         hwndParent,
    LPCMN_ERROR  lpdmError );

```

Parameters**lpszProjectFile**

Pointer to the name of the project file including path and extension.

hwndParent

Handle to the parent window

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Infotext updated

Comment

No update is performed for text reference tags. The function always returns TRUE.

Required files

text_cs.h

text_cs.lib

text_cs.dll

See also

Overview of the functions (Page 2036)

3.6.3.4 TXTRTGetInfoText (RT Professional)

Description

Upon successful execution, this function fills the buffer passed to lpszBuffer with the text described by dwTextID. If the passed buffer is smaller than the text length (buffer length in characters in pdwSize), the text is accordingly truncated. The return value is still TRUE. If NULL is passed in lpszBuffer, the function determines the required buffer size and stores it in pdwSize.

Declaration

```
BOOL TXTRTGetInfoText (  
    DWORD          dwTextID,  
    LPTSTR         lpszBuffer,  
    LPDWORD        pdwSize,  
    LPCMN_ERROR    lpdmError );
```

Parameters**dwTextID**

ID of the text to be read

lpzBuffer

Pointer to a buffer in which the infotext should be stored.

pdwSize

Pointer to the DWORD, which contains the buffer size

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Infotext determined

FALSE

Error

Comment**Error messages**

TXT_RT_OPEN_MMF	Error while opening the memory mapped file
TXT_RT_CREATE_SERVICE_WINDOW	Error while creating the service window
TXT_KEY_NOT_FOUND	The specified TextID could not be found.
TXT_RT_NO_READ_ACCESS	Read access to the memory mapped file denied

Required files

text_rt.h

text_rt.lib

text_rt.dll

Related functions

AUTOHOTSPOT	Read tool tips from the text server

Examples

Get infotext (Page 2065) "TX01.cpp"

See also

Get infotext (Page 2065)

Overview of the functions (Page 2036)

3.6.3.5 TXTRTGetInfoTextMC (RT Professional)

Description

Fetches a text, for example, a tooltip from the specified text server or from the C text list. Upon successful execution, this function fills the buffer passed to `lpszBuffer` with the text described by `dwTextID`. If the passed buffer is smaller than the text length (buffer length in characters in `pdwSize`), the text is accordingly truncated. The return value is still TRUE. If NULL is passed in `lpszBuffer`, the function determines the required buffer size and stores it in `pdwSize`.

Declaration

```
BOOL TXTRTGetInfoTextMC (
    DWORD          dwTextID,
    LPTSTR          lpszBuffer,
    LPDWORD         pdwSize,
    LPTSTR          lpszServer,
    LPCMN_ERROR     lpdmError );
```

Parameters

dwTextID

ID of the text to be read

lpszBuffer

Pointer to a buffer in which the infotext should be stored.

pdwSize

Pointer to the DWORD, which contains the buffer size

lpszServer

Pointer to the symbolic name of the text server (without server delimiter ::)

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Infotext determined

FALSE

Error

Comment

This function only works with multiclient projects

Error messages

TXT_RT_OPEN_MMF	Error while opening the memory mapped file
TXT_RT_CREATE_SERVICE_WINDOW	Error while creating the service window
TXT_KEY_NOT_FOUND	The specified TextID could not be found.
TXT_RT_NO_READ_ACCESS	Read access to the memory mapped file denied
TXT_RT_NODEFAULTSERVER	No default server is configured.
TXT_RT_NOLOCALSERVER	No local server is available.
TXT_RT_NOSERVER	No default server is configured and no local server is available.
TXT_RT_NOMCDEFAULTSERVER	This is not a multiclient project, but "@default" was given as a server prefix.

Required files

text_rt.h

text_rt.lib

text_rt.dll

Related functions

AUTOHOTSPOT	Read tool tips

See also

Overview of the functions (Page 2036)

3.6.4 Functions for processing languages (RT Professional)

3.6.4.1 TXTEnumLanguages (RT Professional)

Description

Calls the passed callback function for all configured languages.

Declaration

```
BOOL TXTEnumLanguages (
    LPCTSTR                lpszProjectFile,
    LPDWORD                lpdwItems,
    TXT_ENUM_LANGUAGES_PROC lpfnEnum,
    LPVOID                 lpvUser,
    LPCMN_ERROR             lpdmError );
```

Parameters

lpszProjectFile

Pointer to the name of the project file including path and extension.

lpdwItems

Pointer to a double word of the application, which contains the number of enumerated languages after the call.

lpfnEnum

Your callback function that receives the configured languages.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Configured languages listed

FALSE

Error

Error messages

TXT_CALLBACK	Callback returns FALSE
TXT_NO_CONNECT	The database is not yet open.
TXT_WRONG_PROJECT	Wrong project name

Required files

text_cs.h
text_cs.lib
text_cs.dll

Related functions

TXT_ENUM_LANGUAGES_PROC (Page 2057)	List configured languages (callback)

Examples

Enumerate infotexts (Page 2067) "TX01.cpp"

See also

TXT_ENUM_LANGUAGES_PROC (Page 2057)
Enumerate infotexts (Page 2067)
Overview of the functions (Page 2036)

3.6.4.2 TXT_ENUM_LANGUAGES_PROC (RT Professional)**Description**

In order to evaluate the listed languages, you must provide a TXT_ENUM_LANGUAGES_PROC type callback function.

Declaration

```

BOOL ( * TXT_ENUM_LANGUAGES_PROC ) (
    DWORD      dwLocaleID,
    LPCTSTR    lpszName,
    LPVOID     lpvUser );

```

Parameters

dwLocaleID

Language code

lpzName

Pointer to the name of the language.

lpvUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Comment

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

text_cs.h

Related functions

TXTEnumLanguages (Page 2054)	List configured languages

Examples

Enumerate infotexts (Page 2067) "TX01.cpp"

See also

TXTEnumLanguages (Page 2054)
Enumerate infotexts (Page 2067)
Overview of the functions (Page 2036)

3.6.4.3 TXTGetFont (RT Professional)

Description

Upon successful execution, fills the LOGFONT structure passed in lpLf with the font set for the desired language.

Declaration

```
BOOL TXTGetFont (
    LPCTSTR      lpszProjectFile,
    DWORD        dwLocale,
    LPLOGFONT     lpLf,
    LPCMN_ERROR   lpdmError );
```

Parameters

lpszProjectFile

Pointer to the name of the project file including path and extension.

dwLocale

Language code of the language, the font of which should be fetched

lpLf

Pointer to a Windows-specific LOGFONT type structure with information about a character set.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Character set determined

FALSE

Error

Error messages

TXT_NO_CONNECT	The database is not yet open.
TXT_NO_LANGUAGE	Invalid language specified
TXT_LANGUAGE_NOT_FOUND	The specified language could not be found.
TXT_WRONG_PROJECT	Wrong project name

Required files

text_cs.h

text_cs.lib

text_cs.dll

See also

[Overview of the functions \(Page 2036\)](#)

3.6.4.4 TXTShowLanguagesDialog (RT Professional)

Description

Opens a dialog with the available languages. If the dialog is closed with "OK", the selected language is stored in `lpdwLocale`. In addition, the corresponding font is passed in the `LOGFONT` structure.



Declaration

```
BOOL TXTShowLanguagesDialog (
    LPCTSTR          lpszProjectFile,
    HWND             hwndParent,
    LPDWORD          lpdwLocale,
    LPLOGFONT         lpjf,
    LPCMN_ERROR       lpdmError );
```

Parameters

lpszProjectFile

Pointer to the name of the project file including path and extension.

hwndParent

Handle to the parent window of the dialog box. The parameter is set to NULL by default.

dwLocale

Pointer to the code of the language to be installed.

lpjf

Pointer to a Windows-specific LOGFONT type structure with information about a character set.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Language selected

FALSE

Error or close dialog with "Cancel"

Comment

In order to offer the installed languages in the dialog box, the languages in the database are enumerated by the function. If an error occurs, the TXT_CALLBACK error message is stored in the error structure.

Error messages

TXT_CALLBACK	Callback returns FALSE
TXT_NO_CONNECT	The database is not yet open.
TXT_WRONG_PROJECT	Wrong project name

Required files

text_cs.h

text_cs.lib

text_cs.dll

See also

Overview of the functions (Page 2036)

3.6.4.5 TXTRTGetLanguageID (RT Professional)

Description

Upon successful executions, fills the buffer passed in lpchLanguageID with the primary language ID loaded in the memory mapped file.

Declaration

```
BOOL TXTRTGetLanguageID (  
    LPBYTE      lpchLanguageID,  
    LPCMN_ERROR lpdmError );
```

Parameters

lpchLanguageID

Pointer to a buffer, in which the primary language ID should be stored.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Activated language determined

FALSE

Error

Error messages

TXT_RT_OPEN_MMF	Error while opening the memory mapped file
TXT_RT_CREATE_SERVICE_WINDOW	Error while creating the service window
TXT_RT_NO_LANGUAGE_FOUND	No language found

Required files

text_rt.h

text_rt.lib

text_rt.dll

See also

Overview of the functions (Page 2036)

3.6.4.6 TXTRTSetLanguage (RT Professional)**Description**

Sets the language for error messages. If the language is not available, the default language (German) is set and FALSE is returned.

This function is no longer supported and always returns the value, TRUE.

Declaration

```

BOOL TXTRTSetLanguage (
    DWORD      dwLocaleID,
    LPCMN_ERROR lpdmError );

```

Parameters**dwLocaleID**

Language code of the language to be set.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Language of error messages switched

Required files

text_rt.h

text_rt.lib

text_rt.dll

See also

Overview of the functions (Page 2036)

3.6.5 Examples (RT Professional)

3.6.5.1 Get infotext (RT Professional)

Example

```
//{{ODK_EXAMPLE}Insert new Infotext (TXT)}
//{{FUNCTION}TXTOpenProject (TXT)}
//{{FUNCTION}TXTNewInfoText (TXT)}
//{{FUNCTION}TXTSetInfoText (TXT)}
//{{FUNCTION}TXTCloseProject (TXT)}
//{{FUNCTION}(END)}
// =====
// Function: MyTxtSetInfotext(void) ODK DM CS
// =====
// Abst.      :      Insert New Text
// =====
void MyTxtNewInfotext(void)
{
    TCHAR szText[255];
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwLocale = 0x0407; // german
    TCHAR szFilter[]="S%"; // all texts with 'S'
    DWORD dwSize =254;
    DWORD dwTextID = 0;
    ret = MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile and
g_szDSNName
    if(TRUE == ret)
    {
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = TXTOpenProject(g_szProjectFile, g_szDSNName, FALSE, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TXTOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTOpenProject"));
            ODKTrace(szText);
            dwTextID = 380;
            dwLocale = 0x0407; // german
            _tcsncpy_s(szText, _countof(szText), _T("NewInfoText_ODK"), _TRUNCATE);
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = TXTGetInfoText(g_szProjectFile, dwTextID, dwLocale, szText, &dwSize,
&Error);
            if(FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTNewInfoText: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
```

3.6 Functions of the text system (RT Professional)

```

        Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTNewInfoText"));
    }
    ODKTrace(szText);
    dwLocale=0x0409; // english
    //dwTextID = 380; // use ID from TXTNewInfoText
    _tcsncpy_s(szText, _countof(szText), _T("InfoText_ODK"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TXTSetInfoText(g_szProjectFile, NULL, dwTextID, dwLocale, szText, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTSetInfoText: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTSetInfoText"));
    }
    ODKTrace(szText);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TXTCloseProject(g_szProjectFile, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTCloseProject"));

        ODKTrace(szText);
    }
    }
}
}
//{{ODK_EXAMPLE} (END) }

```

See also

[TXTCloseProject \(Page 2039\)](#)
[TXTOpenProject \(Page 2042\)](#)
[TXTRTGetInfoText \(Page 2050\)](#)

3.6.5.2 Enumerate infotexts (RT Professional)

Example

```
//{{ODK_EXAMPLE}Enumerate texts (TXT)}
//{{FUNCTION}TXTOpenProject (TXT)}
//{{FUNCTION}TXTCloseProject (TXT)}
//{{FUNCTION}TXTEnumLanguages (TXT)}
//{{FUNCTION}TXT_ENUM_LANGUAGES_PROC (TXT)}
//{{FUNCTION}TXTEnumInfoText (TXT)}
//{{FUNCTION}TXT_ENUM_INFOTEXTS_PROC (TXT)}
//{{FUNCTION}(END)}
// =====
// Function: MyTxtEnums(void) ODK DM CS
// =====
// Abst. : Enumerate Texts
// =====

BOOL MyTXTEnumLanguagesCallback (DWORD dwLocID, LPCTSTR lpszName, LPVOID lpvUser)
{
    lpvUser;
    TCHAR szText[255];
    BOOL ret = TRUE;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    InfoLang: ID=%x ;
    ODKTrace(szText);
    return(ret);
}

BOOL MyTXTEnumInfotextsCallback(DWORD dwTextID, LPCTSTR lpszInfoText, LPVOID lpvUser)
{
    lpvUser;
    TCHAR szText[255];
    BOOL ret = TRUE;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    InfoText: ID=%.4d ; Tx=%s "),
dwTextID, lpszInfoText);
    ODKTrace(szText);
    return(ret);
}

void MyTxtEnums(void)
{
    TCHAR szText[255];
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwLocale = 0x0407; // german
    DWORD dwItems = 0;
    DWORD dwUser = 0;
    TCHAR szFilter[]="S%"; // all texts with 'S'
    //DWORD dwSize =254;
    ret = MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile and
g_szDSNName
    if(TRUE == ret)
    {
        memset(&Error, 0, sizeof(CMN_ERROR));
    }
}
```

3.6 Functions of the text system (RT Professional)

```

ret = TXTOpenProject(g_szProjectFile, g_szDSNName, FALSE, &Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TXTOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTOpenProject"));
    ODKTrace(szText);
    dwItems = 0;
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TXTEnumLanguages(g_szProjectFile, &dwItems, MyTXTEnumLanguagesCallback,
&dwUser, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTOpenProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    dwItems = 0;
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TXTEnumInfoText(g_szProjectFile, dwLocale, &dwItems, szFilter,
MyTXTEnumInfotextsCallback, &dwUser, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTEnumInfoText: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTEnumInfoText: NItems=
%d ; LANG=%x ; Filter=%s "),
        dwItems, dwLocale, szFilter);
    ODKTrace(szText);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TXTCloseProject(g_szProjectFile, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
}
}
}
//{{ODK_EXAMPLE} (END) }

```

See also

TXTCloseProject (Page 2039)

TXTOpenProject (Page 2042)

TXTEnumInfoText (Page 2046)

TXT_ENUM_INFOTEXTS_PROC (Page 2048)

TXT_ENUM_LANGUAGES_PROC (Page 2055)

TXTEnumLanguages (Page 2054)

3.7 Report system functions (RT Professional)**3.7.1 Basics (RT Professional)****3.7.1.1 Overview of the functions (RT Professional)****Overview**

RPJAttach (Page 2081)	Establish connection
RPJCallJobMethod (Page 2104)	Call print job method
RPJCreateJob (Page 2091)	Create a print job
RPJCreatePropertyHandle (Page 2094)	Create handle of a print job property
RPJDeleteJob (Page 2092)	Delete print job
RPJDeletePropertyHandle (Page 2095)	Delete handle of a print job property
RPJDetach (Page 2082)	Disconnect connection
RPJGetJobMethodAt (Page 2105)	Determine name of print job method
RPJGetJobNameAt (Page 2097)	Determine name of print job
RPJGetJobPropertyAt (Page 2107)	Determine name of print job properties
RPJGetJobProps (Page 2109)	Determine print job properties
RPJGetNumJobMethods (Page 2106)	Determine number of print job methods
RPJGetNumJobProperties (Page 2110)	Determine number of print job properties
RPJGetNumJobs (Page 2098)	Determine number of print jobs
RPJGetNumProjectProperties (Page 2084)	Determine number of properties in the project
RPJGetProjectPropertyAt (Page 2085)	Determine property name and type in the project using an index
RPJGetProjectProperty (Page 2086)	Determine value of the property in the project
RPJGetProperty (Page 2111)	Determine print job property
RPJJobLock (Page 2099)	Lock print job against other accesses
RPJJobUnlockAll (Page 2102)	Remove all print job locks
RPJJobUnlock (Page 2101)	Remove a print job lock
RPJMemFree (Page 2083)	Free memory

3.7 Report system functions (RT Professional)

RPJProjectLock (Page 2087)	Lock access to the print job list in the project
RPJProjectUnlockAll (Page 2090)	Remove all print job list locks in the project
RPJProjectUnlock (Page 2089)	Remove a print job list lock in the project
RPJPropertyClear (Page 2114)	Delete handle of a print job property
RPJSetProperty (Page 2115)	Specify print job properties

3.7.1.2 Constants (RT Professional)

Properties for layouts (CS)

Root

- ObjectName
- Context

Geometry

- Left
- Top
- Width
- Height
- RoundCornerWidth
- RoundCornerHeight
- StartAngle
- EndAngle
- Radius
- RadiusWidth
- RadiusHeight
- Columns
- Index
- PointCount
- ActualPointLeft
- ActualPointTop
- PaperSize
- Orientation
- DynMarginLeft
- DynMarginRight
- DynMarginTop
- DynMarginBottom
- PrintMarginLeft

- PrintMarginRight
- PrintMarginTop
- PrintMarginBottom

Styles

- BorderStyle
- BorderWidth
- FillStyle

Colors

- BorderBackColor
- BorderColor
- BackColor
- FillColor
- ForeColor

Characters

- FontName
- FontSize
- FontBold
- FontItalic
- FontUnderline
- Text
- AlignmentLeft
- AlignmentTop
- WordWrap

Miscellaneous

- FirstPage
- LastPage
- MetaFileName
- LayoutFileName
- Format
- List
- Tag
- DataType
- OutFormat
- Calculation
- PageBreak

Properties for print jobs

AbsoluteSelectionFrom	
AbsoluteSelectionTo	
CycleSpan	
DestinationFile	
EnableCycle	
EnableStart	
EndPage	This information enables you to end the printout of the report at a particular page.
JobName	The name of the print job must be unique within a project and conform to the Windows conventions.
LayoutName	You can assign the print job a layout using the layout name.
PrinterName 1	Here, you specify the printer to which the printout is initially sent.
PrinterName 2	Here, you specify the printer to be used if the first printer is not available.
PrinterName 3	Here, you specify the printer to be used if the first two printers are not available.
RelativeSelectionCount	
RelativeSelectionRange	
StartPage	This information enables you to start the printout of the report at a particular page.
StartTime	
UseRelative	
UseOutputFile	
UseOutputPrinter	

3.7.1.3 Error messages (RT Professional)**Overview**

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

Error messages of CS

ERR_NOERROR	0	No error
ERR_ILLEGALPROJECT	1	Invalid project name/project path
ERR_NOMEMORY	2	Memory error
ERR_UNKNOWNERROR	3	Unknown error
ERR_THREADNOTINITIALIZED	4	The utilized thread is not initialized.
ERR_OLEEXCEPTION	5	Error in conjunction with the MFC and OLE
ERR_NOOLESERVERAVAIL	6	No OLE-Server available

Error messages of RT

ERR_NOERROR	0	No error
ERR_ILLEGALPROJECT	1	Invalid project name/project path
ERR_NOMEMORY	2	Memory error
ERR_UNKNOWNERROR	3	Unknown error
ERR_NULLHANDLE	4	Handle could not be created.
ERR_ILLEGALPOINTER	5	Pointer is incorrect or invalid
ERR_ILLEGALJOBINDEX	6	Print job index is incorrect
ERR_UNKNOWNPROPERTY	7	Print job property is unknown
ERR_UNKNOWNMETHOD	8	Print job method is unknown

3.7.1.4 List of object properties (RT Professional)

Properties for all objects

Note

The property names along with a basic description can also be obtained via the context-sensitive help. To access this, right-click inside the properties box for the relevant object in the Report Designer.

Property name	Data type
"Width"	VT_I4
"Height"	VT_I4
"ObjectName"	VT_BSTR
"PageBreak"	VT_I4
"BorderStyle"	VT_I4

"BorderWidth"	VT_I4
"BorderBackColor"	VT_I4
"BorderColor"	VT_I4
"Left"	VT_I4
"Top"	VT_I4

Properties for all "2-dimensional objects":

"FillStyle"	VT_I4
"BackColor"	VT_I4
"FillColor"	VT_I4

Properties for "dynamic objects":

"DataLink"	VT_BSTR

The following properties are object-specific!

Properties for all "system" objects:

Property name	Data type
"FontName"	VT_BSTR
"FontSize"	VT_I4
"FontBold"	VT_I4
"FontItalic"	VT_I4
"FontUnderline"	VT_I4
"Format"	VT_BSTR
"AlignmentLeft"	VT_I4
"AlignmentTop"	VT_I4
"WordWrap"	VT_I4

Properties for "dynamic tables":

"FontName"	VT_BSTR
"FontSize"	VT_I4
"FontBold"	VT_I4
"FontItalic"	VT_I4
"FontUnderline"	VT_I4
"Orientation"	VT_I4
"Columns"	VT_BSTR
"ForeColor"	VT_I4
"List"	VT_I4

Properties for "dynamic texts":

"FontName"	VT_BSTR
"FontSize"	VT_I4
"FontBold"	VT_I4
"FontItalic"	VT_I4
"FontUnderline"	VT_I4
"Orientation"	VT_I4
"AlignmentLeft"	VT_I4
"AlignmentTop"	VT_I4

"WordWrap"	VT_I4
"ForeColor"	VT_I4

Properties for "dynamic picture":

"DynHeight"	VT_I4

Properties for "circular arc":

"Radius"	VT_I4
"StartAngle"	VT_I4
"EndAngle"	VT_I4

Properties for "elliptical arc":

"StartAngle"	VT_I4
"EndAngle"	VT_I4
"RadiusWidth"	VT_I4
"RadiusHeight"	VT_I4

Properties for "circles":

"Radius"	VT_I4

Properties for "ellipses":

"RadiusWidth"	VT_I4
"RadiusHeight"	VT_I4

Properties for "polygons":

"Polyline"	VT_BSTR
"PointCount"	VT_I4
"Index"	VT_I4
"ActualPointLeft"	VT_I4
"ActualPointTop"	VT_I4

Properties for "circle segments":

"Radius"	VT_I4
"StartAngle"	VT_I4
"EndAngle"	VT_I4

Properties for "ellipse segments":

"StartAngle"	VT_I4
"EndAngle"	VT_I4
"RadiusWidth"	VT_I4
"RadiusHeight"	VT_I4

Properties for "polyline":

"Polyline"	VT_BSTR
"PointCount"	VT_I4
"Index"	VT_I4
"ActualPointLeft"	VT_I4
"ActualPointTop"	VT_I4

Properties for "rounded rectangle":

"RoundCornerWidth"	VT_I4
"RoundCornerHeight"	VT_I4

Properties for "text objects":

"FontName"	VT_BSTR
"FontSize"	VT_I4
"FontBold"	VT_I4
"FontItalic"	VT_I4
"FontUnderline"	VT_I4
"Orientation"	VT_I4
"Text"	VT_BSTR
"AlignmentLeft"	VT_I4
"AlignmentTop"	VT_I4
"WordWrap"	VT_I4
"ForeColor"	VT_I4

Properties for "inserted layouts":

"LayoutFileName"	VT_BSTR

Properties for "layouts":

"PrintMarginLeft"	VT_I4
"PrintMarginRight"	VT_I4
"PrintMarginTop"	VT_I4
"PrintMarginBottom"	VT_I4
"PaperSize"	VT_BSTR
"FirstPage"	VT_I4
"LastPage"	VT_I4
"Orientation"	VT_I4
"Printer1"	VT_BSTR
"Printer2"	VT_BSTR
"Printer3"	VT_BSTR
"BackColor"	VT_I4
"BorderColor"	VT_I4
"FillColor"	VT_I4
"DynMarginLeft"	VT_I4
"DynMarginRight"	VT_I4
"DynMarginTop"	VT_I4
"DynMarginBottom"	VT_I4
"Context"	VT_BSTR

Properties for "MetaFiles":

"MetaFileName"	VT_BSTR

Properties for "tag" objects:

"Tag"	VT_BSTR
"FontSize"	VT_I4
"FontName"	VT_BSTR
"FontBold"	VT_I4
"FontItalic"	VT_I4
"FontUnderline"	VT_I4
"AlignmentLeft"	VT_I4
"AlignmentTop"	VT_I4
"WordWrap"	VT_I4
"ForeColor"	VT_I4
"OutFormat"	VT_BSTR

"DataType"	VT_I4
"Calculation"	VT_VARIANT

3.7.1.5 General procedure for processing print job properties (Report Designer) (RT Professional)

Establishing and initializing connection to RPJAPI.DLL

RPJAttach.

Creating a handle for print job properties

To process the properties of a print job, a print job handle must be created. This handle is used to identify a memory area through which the print job properties can be addressed.

Function: RPJCreatePropertyHandle.

Reading print job properties

The properties of the print jobs are stored in the project database. To edit the properties, they must be loaded from the database to the area specified by the handle.

Function: RPJGetJobProps.

Determining print job properties

A certain print job property is referenced using its name. To determine the value of a property, it must be read.

Function: RPJGetProperty.

Specifying print job properties

A certain print job property is referenced using its name. To set the property, the RPJSetProperty function is called.

Storing print job properties

The modified print job properties are stored in the project database. The name of the print job to be stored must be specified.

Function: RPJSetJobProps.

Reinitializing a print job property

The memory area referenced by the handle is reinitialized without freeing the handle.

Function: RPJPropertyClear.

Specifying print job properties

A certain print job property is referenced using its name. The property is set by calling a function.

Function: RPJSetProperty.

Storing print job properties

The modified print job properties are stored in the project database. The name of the print job to be stored must be specified.

Function: RPJSetJobProps.

Deleting a handle

A print job property handle is deleted. Delete each handle created with RPJCreatePropertyHandle when it is no longer needed.

Function: RPJDeletePropertyHandle.

Terminating connection to RPJAPI.DLL

RPJDetach.

Example: Starting a print job in Print Preview

General

To process the properties of a print job, a print job handle must be provided. This handle is used to identify a memory area through which the print job properties can be addressed.

Establishing and initializing connection to RPJAPI.DLL

RPJAttach.

Creating a handle: Function

RPJCreatePropertyHandle.

Reading in properties of the print job

The properties of the print job: "Control Center Redocumentation" are loaded from the database to the area specified by the hProp.

Function: RPJGetJobProps.

Initiating print job method

The print job is initiated by specifying the PREVIEW method. The print job itself is specified by the hProp.

Function: RPJCallJobMethod.

Deleting a handle

A print job property handle is deleted. Each handle created with RPJCreatePropertyHandle should be deleted when it is no longer needed.

Function: RPJDeletePropertyHandle.

Terminating connection to RPJAPI.DLL

RPJDetach.

Example

```
void PrintJob(void)
{
    CMN_ERROR err; //error structure
    BOOL ret;
    HPROPERTIES hProp;
    char jobname[200];
    char method[200];
    LPCSTR szProjectName = "c:\\rest\\test.mcp"; //name of project
    strcpy(method, "PREVIEW");
    strcpy(jobname, "Backdokumentation Control Center");
    ret = RPJAttach(&err);
    if (TRUE == ret)
    {
        hProp = RPJCreatePropertyHandle(szProjectName, &err);
        if (NULL == hProp)
        {
            ErrMsg("Error RPJCreatePropertyHandle", &err);
        }
        else
        {
            ret = RPJGetJobProps(hProp, jobname, &err);
            if (FALSE == ret)
            {
                ErrMsg("Error RPJGetJobProbs", &err);
            }
            else
            {
                ret = RPJCallJobMethod(hProp, method, &err);
                if (FALSE == ret)
                {
                    ErrMsg("Error executing RPJCallJobMethod", &err);
                }
                else
                {
                    Msg("Print job started.");
                }
            }
            RPJDeletePropertyHandle(hProp, &err);
        }
        RPJDetach(&err);
    }
    else
    {
        Msg("No connection to report designer DLL");
    }
}
```

3.7.2 Functions for establishing connections (RT Professional)

3.7.2.1 RPJAttach (RT Professional)

Description

This function is used to established and initialized a connection to RPJAPI.DLL.

Declaration

```
BOOL RPJAttach (
    CMN_ERROR*    pcmnerror );
```

Parameters

pcmerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection established.

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJDetach (Page 2082)	Terminate connection

See also

RPJDetach (Page 2082)

Overview of the functions (Page 2067)

3.7.2.2 RPJDetach (RT Professional)

Description

This function is used to close an established connection to RPJAPI.DLL.

Declaration

```
BOOL RPJDetach (  
    CMN_ERROR*    pcmnerror );
```

Parameters

pcmerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection is closed again.

FALSE

Error

Comment

Note

The call may not be used in the destructor of an application (EXE, DLL, OCX, etc.). Due to Microsoft-specific mechanisms, this may cause the call to freeze and the program to crash.

Required files

rpjapi.h
rpjapi.lib
rpjapi.dll

Related functions

RPJAttach (Page 2079)	Establish connection

See also

RPJAttach (Page 2079)

Overview of the functions (Page 2067)

3.7.2.3 RPJMemFree (RT Professional)**Description**

This function is used to close an established connection to RPJAPI.DLL and release the allocated memory.

Declaration

```
BOOL RPJMemFree (
    const PVOID    pvMemBlock,
    CMN_ERROR*     pcmnerror );
```

Parameters**pvMemBlock**

Pointer to a released memory area that was previously allocated by a RPJ function.

If pvMemBlock = NULL, nothing happens.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The memory was released.

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

See also

Overview of the functions (Page 2067)

3.7.3 Functions for processing project properties (RT Professional)

3.7.3.1 RPJGetNumProjectProperties (RT Professional)

Description

The function determines the number of properties for the project.

Declaration

```
BOOL RPJGetNumProjectProperties (
    LPCSTR      pszReserved,
    DWORD*      pdwNumProperties
    CMN_ERROR*   pcmnerror );
```

Parameters

pszReserved

The parameter is reserved for future development.

pdwNumProperties

Pointer to a DWORD tag, in which the number of properties is returned.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

The number of properties was determined.

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

See also

Overview of the functions (Page 2067)

3.7.3.2 RPJGetPropertyAt (RT Professional)

Description

The function determines the name and type of a property based on the index in the project.

Declaration

```
BOOL RPJGetPropertyAt (  
    LPCSTR      pszReserved,  
    DWORD       dwPropIndex,  
    LPSTR       pszBuffer,  
    DWORD       dwCharMax,  
    DWORD*      pdwPrpType,  
    CMN_ERROR*   pcmnerror );
```

Parameters

pszReserved

The parameter is reserved for future expansions.

dwPropIndex

Index of the property in the project, from which the information is to be returned.

pszBuffer

Pointer to a buffer in which the name of the property should be returned. The size of the buffer is specified with dwCharMax.

dwCharMax

Maximum size of the buffer specified with pszBuffer.

pdwPropType

Pointer to a DWORD tag, in which the type of the property in the project is returned.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

The name and type of property in the project was determined.

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

See also

Overview of the functions (Page 2067)

3.7.3.3 RPJGetProjectProperty (RT Professional)

Description

The function determines the value of a property. The project name and the property name must be passed.

Declaration

```
BOOL RPJGetProjectProperty (  
    LPCSTR      pszProjectName,  
    LPCSTR      pszPropName,  
    PVOID       pvPropValue,  
    VARTYPE     vtPropType,  
    DWORD       dwBufferSize,  
    CMN_ERROR*  pcmnerror );
```

Parameters

pszProjectName

Pointer to the name of the project, consisting of the path, project name, and extension.

pszPropName

Name of the property in the project.

pvPropValue

Pointer to a buffer in which the value of the property should be returned. The size of the buffer in pvPropValue is specified with dwBufferSize.

vtPropType

Type in which the value of the property in the pvPropValue buffer is expected.

VT_I4	Integer or Boolean
VT_LPSTR	Text in ANSI format
VT_LPWSTR	Text in UNICODE format
VT_DATE	Date/time

dwBufferSize

Size of the buffer in pvPropValue to which the value of the property should be returned. The size may only be specified in BYTE and not in CHAR.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The value of property was determined.

FALSE

Error

Required files

rpjapi.h
rpjapi.lib
rpjapi.dll

See also

Overview of the functions (Page 2067)

3.7.3.4 RPJProjectLock (RT Professional)**Description**

Access to the print job list in the project is locked to others. The name of the locking party must be specified in pszLockerNameNew. If an access conflict occurs, the name of the locking party is returned via ppszLockerNameCur.

Declaration

```

BOOL RPJProjectLock (
    LPCSTR      pszProjectName,
    BOOL        fWriteLock,
    BOOL        fDoNotWait,
    LPCSTR      pszLockerNameNew,
    LPSTR*      ppszLockerNameCur,
    CMN_ERROR*  pcmnerror );

```

Parameters

pszProjectName

Pointer to the name of the project, consisting of the path, project name, and extension.

fWriteLock

Specifies the type of locking.

TRUE	Locks writing and reading.
FALSE	Locks reading only.

fDoNotWait

TRUE	An error is immediately reported if a lock exists.
FALSE	If a lock exists, it is checked cyclically at first. After a few seconds, the check stops with a timeout error.

pszLockerNameNew

Contains the name of the locking party. If an access conflict occurs, the name is returned via ppszLockerNameCur.

ppszLockerNameCur

If an access conflict occurs, the name of the current locking party is returned in ppszLockerNameCur.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Print job list is locked.

FALSE

An access conflict has occurred. The print job list of the project is already locked. The name of the locking party is returned via ppszLockerNameCur.

Required files

rpjapi.h
rpjapi.lib
rpjapi.dll

Related functions

RPJProjectUnlock (Page 2089)	Removing a lock
RPJProjectUnlockAll (Page 2090)	Removing all locks

See also

RPJProjectUnlock (Page 2089)
RPJProjectUnlockAll (Page 2090)
Overview of the functions (Page 2067)

3.7.3.5 RPJProjectUnlock (RT Professional)

Description

The lock of a print job list is removed.

Declaration

```
BOOL RPJProjectUnlock (  
    LPCSTR      pszProjectName,  
    CMN_ERROR*  pcmnerror );
```

Parameters

pszProjectName

Pointer to the name of the project file consisting of the path, project name and extension.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

The lock of the print job list was removed.

FALSE

An access conflict has occurred.

Required files

- rpjapi.h
- rpjapi.lib
- rpjapi.dll

Related functions

RPJProjectLock (Page 2085)	Creating a lock
RPJProjectUnlockAll (Page 2090)	Removing all locks

See also

- RPJProjectLock (Page 2085)
- RPJProjectUnlockAll (Page 2090)
- Overview of the functions (Page 2067)

3.7.3.6 RPJProjectUnlockAll (RT Professional)

Description

All locks of a print job list are removed.

Declaration

```
BOOL RPJProjectUnlockAll (  
    LPCSTR      pszProjectName,  
    CMN_ERROR*  pcmnerror );
```

Parameters

pszProjectName

Pointer to the name of the project file consisting of the path, project name and extension.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

All locks of the print job list were removed.

FALSE

An access conflict has occurred.

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJProjectLock (Page 2085)	Creating a lock
RPJProjectUnlock (Page 2087)	Removing a lock

See also

RPJProjectLock (Page 2085)

RPJProjectUnlock (Page 2087)

Overview of the functions (Page 2067)

3.7.4 Functions for processing print jobs (RT Professional)**3.7.4.1 RPJCreateJob (RT Professional)****Description**

Creates a new print job.

Declaration

```

BOOL RPJCreateJob (
    LPCSTR      pszProjectName,
    LPCSTR      pszJobName,
    CMN_ERROR*   pcmnerror );

```

Parameters

- pszProjectName**
Pointer to the name of the project file consisting of the path, project name and extension.
- pszJobName**
Pointer to the name of the print job to be created.
- pcmnerror**
Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

- TRUE**
Print job created.
- FALSE**
Error

Required files

- rpjapi.h
rpjapi.lib
rpjapi.dll

Related functions

RPJDeleteJob (Page 2092)	Delete print job

See also

- RPJDeleteJob (Page 2092)
Overview of the functions (Page 2067)

3.7.4.2 RPJDeleteJob (RT Professional)

Description

A print job specified by pszJobName is deleted.

Declaration

```

BOOL RPJDeleteJob (
    LPCSTR      pszProjectName,
    LPCSTR      pszJobName,
    CMN_ERROR*   pcmnerror );

```

Parameters

pszProjectName

Pointer to the name of the project file consisting of the path, project name and extension.

pszJobName

Pointer to the name of the print job to be deleted.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Print job deleted.

FALSE

Error

Required files

rpjapi.h
rpjapi.lib
rpjapi.dll

Related functions

RPJCreateJob (Page 2089)	Create a print job

See also

RPJCreateJob (Page 2089)
Overview of the functions (Page 2067)

3.7.4.3 RPJCreatePropertyHandle (RT Professional)

Description

To edit the properties of a print job, you first have to configure a handle on the job using this function. You get the handle in the return value.

Declaration

```
HPROPERTIES RPJCreatePropertyHandle (  
    LPCSTR      pszProjectName,  
    CMN_ERROR*  pcmnerror );
```

Parameters

pszProjectName

Pointer to the name of the project file consisting of the path, project name and extension.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

Handle on the print job property.

If an error occurs, the return value is ZERO.

Comment

The handle provided here is required by the RPJGetJobProps, RPJGetProperty and RPJSetProperty functions, for example.

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJDeletePropertyHandle (Page 2095)	Delete handle
RPJPropertyClear (Page 2114)	Initialize memory area
RPJGetJobProps (Page 2109)	Determine print job properties

RPJGetProperty (Page 2111)	Determine print job property
RPJSetProperty (Page 2115)	Specify print job properties

Examples

Show print job preview (Page 2133) "RD02.cpp"
 Modify print job properties (Page 2128) "RD02.cpp"
 Get print job names (Page 2122) "RD02.cpp"
 Get print job method name (Page 2119) "RD02.cpp"
 Get print job properties (Page 2125) "RD02.cpp"

See also

Get print job method name (Page 2119)
 Get print job names (Page 2122)
 Get print job properties (Page 2125)
 Modify print job properties (Page 2128)
 Show print job preview (Page 2133)
 RPJDeletePropertyHandle (Page 2095)
 RPJGetProperty (Page 2111)
 RPJGetJobProps (Page 2109)
 RPJSetProperty (Page 2115)
 RPJPropertyClear (Page 2114)
 Overview of the functions (Page 2067)

3.7.4.4 RPJDeletePropertyHandle (RT Professional)

Description

Each handle created with RPJCreatePropertyHandle, is deleted with this function when it is no longer needed.

Declaration

```
BOOL RPJDeletePropertyHandle (
    HPROPERTIES    hproperties,
    CMN_ERROR*     pcmnerror );
```

Parameters

hproperties

The handle of the print job property is created by the RPJCreatePropertyHandle function.

pcmncerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Handle deleted

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJCreatePropertyHandle (Page 2092)	Create handle

Examples

Show print job preview (Page 2133) "RD02.cpp"

Modify print job properties (Page 2128) "RD02.cpp"

Get print job names (Page 2122) "RD02.cpp"

Get print job method name (Page 2119) "RD02.cpp"

Get print job properties (Page 2125) "RD02.cpp"

See also

RPJCreatePropertyHandle (Page 2092)

Get print job method name (Page 2119)

Get print job names (Page 2122)

Get print job properties (Page 2125)

Modify print job properties (Page 2128)

Show print job preview (Page 2133)

Overview of the functions (Page 2067)

3.7.4.5 RPJGetJobNameAt (RT Professional)

Description

Determines the name of a print job specified by dwJobIndex.

Declaration

```

BOOL RPJGetJobNameAt (
    LPCSTR      pszProjectName,
    DWORD       dwJobIndex,
    LPSTR       pszBuffer,
    DWORD       dwCharMax,
    CMN_ERROR*  pcmnerror );

```

Parameters

PszProjectName

Pointer to the name of the project file consisting of the path, project name and extension.

dwJobIndex

Index of the print job, the name of which should be determined.

pszBuffer

Pointer to a receive buffer in which the name of the print job should be returned.

dwCharMax

Size of the receive buffer in number of characters including the zero termination. This value must be greater than 1.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Name of the print job determined.

FALSE

Error

Comment

An index error indicated by RPJGetJobNameAt may be due to the fact that the number of jobs changed after the RPJGetNumJobs was called.

Required files

rpjapi.h
rpjapi.lib
rpjapi.dll

Examples

Get print job names (Page 2122) "RD02.cpp"

See also

Get print job names (Page 2122)
Overview of the functions (Page 2067)

3.7.4.6 RPJGetNumJobs (RT Professional)

Description

Determines the number of currently pending print jobs. After the function is called, the number may change due to deletion or addition of print jobs.

Declaration

```
BOOL RPJGetNumJobs (  
    LPCSTR      pszProjectName,  
    DWORD*      pdwNumJobs,  
    CMN_ERROR*   pcmnerror );
```

Parameters

pszProjectName

Pointer to the name of the project file consisting of the path, project name and extension.

pdwNumJobs

Pointer to the location where the number of print jobs should be stored.

pcmerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Number of print jobs determined.

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Examples

Get print job names (Page 2122) "RD02.cpp"

See also

Get print job names (Page 2122)

Overview of the functions (Page 2067)

3.7.4.7 RPJJobLock (RT Professional)**Description**

Access to the specified print job is locked to others. The name of the locking party must be specified in pszLockerNameNew. If an access conflict occurs, the name of the locking party is returned via ppszLockerNameCur.

Declaration

```

BOOL RPJJobLock (
    LPCSTR      pszProjectName,
    LPCSTR      pszJobName,
    BOOL        fWriteLock,
    BOOL        fDoNotWait,
    LPCSTR      pszLockerNameNew,
    LPSTR*      ppszLockerNameCur,
    CMN_ERROR*  pcmnerror );

```

Parameters

pszProjectName

Pointer to the name of the project, consisting of the path, project name, and extension.

pszJobName

Pointer to the name of the print job to be locked.

fWriteLock

Specifies the type of locking.

TRUE	Locks writing and reading.
FALSE	Locks reading only.

fDoNotWait

TRUE	An error is immediately reported if a lock exists.
FALSE	If a lock exists, it is checked cyclically at first. After a few seconds, the check stops with a timeout error.

pszLockerNameNew

Contains the name of the locking party. If an access conflict occurs, the name is returned via ppszLockerNameCur.

ppszLockerNameCur

If an access conflict occurs, the name of the current locking party is returned in ppszLockerNameCur.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Print job is locked.

FALSE

An access conflict has occurred. The print job list of the project is already locked. The name of the locking party is returned via ppszLockerNameCur.

Required files

rpjapi.h
rpjapi.lib
rpjapi.dll

Related functions

RPJJobUnlock (Page 2101)	Removing a lock
RPJJobUnlockAll (Page 2102)	Removing all locks

See also

RPJJobUnlockAll (Page 2102)
RPJJobUnlock (Page 2101)
Overview of the functions (Page 2067)

3.7.4.8 RPJJobUnlock (RT Professional)**Description**

The lock of a print job is removed.

Declaration

```

BOOL RPJJobUnlock (
    LPCSTR      pszProjectName,
    LPCSTR      pszJobName,
    CMN_ERROR*  pcmnerror );

```

Parameters**pszProjectName**

Pointer to the name of the project file consisting of the path, project name and extension.

pszJobName

Pointer to the name of the print job.

pcmnnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The lock of the print job was removed.

FALSE

An access conflict has occurred.

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJJobLock (Page 2097)	Creating a lock
RPJJobUnlockAll (Page 2102)	Removing all locks

See also

RPJJobUnlockAll (Page 2102)

RPJJobLock (Page 2097)

Overview of the functions (Page 2067)

3.7.4.9 RPJJobUnlockAll (RT Professional)**Description**

All locks of a print job are removed.

Declaration

```
BOOL RPJJobUnlockAll (  
    LPCSTR      pszProjectName,  
    LPCSTR      pszJobName,  
    CMN_ERROR*  pcmnnerror );
```

Parameters

pszProjectName

Pointer to the name of the project file consisting of the path, project name and extension.

pszJobName

Pointer to the name of the print job.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

All locks of the specified print job are removed.

FALSE

An access conflict has occurred

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJJobLock (Page 2097)	Creating a lock
RPJJobUnlock (Page 2099)	Removing a lock

See also

RPJJobUnlock (Page 2099)

RPJJobLock (Page 2097)

Overview of the functions (Page 2067)

3.7.5 Functions for processing print job methods (RT Professional)

3.7.5.1 RPJCallJobMethod (RT Professional)

Description

This function allows you to specify a specific job method. Currently, there are two methods: PrintJob and PreviewJob.

Declaration

```
RPJCallJobMethod (
    HPROPERTIES    hproperties,
    LPCSTR         pszMethodName
    CMN_ERROR*     pcmnerror );
```

Parameters

hproperties

The handle of the print job property is created by the RPJCreatePropertyHandle function.

pszMethodName

Pointer to the name of the job method to be used.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Print job method specified

FALSE

Error

Comment

In order to specify a print job method with RPJCallJobMethod, you first need to execute the RPJGetJobProps function.

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Examples

Show print job preview (Page 2133) "RD02.cpp"

See also

Show print job preview (Page 2133)

Overview of the functions (Page 2067)

3.7.5.2 RPJGetJobMethodAt (RT Professional)

Description

The function determines the name of the printing method specified in dwMethodIndex.

Declaration

```
BOOL RPJGetJobMethodAt (  
    DWORD          dwMethodIndex,  
    LPSTR          pszBuffer,  
    DWORD          dwCharMax,  
    CMN_ERROR*     pcmnerror );
```

Parameters

dwMethodIndex

Index of the printing method whose name is determined.

pszBuffer

Pointer to a buffer in which the name of the printing method should be returned.

dwCharMax

Size of the buffer for receiving the name in number of characters including the zero termination. This value must be greater than 1.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Name of the print job method determined

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Examples

Get print job method name (Page 2119) "RD02.cpp"

See also

Get print job method name (Page 2119)

Overview of the functions (Page 2067)

3.7.5.3 RPJGetNumJobMethods (RT Professional)

Description

The function determines how many print job methods are currently available.

Declaration

```
BOOL RPJGetNumJobMethods (  
    DWORD* pdwNumMethods,  
    CMN_ERROR* pcmnerror );
```

Parameters

pdwNumMethods

Pointer to the location where the number of print job methods should be stored.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Number of print job methods determined

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Examples

Get print job method name (Page 2119) "RD02.cpp"

See also

Get print job method name (Page 2119)

Overview of the functions (Page 2067)

3.7.6 Functions for processing print job methods (RT Professional)

3.7.6.1 RPJGetJobPropertyAt (RT Professional)

Description

Determines the name and type of job property specified in dwPropindex.

Declaration

```
BOOL RPJGetJobPropertyAt (
    DWORD          dwPropIndex,
    LPSTR          pszBuffer,
    DWORD          dwCharMax,
    DWORD*         pdwPropType,
    CMN_ERROR*     pcmnerror );
```

Parameters

dwPropIndex

Index of the print job property, the name of which should be determined.

pszBuffer

Pointer to a buffer that receives the name of the job property.

dwCharMax

Size of the receive buffer in number of characters including the zero termination. This value must be greater than 1.

pdwPropType

Pointer to the location where the type of job property should be stored.

pcmerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Name of the print job property determined

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Examples

Get print job properties (Page 2125) "RD02.cpp"

See also

Get print job properties (Page 2125)

Overview of the functions (Page 2067)

3.7.6.2 RPJGetJobProps (RT Professional)

Description

The function determines the properties of a print job and stores it in the internal hProps buffer.

Declaration

```
BOOL RPJGetJobProps (  
    HPROPERTIES    hproperties,  
    LPCSTR         pszJobName,  
    CMN_ERROR*     pcmnerror );
```

Parameters

hproperties

The handle of the print job property is created by the RPJCreatePropertyHandle function.

pszJobName

Pointer to the name of the print job.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Comment

This function is a prerequisite for functions such as RPJGetProperty and RPJSetProperty.

Return value

TRUE

Print job properties determined

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJCreatePropertyHandle (Page 2092)	Create handle
RPJGetProperty (Page 2111)	Determine print job property
RPJSetProperty (Page 2115)	Specify print job properties

Examples

Show print job preview (Page 2133) "RD02.cpp"

Modify print job properties (Page 2128) "RD02.cpp"

See also

RPJGetProperty (Page 2111)

RPJSetProperty (Page 2115)

Modify print job properties (Page 2128)

Show print job preview (Page 2133)

RPJCreatePropertyHandle (Page 2092)

Overview of the functions (Page 2067)

3.7.6.3 RPJGetNumJobProperties (RT Professional)**Description**

Determines the number of available print job properties.

Declaration

```

BOOL RPJGetNumJobProperties (
    DWORD*          pdwNumProperties,
    CMN_ERROR*      pcmnerror );

```

Parameters**pdwNumProperties**

Pointer to the location where the number of print job properties should be stored.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Number of print job properties determined

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Examples

Get print job properties (Page 2125) "RD02.cpp"

See also

Get print job properties (Page 2125)

Overview of the functions (Page 2067)

3.7.6.4 RPJGetProperty (RT Professional)**Description**

Before you can use RPJGetProperty to determine the value of the print job property specified by pszPropName, you first need to use RPJGetJobProps to store the job properties in the internal hProp buffer.

Declaration

```
BOOL RPJGetProperty (
    HPROPERTIES    hproperties,
    LPCSTR         pszPropName,
    PVOID          pvPropValue,
    VARTYPE        vtPropType,
    DWORD          dwBufferSize,
    CMN_ERROR*     pcmnerror );
```

Parameters

hproperties

The handle of the print job property is created by the RPJCreatePropertyHandle function.

pszPropName

Pointer to the name of the print job property:

AbsoluteSelectionFrom	
AbsoluteSelectionTo	
CycleSpan	
DestinationFile	
EnableCycle	
EnableStart	
EndPage	This information enables you can to stop the printout of the report at a particular page.
JobName	The name of the print job must be unique within a project and conform to the Windows conventions.
LayoutName	You can assign the print job a layout using the layout name.
PrinterName 1	Here, you specify the printer to which the printout is initially sent.
PrinterName 2	This specifies the printer to be used if the first printer is not available.
PrinterName 3	This specifies the printer to be used if the first two printers are not available.
RelativeSelectionCount	
RelativeSelectionRange	
StartPage	This information enables you can to start the printout of the report at a particular page.
StartTime	
UseRelative	
UseOutputFile	
UseOutputPrinter	

pvPropValue

Pointer to a buffer that should receive the name.

vtPropType

The type of the object property depends on pszPropName:

AbsoluteSelectionFrom	VT_DATE
AbsoluteSelectionTo	VT_DATE
CycleSpan	VT_I4
DestinationFile	VT_LPSTR
EnableCycle	VT_I4
EnableStart	VT_I4
EndPage	VT_I4
JobName	VT_LPSTR / VT_LPWSTR

LayoutName	VT_LPSTR / VT_LPWSTR
PrinterName	VT_LPSTR / VT_LPWSTR
PrinterName2	VT_LPSTR / VT_LPWSTR
PrinterName3	VT_LPSTR / VT_LPWSTR
RelativeSelectionCount	VT_I4
RelativeSelectionRange	VT_I4
StartPage	VT_I4
StartTime	VT_DATE
UseRelative	VT_I4
UseOutputFile	VT_I4
UseOutputPrinter	VT_I4

If you want to change a string and your application is compiled in Unicode, you must use vtProp VT_LPWSTR as the data type. Otherwise, vtProp has a VT_LPTSTR data type.

If vtProp has a VT_DATE data type, the prop parameter must refer to a SYSTEMTIME structure.

dwBufferSize

Size of the buffer to which prop refers in number of bytes.

pcmerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Print job property determined

FALSE

Error

Comment

If the buffer for receiving the job property is configured too small, which can happen with job name and layout name, for example, the buffer is filled but the function returns the ERR_UNKNOWNERROR error message.

Error messages

ERR_UNKNOWNERROR	Unknown error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJCreatePropertyHandle (Page 2092)	Create handle
RPJGetJobProps (Page 2107)	Determine print job properties
RPJSetProperty (Page 2115)	Specify print job properties

Examples

Modify print job properties (Page 2128) "RD02.cpp"

See also

RPJGetJobProps (Page 2107)
 RPJSetProperty (Page 2115)
 RPJCreatePropertyHandle (Page 2092)
 Modify print job properties (Page 2128)
 Overview of the functions (Page 2067)

3.7.6.5 RPJPropertyClear (RT Professional)**Description**

This function is useful if you want to initialize a memory area referenced by a handle without deleting the handle.

Declaration

```

BOOL RPJPropertyClear (
    HPROPERTIES    hproperties,
    CMN_ERROR*     pcmnerror );

```

Parameters**hproperties**

The handle of the print job property is created by the RPJCreatePropertyHandle function.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Memory initialized.

FALSE

Error

Required files

rpjapi.h

rpjapi.lib

rpjapi.dll

Related functions

RPJCreatePropertyHandle (Page 2092)	Create handle

See also

RPJCreatePropertyHandle (Page 2092)

Overview of the functions (Page 2067)

3.7.6.6 RPJSetProperty (RT Professional)**Description**

This function allows you to temporarily change individual print job properties.

Before you can use RPJSetProperty to reset the value of the print job property specified by pszPropName, you first need to use RPJGetJobProps to store the job properties in the internal hProp buffer.

Declaration

```

BOOL RPJSetProperty (
    HPROPERTIES      hproperties,
    LPCSTR           pszPropName,
    PVOID            pvPropValue,
    VARTYPE          vtPropType,
    DWORD            dwReserved,
    CMN_ERROR*       pcmnerror );

```

Parameters

hproperties

Handle of the print job properties. The handle is created by the RPJCreatePropertyHandle function.

pszPropName

Pointer to the name of the print job property:

AbsoluteSelectionFrom	
AbsoluteSelectionTo	
CycleSpan	
DestinationFile	
EnableCycle	
EnableStart	
EndPage	This information enables you can to stop the printout of the report at a particular page.
JobName	The name of the print job must be unique within a project and conform to the Windows conventions.
LayoutName	You can assign the print job a layout using the layout name.
PrinterName 1	Here, you specify the printer to which the printout is initially sent.
PrinterName 2	This specifies the printer to be used if the first printer is not available.
PrinterName 3	This specifies the printer to be used if the first two printers are not available.
RelativeSelectionCount	
RelativeSelectionRange	
StartPage	This information enables you can to start the printout of the report at a particular page.
StartTime	
UseRelative	
UseOutputFile	
UseOutputPrinter	

pvPropValue

Pointer on the value of the object property.

vtPropType

The type of the object property depends on pszPropName:

AbsoluteSelectionFrom	VT_DATE
AbsoluteSelectionTo	VT_DATE
CycleSpan	VT_I4
DestinationFile	VT_LPSTR
EnableCycle	VT_I4
EnableStart	VT_I4
EndPage	VT_I4

JobName	VT_LPSTR / VT_LPWSTR
LayoutName	VT_LPSTR / VT_LPWSTR
PrinterName	VT_LPSTR / VT_LPWSTR
PrinterName2	VT_LPSTR / VT_LPWSTR
PrinterName3	VT_LPSTR / VT_LPWSTR
RelativeSelectionCount	VT_I4
RelativeSelectionRange	VT_I4
StartPage	VT_I4
StartTime	VT_DATE
UseRelative	VT_I4
UseOutputFile	VT_I4
UseOutputPrinter	VT_I4

If you want to change a string and your application is compiled in Unicode, you must use vtProp VT_LPWSTR as the data type. Otherwise, vtProp has a VT_LPTSTR data type.

If vtProp has a VT_DATE data type, the prop parameter must refer to a SYSTEMTIME structure.

dwReserved

The parameter is reserved for future expansions.

pcmnerror

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Print job property temporarily changed

FALSE

Error

Comment

All available print job properties can be determined with the RPJGetNumJobProperties and RPJGetJobPropertyAt functions.

Required files

rpjapi.h
rpjapi.lib
rpjapi.dll

Related functions

RPJCreatePropertyHandle (Page 2092)	Create handle
RPJGetJobProps (Page 2107)	Determine print job properties
RPJGetProperty (Page 2109)	Determine print job property

Examples

Modify print job properties (Page 2128) "RD02.cpp"

See also

RPJGetJobProps (Page 2107)
RPJGetProperty (Page 2109)
RPJCreatePropertyHandle (Page 2092)
Modify print job properties (Page 2128)
Overview of the functions (Page 2067)

3.7.7 Examples (RT Professional)

3.7.7.1 Get print job method name (RT Professional)

Example

```
//{{ODK_EXAMPLE}Get print job methodname (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetNumJobMethods (RPT)}
//{{FUNCTION}RPJGetJobMethodAt (RPT)}
//{{FUNCTION}(END)}
// =====
// Function: MyPrintJobPropertyInquire( void ) ODK RD RT
// =====
// : Get print job methodname (Enum) !
// =====
void MyRPJGetJobMethodAt(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[255];
    TCHAR szText[255];
    DWORD d = 0L;
    DWORD i = 0L;
    char buf[500+1];
    DWORD dwBufsize = 500L;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("-----"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyRPJGetJobMethodAt:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
```

3.7 Report system functions (RT Professional)

```

{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
    return;
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle = 0x
%08lx"), hProp);
    ODKTrace(szText);
}
memset(&Error, 0, sizeof(CMN_ERROR));
ret = RPJGetNumJobMethods(&d, &Error);
if (FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetNumJobMethods:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
}
else
{
    for (i = 0; i < d; i++)
    {
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = RPJGetJobMethodAt(i, buf, dwBufsize, &Error);
        if (ret == FALSE)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    %d PrintJobMethod=
%s"), i+1, buf);
        }
        ODKTrace(szText);
    }
}
memset(&Error, 0, sizeof(CMN_ERROR));
ret = RPJDeletePropertyHandle(hProp, &Error);

```

```

    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE} (END) }

```

See also

[RPJCreatePropertyHandle \(Page 2092\)](#)
[RPJDeletePropertyHandle \(Page 2093\)](#)
[RPJGetJobMethodAt \(Page 2103\)](#)
[RPJGetNumJobMethods \(Page 2104\)](#)

3.7.7.2 Get print job names (RT Professional)

Example

```

//{{ODK_EXAMPLE}Get print job names (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetNumJobs (RPT)}
//{{FUNCTION}RPJGetJobNameAt (RPT)}
//{{FUNCTION}(END)}
// =====
// Function: MyRPJGetJobNameAt( void ) ODK RD RT
// =====
// : Get all print job names (Enum) !
// =====
void MyRPJGetJobNameAt(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[255];
    TCHAR szText[255];
    DWORD d = 0L;
    DWORD i = 0L;
    char buf[500+1];
    DWORD dwBufsize = 500;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("-----"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyRPJGetJobNameAt:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),

```



```

        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
    return;
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle = 0x
%08lx"), hProp);
    ODKTrace(szText);
}
memset(&Error,0,sizeof(CMN_ERROR));
ret = RPJGetNumJobs(/*PROJ_PATH*/g_szProjectFile, &d, &Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetNumJobs: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
}
else
{
    for (i = 0; i < d; i++)
    {
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = FALSE;
        ret = RPJGetJobNameAt(/*PROJ_PATH*/g_szProjectFile, i, buf, dwBufsize, &Error);
        if (ret == FALSE)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    %d PrintJobName=
%s"),i+1, buf);
        }
        ODKTrace(szText);
    }
}
memset(&Error,0,sizeof(CMN_ERROR));
ret = FALSE;
ret = RPJDeletePropertyHandle(hProp, &Error);
if (ret == FALSE)

```

3.7 Report system functions (RT Professional)

```

{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
}
ODKTrace(szText);
memset(&Error, 0, sizeof(CMN_ERROR));
ret = RPJDetach(&Error);
if (FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
}
ODKTrace(szText);
}
//{{ODK_EXAMPLE} (END) }

```

See also

[RPJCreatePropertyHandle \(Page 2092\)](#)

[RPJDeletePropertyHandle \(Page 2093\)](#)

[RPJGetJobNameAt \(Page 2095\)](#)

[RPJGetNumJobs \(Page 2096\)](#)

3.7.7.3 Get print job properties (RT Professional)

Example

```
//{{ODK_EXAMPLE}Get print job properties (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetNumJobProperties (RPT)}
//{{FUNCTION}RPJGetJobPropertyAt (RPT)}
//{{FUNCTION}(END)}
// =====
// Function: MyRPJGetJobPropertyAt( void ) ODK RD RT
// =====
// : Inquire print job properties (Enum) !
// =====
void MyRPJGetJobPropertyAt(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[255];
    TCHAR szText[255];
    DWORD d = 0L;
    DWORD f = 0L;
    DWORD i = 0L;
    char buf[500+1];
    DWORD dwBufsize = 500L;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("-----"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyRPJGetJobPropertyAt:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {
```

3.7 Report system functions (RT Professional)

```

        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle = 0x
%08lx"), hProp);
        ODKTrace(szText);
    }
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJGetNumJobProperties(&d, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJGetNumJobProperties: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        for (i = 0; i < d; i++)
        {
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = RPJGetJobPropertyAt(i, buf, dwBufsize, &f, &Error);
            if (FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJGetJobPropertyAt: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    %d PrintJobProperty=
%s"), i+1, buf);
            }
            ODKTrace(szText);
        }
    }
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJDeletePropertyHandle(hProp, &Error);
    if (FALSE == ret)

```

```

{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
}
ODKTrace(szText);
memset(&Error, 0, sizeof(CMN_ERROR));
ret = RPJDetach(&Error);
if (FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
}
ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}

```

See also

[RPJCreatePropertyHandle \(Page 2092\)](#)
[RPJDeletePropertyHandle \(Page 2093\)](#)
[RPJGetJobPropertyAt \(Page 2105\)](#)
[RPJGetNumJobProperties \(Page 2108\)](#)

3.7.7.4 Modify print job properties (RT Professional)

Example

```

//{{ODK_EXAMPLE}Modify print job properties (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetJobProps (RPT)}
//{{FUNCTION}RPJGetProperty (RPT)}
//{{FUNCTION}RPJSetJobProps (RPT)}
//{{FUNCTION}RPJSetProperty (RPT)}
//{{FUNCTION}(END)}
// =====
// Function: MyModifyPrintJob( void ) ODK RD RT
// =====
// : Modify print job properties (change starttime) !
// =====
void MyModifyPrintJob(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    TCHAR szText[255];
    HPROPERTIES hProp = NULL;
    SYSTEMTIME st;
    LPVOID ptr;
    DWORD typ;
    TCHAR jobname[200];
    TCHAR propname[200];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("-----"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyModifyPrintJob:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
    // read properties
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {

```

```

        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle = 0x
%08lx"), hProp);
        ODKTrace(szText);
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJGetJobProps (hProp, jobname, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDeletePropertyHandle (hProp, &Error);
        if(ret == FALSE)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
        }
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
    }
}

```

3.7 Report system functions (RT Professional)

```

        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    RPJGetJobProps"));
        ODKTrace(szText);
    }
    typ = VT_DATE;
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE);
    _tcsncpy_s(propname, _countof(propname), _T("STARTTIME"), _TRUNCATE);
    memset(&st, 0, sizeof(SYSTEMTIME));
    ptr = (LPVOID)&st;
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJGetProperty(hProp, propname, ptr, (VARTYPE)typ, 16, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetProperty: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    RPJGetProperty"));
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    Jobname=%s Propname=
%s"), jobname, propname);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    %02d.%02d.%04d %02d:
%02d:%02d"),
            (WORD)st.wDay,
            (WORD)st.wMonth,
            (WORD)st.wYear,
            (WORD)st.wHour,
            (WORD)st.wMinute,
            (WORD)st.wSecond);
    }
    ODKTrace(szText);
    // write properties
    st.wHour = 11;
    st.wMinute = 12;
    st.wSecond = 13;
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJSetProperty(hProp, propname, ptr, (VARTYPE)typ, 16, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJSetProperty: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    RPJSetProperty"));
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    Jobname=%s ;Propname=
%s"), jobname, propname);
    }
}

```



```

        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    %02d.%02d.%04d %02d:
%02d:%02d"),
            (WORD)st.wDay,
            (WORD)st.wMonth,
            (WORD)st.wYear,
            (WORD)st.wHour,
            (WORD)st.wMinute,
            (WORD)st.wSecond);
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJSetJobProps (hProp, jobname, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJSetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    RPJSetJobProps"));
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDeletePropertyHandle (hProp, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE,
_T("RPJDeletePropertyHandle"));    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE} (END) }

```

See also

RPJCreatePropertyHandle (Page 2092)

RPJDeletePropertyHandle (Page 2093)

RPJGetJobProps (Page 2107)

RPJGetProperty (Page 2109)

RPJSetProperty (Page 2113)

3.7.7.5 Show print job preview (RT Professional)

Example

```
// =====
// =====
// : Modul with examples of Report Designer
// *****
// Copyright (C) 1995-99 SIEMENS AG, A&D PT1 D2 All rights reserved
// *****
#include "stdafx.h" // if MFC classes
#include "RD02.h"
#include "DM01.h"

//{{ODK_EXAMPLE}Show print job preview (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetJobProps (RPT)}
//{{FUNCTION}RPJCallJobMethod (RPT)}
//{{FUNCTION}(END)}
// =====
// Function: MyShowPrintJobPreview( void ) ODK RD RT
// =====
// : Show print job preview
// =====
void MyShowPrintJobPreview(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    TCHAR szText[255];
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[200];
    TCHAR methode[200];
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("-----"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyShowPrintJobPreview:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
}
```

3.7 Report system functions (RT Professional)

```

}
memset(&Error, 0, sizeof(CMN_ERROR));
hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
if(NULL == hProp)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    ret = RPJDetach(&Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
    return;
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle = 0x
%08lx"), hProp);
    ODKTrace(szText);
}
memset(&Error, 0, sizeof(CMN_ERROR));
ret = RPJGetJobProps (hProp, jobname, &Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    ret = RPJDeletePropertyHandle (hProp, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
    ret = RPJDetach(&Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
}

```

```

        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    RPJGetJobProps"));
        ODKTrace(szText);
    }
    _tcsncpy_s(methode, _countof(methode), _T("PREVIEW"), _TRUNCATE); // preview
    // _tcsncpy_s(methode, _countof(methode), _T("PRINTJOB"), _TRUNCATE); // print
    // _tcsncpy_s(methode, _countof(methode), _T("SETSELECTIONALLPAGES"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJCallJobMethod (hProp, methode, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    RPJCallJobMethod"));
    }
    ODKTrace(szText);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJDeletePropertyHandle (hProp, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}

```

See also

RPJCreatePropertyHandle (Page 2092)

RPJDeletePropertyHandle (Page 2093)

RPJCallJobMethod (Page 2102)

RPJGetJobProps (Page 2107)

3.8 Tag and log functions (RT Professional)**3.8.1 Basics (RT Professional)****3.8.1.1 Overview of the functions (RT Professional)****Overview**

TLG_ENUM_ARCHIV_CALLBACK (Page 2210)	List logs (callback)
TLG_ENUM_PROJECT_NAME_CALLBACK (Page 2201)	List projects (callback)
AUTOHOTSPOT	List entries in trend templates (callback)
TLG_ENUM_TIME_NAME_CALLBACK (Page 2247)	List time objects (callback)
TLG_ENUM_VARIABLE_NAME_CALLBACK (Page 2203)	List tags (callback)
TLG_ENUMBACKUP_ENTRIES (Page 2254)	List backups (callback)
TLG_ENUMTABLES (Page 2217)	List logs (callback)
TLG_ENUMTIMES_CALLBACK (Page 2250)	List time objects (callback)
TLG_ENUMVARIABLES (Page 2206)	List tags (callback)
TLG_GETARCHIVDATA_CALLBACK (Page 2222)	Read out data of a log (callback)
TLGChangeLanguage (Page 2192)	Language switching
TLGCloseProject (Page 2196)	Close project
TLGCloseWindow (Page 2237)	Close print job/script diagnostics
TLGConnect (Page 2193)	Establish connection to Tag Logging RT
TLGCSConnectEx (Page 2190)	Establish connection to Tag Logging CS
TLGCSConnect (Page 2188)	Establish connection to Tag Logging CS
TLGDisconnect (Page 2195)	Terminate connection to Tag Logging RT
TLGDrawCurvesInDC (Page 2238)	Show trends
TLGEnumArchives (Page 2209)	List logs
TLGEnumArchivsEx (Page 2214)	List logs
TLGEnumArchivs (Page 2212)	List logs
TLGEnumArchivsSel (Page 2215)	List logs

TLGEnumBackupEntries (Page 2252)	List backups
TLGEnumProject (Page 2200)	List projects
TLGEnumTime (Page 2245)	List time objects
TLGEnumTimes (Page 2248)	List time objects
TLGEnumVariablesEx (Page 2204)	List tags
TLGEnumVariables (Page 2202)	List tags
TLGExport (Page 2255)	Export log
TLGFreeMemory (Page 2219)	Free memory
TLGGetArchivDataEx (Page 2224)	Read out data of a log
TLGGetArchivData (Page 2220)	Read out data of a log
TLGGetBackupSize (Page 2257)	Determine size of an exported data record
TLGGetClosestTimeEx (Page 2228)	Determine logging time
TLGGetClosestTime (Page 2226)	Determine logging time
TLGInsertArchivData (Page 2230)	Insert data in a log
TLGInsertTemplateItem (Page 2240)	Write entries to trend window templates, write entries to table window templates
TLGLockArchiv (Page 2233)	Enable log, lock log
TLGLockVariable (Page 2234)	Enable tag, lock tag
TLGOpenProject (Page 2198)	Open project
TLGPressToolBarButton (Page 2241)	Activate buttons in the toolbar
TLGReadArchiv (Page 2236)	Read log parameters
TLGReadTime (Page 2251)	Determine time object parameters
TLGReadVariable (Page 2207)	Determine tag parameters
TLGSetRulerWindowVisible (Page 2243)	Also display reading line window
TLGShowWindow (Page 2244)	Display print job/script diagnostics

3.8.1.2 Overview of structures (RT Professional)

Overview

TLG_ARCHIV_STR (Page 2149)	Log parameters
TLG_ARCHIVDATARAW (Page 2152)	Log data
TLG_BACKUP_TABLE_INFO (Page 2155)	Information on export table
TLG_CURVESCALEX (Page 2157)	Scaling of the x-axis
TLG_CURVESCALEY (Page 2162)	Scaling of the y-axis
TLG_GETARCHIVDATA (Page 2166)	Log data of process value logs, log data of compressed logs
TLG_IO_BACKUP_SELECT (Page 2167)	Selection criteria for backup functions
TLG_PROT_CURVE_INFOS (Page 2168)	Report structure of trends
TLG_TABLE_INFO (Page 2171)	Report structure of tables
TLG_SCAL_STR (Page 2170)	Tag limits
TLG_TABLESCALE (Page 2172)	Table column (properties)
TLG_TEMPLATEITEM_INFO (Page 2175)	Trend (properties) Table column (properties)

3.8 Tag and log functions (RT Professional)

TLG_TIME_STR (Page 2177)	Time object
TLG_TIMEDATA (Page 2178)	Time object
TLG_TPLITEM_CURVE (Page 2179)	Trend (properties)
TLG_TPLITEM_INFO (Page 2180)	Trend (properties) Table column (properties)
TLG_TPLITEM_TABLE (Page 2181)	Table column (properties)
TLG_VAR_STR (Page 2182)	Tag (properties)
TLG_VARIABLE_INFO (Page 2187)	Tag

3.8.1.3 Error messages (RT Professional)

Overview

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

TLG_API_ERROR	0x10000000	Unspecified error
TLG_API_NO_TIME_EXIST	0x10000001	No time object exists
TLG_API_TIME_NAME_EXIST	0x10000002	A time object with this name already exists
TLG_API_TIME_NAME_NOT_EXIST	0x10000003	A time object with this name was not found
TLG_API_NO_ARCHIV_EXIST	0x10000004	No log available
TLG_API_ARCHIV_NAME_EXIST	0x10000005	A log with this name already exists
TLG_API_ARCHIV_NAME_NOT_EXIST	0x10000006	A log with this name was not found
TLG_API_INVALID_ARCHIV_TYP	0x10000007	Invalid log type
TLG_API_NO_VARIABLE_EXIST	0x10000008	No tags exist
TLG_API_VARIABLE_NAME_EXIST	0x10000009	A tag with this name already exists
TLG_API_VARIABLE_NAME_NOT_EXIST	0x1000000A	A tag with this name was not found
TLG_API_VARTYPE_MISMATCH	0x1000000B	Incorrect tag type
TLG_API_SRC_NO_VARIABLE_EXIST	0x1000000C	SRC: no tags exist
TLG_API_SRC_VARIABLE_NAME_NOT_EXIST	0x1000000D	SRC: a tag with this name was not found
TLG_API_SRC_ARCHIV_NOT_EXIST	0x1000000E	SRC: log not found
TLG_API_NO_FIELD_EXIST	0x1000000F	No fields are available
TLG_API_FIELD_NAME_EXIST	0x10000010	A field with this name already exists
TLG_API_FIELD_NAME_NOT_EXIST	0x10000011	A field with this name was not found
TLG_API_NO_PROJECT_EXIST	0x10000012	No project exists
TLG_API_PROJECT_NAME_NOT_EXIST	0x10000013	Project name was not found
TLG_API_PROJECT_IS_ACTIVE	0x20000014	Project is activated

TLG_API_NO_CONNECTION	0x10000015	No connection to an activated project
TLG_API_RENAME_NAME_EXIST	0x10000016	Assigned name already exists
TLG_API_NAME_WRONG_CHAR	0x10000017	Name contains incorrect character
TLG_API_NAME_TOLONG	0x10000018	Name is too long
TLG_API_ERR_ARCHIVSTYLE	0x10000019	Error in log design
TLG_API_NO_TYP_CHANGE_ALLOWED	0x1000001A	Type change is not allowed
TLG_API_ERR_SUPPLY	0x1000001B	An error has occurred
TLG_API_INVALID_PARAM	0x1000001C	Invalid/incorrect parameter assignment
TLG_API_NOT_SUPPORTED	0x1000001D	This function is not supported
TLG_API_NO_INTERFACE	0x1000001E	Access to lower level COM interfaces failed
TLG_API_NODEFAULTSERVER	0x10000021	No default server configured
TLG_API_NOLOCALSERVER	0x10000022	No local server available
TLG_API_NOSERVER	0x10000023	No default server configured and no local server available
TLG_API_NOMC	0x10000024	No multiclient project (not used here)
TLG_API_NOMCDEFAULTSERVER	0x10000025	No multiclient project, but "@default::" specified as server prefix (is not used here)
TLG_API_UNKNOWN_ERROR	0x10000030	Unknown callback error (WinCC V6.2 and higher)
TLG_API_ASYNC_TIMEOUT	0x10000031	Asynchronous function canceled with timeout (WinCC V6.2 and higher)
TLG_API_EXPORT_NO_WRITE	0x10000032	Export file was not written (WinCC V6.2 and higher)
TLG_API_CORRUPT_DATA	0x10000033	Data unusable/defective (WinCC V6.2 and higher)
IDS_API_ERROR_CREATING_CLIENT_WND	30001	Client window could not be generated
IDS_API_ERROR_WINDOW_NOT_FOUND	30002	Button (window) could not be found

3.8.1.4 Constants (RT Professional)

Runtime window

TLG_CURVE_WINDOW	0x00000001	Trend window
TLG_TABLE_WINDOW	0x00000002	Table window
TLG_CTRL_WINDOW	0x00000004	TLGRT control window

Export functions

TLG_BACKUP_EXPORT	0x00000001	Export without deletion of source data.
TLG_BACKUP_EVACUATE	0x00000002	Export with deletion of source data.
TLG_BACKUP_DELETE	0x00000004	Deletion of data records without export.
TLG_BACKUP_RAW	0x00000008	Do not handle log on change separately.

Export/import formats

TLG_BAKFMT_CSV	0x00000001	CSV format (comma separated value)

Import functions

TLG_RESTORE_OVERWRITE	0x00000001	Import with overwriting.
TLG_RESTORE_MERGE	0x00000002	Import with check, error via callback.
TLG_RESTORE_BREAK	0x00000004	Import - canceled due to error

Log types

TLG_ARCTYP_USER	0x00000001	User log (no longer supported in WinCC V5.0 and higher)
TLG_ARCTYP_PROCESS	0x00000002	Process value log
TLG_ARCTYP_COMPRESS	0x00000004	Compressed log

Log types

TLG_ARCTYP_ALL	0x00000000	Circular or sequential log
TLG_ARCTYP_CIRCULAR	0x00010000	Circular log
TLG_ARCTYP_FOLLOW	0x00020000	Sequential log

Log flags

TLG_API_FLG_FAST_INSERT	0x00000001	WinCC V6.0 SP1 and higher: Fast insertion of data in SQL Server with TLGInsertArchivData

Window templates

TLG_TEMPLATE_ALL	0x00000000	All window templates
TLG_TEMPLATE_CURVE	0x00000001	Trend window template
TLG_TEMPLATE_TABLE	0x00000002	Table window template
TLG_TYP_TEMPLATE_CURVE	0x00800001	Trend window template
TLG_TYP_TEMPLATE_TABLE	0x00800002	Table window template

Templates

TLG_TEMPLATEITEM_ALL	0x00000000	All template items
TLG_TEMPLATEITEM_CURVE	0x00000001	Trend template items
TLG_TEMPLATEITEM_TABLE	0x00000002	Table template items

Modify templates items

TLG_TI_ACTION_COLOR	0x00000001	Change color
TLG_TI_ACTION_SCALE_X	0x00000002	Change X-scaling
TLG_TI_ACTION_SCALE_Y	0x00000004	Change Y-scaling
TLG_TI_ACTION_VISIBLE	0x00000005	Make visible/invisible
TLG_TI_ACTION_RANGE_X	0x00000006	Display range X-axis
TLG_TI_ACTION_RANGE_Y	0x00000007	Display range Y-axis

Limit values

TLG_MAX_SQL_SELECT	512	Maximum length of an SQL command
TLG_MAX_TEMPLATE_NAME	32	Maximum length of the name of a trend or table window template
TLG_MAX_TEMPLATEITEM_NAME	32	Maximum length of the name of a trend or table template
TLG_MAX_FUNCTION_NAME	32	Maximum length of the name of an action or function
TLG_MAX_DLL_NAME	32	Maximum length of the name of a DLL
TLG_MAX_STD_TEXT_NAME	32	TLGSetTemplateData
TLG_SETDATA_DEFAULT	0x00000000	$y=f(x)$
TLG_SETDATA_RESET	0x00000001	The trend buffer is cleared.
TLG_SETDATA_TIME_RANGE_X	0x00000002	The x-values must be interpreted as a time range: $y=f(t)$.
TLG_SETDATA_TIME_RANGE_Y	0x00000004	The y-values must be interpreted as a time range: $x=f(t)$.

Trend shape

TLG_CURVEFORM_STEP	0x00000011	Representation of the tag values as a step trend
TLG_CURVEFORM_POINTS	0x00000012	Representation of the tag values as single points.
TLG_CURVEFORM_DIRECT	0x00000014	Linear interpolation of tag values
TLG_CURVEFORM_DIRFILL	0x00000021	
TLG_CURVEFORM_STEPFILL	0x00000022	

Data range types

TLG_DATATYP_TIMERANGE	0x00000001	
TLG_DATATYP_USERARCHIV	0x00000002	
TLG_DATATYP_BLOCKDATA	0x00000004	

Time ranges

TLG_TR_MINUTE	1	
TLG_TR_HOUR	2	
TLG_TR_DAY	3	
TLG_TR_WEEK	4	
TLG_TR_MONTH	5	
TLG_TR_YEAR	6	

Axes

TLG_AXIS_X_TOP	0x00000001	
TLG_AXIS_X_MIDDLE	0x00000002	
TLG_AXIS_X_BOTTOM	0x00000004	
TLG_AXIS_Y_LEFT	0x00000001	
TLG_AXIS_Y_MIDDLE	0x00000002	
TLG_AXIS_Y_RIGHT	0x00000004	

Protocol connection

TLG_PROTFLG_TIME_RANGE	0x00000001	
TLG_PROTFLG_DATA_RANGE	0x00000002	
TLG_PROTFLG_DATA	0x00000001	
TLG_PROTFLG_TITLE	0x00000002	
TLG_PROTFLG_TIMEFIELD	0x00000004	
TLG_PROTFLG_DATACOUNT	0x00000008	
TLG_PROTFLG_DATEFIELD	0x00000010	
TAG_PROT_CURVE_FORM_STEP	0x00000001	
TAG_PROT_CURVE_FORM_POINTS	0x00000002	
TAG_PROT_CURVE_FORM_DIRECT	0x00000003	
TAG_PROT_CURVE_FORM_AREA	0x00000004	
TLG_PROTFLG_FIRST_COL	0x00000000	
TLG_PROTFLG_FIRST_COL_TIME	0x00000001	
TLG_PROTFLG_FIRST_COL_DATA	0x00000002	
TLG_PROTFLG_FIRST_COL_DATE	0x00000004	

Read log data

TLG_DATA_FROM_POS	0x00000000	
TLG_DATA_FROM_BEGIN	0x00000001	
TLG_DATA_FROM_END	0x00000002	
TLG_DATA_FROM_ACTUAL	0x00000004	
TLG_DATA_FROM_LEFT_ABS	0x00000008	
TLG_DATA_FROM_RIGHT_ABS	0x00000010	

Buttons on the Runtime toolbar**Buttons for trends and tables**

TLG_BASIC_BTN_HELP	0x00000001	Identification of the button for calling on-line help.
TLG_BASIC_BTN_DLG	0x00000002	Identification of the button for opening the parameter assignment dialog for trend and table window templates.
TLG_BASIC_BTN_FIRST	0x00000004	Identification of the button for displaying the first data records of a log.
TLG_BASIC_BTN_PREV	0x00000008	Identification of the button for scrolling back in the log.
TLG_BASIC_BTN_NEXT	0x00000010	Identification of the button for scrolling ahead in the log.
TLG_BASIC_BTN_LAST	0x00000020	Identification of the button for displaying the last data records of a log.
TLG_BASIC_BTN_STARTSTOP	0x00000040	Identification of the button for pausing or resuming the updated trend or table display.
TLG_BASIC_BTN_PREV_ITEM	0x00000800	Previous trend in the foreground or table column first.
TLG_BASIC_BTN_NEXT_ITEM	0x00001000	Next trend in the foreground or table column first.
TLG_BASIC_BTN_ARC_VAR_SELECT	0x00000100	Opens the log tag selection dialog.
TLG_BASIC_BTN_ITEM_SELECT	0x00000200	Opens the selection dialog for trends or table columns.
TLG_BASIC_BTN_TIME_SELECT	0x00000400	Opens the selection dialog for the display time range.

Buttons for trends

TLG_CURVE_BTN_ZOOMIN	0x01000000	Identification of the button for activating the zoom function for the area to be zoomed. The size of the zoom segment can only be defined using the mouse!
TLG_CURVE_BTN_ZOOMOUT	0x02000000	Identification of the button for deactivating the zoom function. This does not result in switching back to the normal view.

3.8 Tag and log functions (RT Professional)

TLG_CURVE_BTN_LINEAL	0x04000000	Identification of the button for activating and deactivating the ruler for determining the coordinates of a measuring point.
TLG_CURVE_BTN_1_TO_1	0x08000000	Identification of the button for switching to normal trend view.

Buttons for tables

TLG_TABLE_BTN_EDIT	0x00040000	Identification of the button for activating and deactivating the editing function.

Field definitions for structures

PDE_DB_PRJ_PROJNAME_MAX_LENGTH TLG_DB_PRJ_PROJNAME_MAX_LENGTH	128	
PDE_DB_PRJ_COMMENT_MAX_LENGTH TLG_DB_PRJ_COMMENT_MAX_LENGTH	200	
PDE_DB_PRJ_USERDATE_MAX_LENGTH TLG_DB_PRJ_USERDATE_MAX_LENGTH	12	
PDE_DB_PRJ_USERNAME_MAX_LENGTH TLG_DB_PRJ_USERNAME_MAX_LENGTH	30	
PDE_DB_ARC_NAME_MAX_LENGTH TLG_DB_ARC_NAME_MAX_LENGTH	32	
PDE_DB_ARC_NAME_MAX_LENGTH TLG_DB_ARC_NAME_MAX_LENGTH	32	
TLG_DB_ARC_USER_NAME_MAX_LENGTH	8	
PDE_DB_ARC_SERVERNAME_MAX_LENGTH TLG_DB_ARC_SERVERNAME_MAX_LENGTH	50	
PDE_DB_ARC_COMMENT_MAX_LENGTH TLG_DB_ARC_COMMENT_MAX_LENGTH	100	
PDE_DB_ARC_ACTIONNAME_MAX_LENGTH TLG_DB_ARC_ACTIONNAME_MAX_LENGTH	30	
PDE_DB_VAR_NAME_MAX_LENGTH TLG_DB_VAR_NAME_MAX_LENGTH	64	
TLG_DB6_VAR_NAME_MAX_LENGTH	128	
TLG_DB_MCP_VAR_MAME_MAX_LENGTH	128	
PDE_DB_VAR_COMMENT_MAX_LENGTH TLG_DB_VAR_COMMENT_MAX_LENGTH	100	
PDE_DB_VAR_WRITEBACK_MAX_LENGTH TLG_DB_VAR_WRITEBACK_MAX_LENGTH	64	
TLG_DB6_VAR_WRITEBACK_MAX_LENGTH	128	
PDE_DB_VAR_CYCLENAME_MAX_LENGTH TLG_DB_VAR_CYCLENAME_MAX_LENGTH	30	
PDE_DB_VAR_SIGNALTEX_MAX_LENGTH TLG_DB_VAR_SIGNALTEX_MAX_LENGTH	30	

PDE_DB_VAR_DLLNAME_MAX_LENGTH TLG_DB_VAR_DLLNAME_MAX_LENGTH	30	
PDE_DB_VAR_FUNCNAME_MAX_LENGTH TLG_DB_VAR_FUNCNAME_MAX_LENGTH	30	
PDE_DB_VAR_UNITDIR_MAX_LENGTH TLG_DB_VAR_UNITDIR_MAX_LENGTH	30	
PDE_DB_VAR_UNITSTR_MAX_LENGTH TLG_DB_VAR_UNITSTR_MAX_LENGTH	64	
TLG_DB6_VAR_UNITSTR_MAX_LENGTH	128	
PDE_DB_FUNCNAME_MAX_LENGTH TLG_DB_FUNCNAME_MAX_LENGTH	30	
PDE_DB_DLLNAME_MAX_LENGTH TLG_DB_DLLNAME_MAX_LENGTH	30	
PDE_DB_SCALE_MAX_LENGTH TLG_DB_SCALE_MAX_LENGTH	30	
PDE_DB_TIMENAME_MAX_LENGTH TLG_DB_TIMENAME_MAX_LENGTH	30	
PDE_DB_FIELDNAME_MAX_LENGTH TLG_DB_FIELDNAME_MAX_LENGTH	64	
PDE_DB_STRUCTNAME_MAX_LENGTH TLG_DB_STRUCTNAME_MAX_LENGTH	30	

Field definitions of VARIANT record for log tag

For TLGCSCreateTagMulti, TLGCSReadTagMulti, and TLGCSModifyTagMulti

see Include file pde_typ.h

These functions are available starting with WinCC V6.2.

Definition in ENUM_TLG_TAG _FIELDINDEX	Field index	Associated type definition	Type	Comments/ limita- tions
ENUM_TLG_TAG _ENUMTYPE	0	VT_TLG_TAG_EN UMTYPE	VT_I4	Set internally (val- ue >= 8), cannot be changed
ENUM_TLG_TAG _VERSION	1	VT_TLG_TAG_VE RSION	VT_I4	Set internally (val- ue >= 1), cannot be changed
ENUM_TLG_TAG _CREATORID	2	VT_TLG_TAG_CR EATORID	VT_I4	
ENUM_TLG_TAG _NAME	3	VT_TLG_TAG_NA ME	VT_BSTR	Required for Cre- ate
ENUM_TLG_TAG _ARCHIVNAME	4	VT_TLG_TAG_AR CHIVNAME	VT_BSTR	Required for Cre- ate
ENUM_TLG_TAG _TYPE	5	VT_TLG_TAG_TY PE	VT_I4	Required for Cre- ate

3.8 Tag and log functions (RT Professional)

Definition in ENUM_TLG_TAG _FIELDINDEX	Field index	Associated type definition	Type	Comments/ limita- tions
ENUM_TLG_TAG _WRITEBACK- NAME	6	VT_TLG_TAG_W RITEBACKNAME	VT_BSTR	
ENUM_TLG_TAG _COMMENT	7	VT_TLG_TAG_CO MMENT	VT_BSTR	
ENUM_TLG_TAG _ARCTYPE	8	VT_TLG_TAG_AR CTYPE	VT_I4	
ENUM_TLG_TAG _SUPPLYTYPE	9	VT_TLG_TAG_SU PPLYTYPE	VT_I4	
ENUM_TLG_TAG _LOCKED	10	VT_TLG_TAG_LO CKED	VT_I4	
ENUM_TLG_TAG _SCANTIME	11	VT_TLG_TAG_SC ANTIME	VT_BSTR	Required for Cre- ate
NUM_TLG_TAG_ ARCTIME	12	VT_TLG_TAG_AR CTIME	VT_BSTR	Required for Cre- ate
ENUM_TLG_TAG _ARCFACTOR	13	VT_TLG_TAG_AR CFACTOR	VT_I4	Required for Cre- ate
ENUM_TLG_TAG _VARCOUNT- FORWARD	14	VT_TLG_TAG_VA RCOUNTFOR- WARD	VT_I4	
ENUM_TLG_TAG _VARCOUNT- BACKWARD	15	VT_TLG_TAG_VA RCOUNTBACK- WARD	VT_I4	
ENUM_TLG_TAG _SAVEBYFAULT	16	VT_TLG_TAG_SA VEBYFAULT	VT_I4	
ENUM_TLG_TAG _ARCTRIGGER	17	VT_TLG_TAG_AR CTRIGGER	VT_I4	Required for Cre- ate
ENUM_TLG_TAG _STATETEX- THIGH	18	VT_TLG_TAG_ST ATETEXTHIGH	VT_BSTR	
ENUM_TLG_TAG _STATETEXT- LOW	19	VT_TLG_TAG_ST ATETEXTLOW	VT_BSTR	
ENUM_TLG_TAG _VARPRO	20	VT_TLG_TAG_VA RPRO	VT_I4	
ENUM_TLG_TAG _FUNCVALPRO	21	VT_TLG_TAG_FU NCVALPRO	VT_BSTR	
ENUM_TLG_TAG _DLLVARPRO	22	VT_TLG_TAG_DL LVARPRO	VT_BSTR	
ENUM_TLG_TAG _FUNCSTARTE- VENT	23	VT_TLG_TAG_FU NCSTARTEVENT	VT_BSTR	
ENUM_TLG_TAG _DLLSTARTE- VENT	24	VT_TLG_TAG_DL LSTARTEVENT	VT_BSTR	
ENUM_TLG_TAG _FUNCSTOPE- VENT	25	VT_TLG_TAG_FU NCSTOPEVENT	VT_BSTR	

Definition in ENUM_TLG_TAG _FIELDINDEX	Field index	Associated type definition	Type	Comments/ limita- tions
ENUM_TLG_TAG _DLLSTOPE- VENT	26	VT_TLG_TAG_DL LSTOPEVENT	VT_BSTR	
ENUM_TLG_TAG _FUNCDYNAMIC	27	VT_TLG_TAG_FU NCDYNAMIC	VT_BSTR	
ENUM_TLG_TAG _DLLDYNAMIC	28	VT_TLG_TAG_DL LDYNAMIC	VT_BSTR	
ENUM_TLG_TAG _UNITDIRECT	29	VT_TLG_TAG_UN ITDIRECT	VT_BSTR	
ENUM_TLG_TAG _NAMEUNIT- STRUCT	30	VT_TLG_TAG_NA MEUNITSTRUCT	VT_BSTR	
ENUM_TLG_TAG _UNITINDEX	31	VT_TLG_TAG_UN ITINDEX	VT_I4	
ENUM_TLG_TAG _VARUPPER	32	VT_TLG_TAG_VA RUPPER	VT_R8	
ENUM_TLG_TAG _VARLOWER	33	VT_TLG_TAG_VA RLOWER	VT_R8	
ENUM_TLG_TAG _ARCUPPER	34	VT_TLG_TAG_AR CUPPER	VT_R8	
ENUM_TLG_TAG _ARCLOWER	35	VT_TLG_TAG_AR CLOWER	VT_R8	
ENUM_TLG_TAG _NAMESCALE	36	VT_TLG_TAG_NA MESCALE	VT_BSTR	
ENUM_TLG_TAG _NAMESOUR- CEARC	37	VT_TLG_TAG_NA MESOURCEARC	VT_BSTR	
ENUM_TLG_TAG _NAMESOURCE- VAR	38	VT_TLG_TAG_NA MESOURCEVAR	VT_BSTR	
ENUM_TLG_TAG _NAMERAWVAR	39	VT_TLG_TAG_NA MERAWVAR	VT_BSTR	
ENUM_TLG_TAG _CONVDLL	40	VT_TLG_TAG_CO NVDLL	VT_BSTR	
ENUM_TLG_TAG _PROCVARNAME	41	VT_TLG_TAG_PR OCVARNAME	VT_BSTR	Required for Cre- ate
ENUM_TLG_TAG _TIMEMODIFY	42	VT_TLG_TAG_TI MEMODIFY	VT_BSTR	Set internally (new time stamp for Cre- ate and Modify), cannot be changed
ENUM_TLG_TAG _ARCON- CHANGE	43	VT_TLG_TAG_AR CONCHANGE	VT_I4	
ENUM_TLG_TAG _ARCONHYS- TERESE	44	VT_TLG_TAG_AR CONHYSTERESE	VT_R8	

3.8 Tag and log functions (RT Professional)

Definition in ENUM_TLG_TAG_ _FIELDINDEX	Field index	Associated type definition	Type	Comments/ limita- tions
ENUM_TLG_TAG_ _FLAGS	45	VT_TLG_TAG_FL AGS	VT_I4	
ENUM_TLG_TAG_ _ALIAS	46	VT_TLG_TAG_ALI AS	VT_BSTR	
ENUM_TLG_TAG_ _VARSTARTE- VENT	47	VT_TLG_TAG_VA RSTARTEVENT	VT_BSTR	
ENUM_TLG_TAG_ _VARSTOPE- VENT	48	VT_TLG_TAG_VA RSTOPEVENT	VT_BSTR	
ENUM_TLG_TAG_ _PRECISION	49	VT_TLG_TAG_PR ECISION	VT_BSTR	
ENUM_TLG_TAG_ _TAGID	50	VT_TLG_TAG_TA GID	VT_I4	Set internally, can- not be changed

Flags for log tags

see Include-Datei pde_typ.h ENUM_TLG_TAG_FLAGVALUES WinCC V6.2 and higher

ENUM_TLG_TAG_FLAGVALUE_NOFLAGS	0	Default
ENUM_TLG_TAG_FLAGVALUE_LONG- TERM_DISABLE	1	
ENUM_TLG_TAG_FLAGVALUE_ARCHIV- ING_ON_SEGMENTCHANGE	2	

Selection flags for TLGCSReadTagMulti and TLGCSModifyTagMulti

These functions are available starting with WinCC V6.2.

TLG_TAG_SELECTION_BY_NAME	1	Selection with log name and tag name in each case.
---	---	---
TLG_TAG_DEFAULT_SELECTION	TLG_TAG_SELEC- TION_BY_NAME	Preferred selection type.
TLG_TAG_SELECTION_FLAGS_MASK	0x0000000FL	Masking of the selection in the flags.
TLG_TAG_SELECTION_FLAGS_OFFSET	0x0L	Value offset of the selection after masking in the flags.

3.8.2 Structures (RT Professional)

3.8.2.1 TLG_ARCHIV_STR (RT Professional)

Declaration

```
typedef struct {
    TCHAR          szName[
        PDE_DB_ARC_NAME_MAX LENGHT+1];
    TCHAR          szComment[
        PDE_DB_ARC_COMMENT_MAX LENGHT+1];
    TCHAR          szServername[
        PDE_DB_ARC_SERVERNAME_MAX LENGHT +1];
    DWORD          dwTyp;
    DWORD          dwAccessRead;
    DWORD          dwAccessWrite;
    TCHAR          szArchivAction[
        PDE_DB_ARC_ACTIONNAME_MAX LENGHT+1];
    BOOL           fLocked;
    DWORD          dwRecordTyp;
    DWORD          dwFillMessage;
    DWORD          dwRecordSize;
    DWORD          dwStorage;
    TCHAR          szCircularAction[
        PDE_DB_ARC_ACTIONNAME_MAX LENGHT+1];
    TCHAR          szCompresTime[
        PDE_DB_TIMENAME_MAX LENGHT+1];
    DWORD          dwSourceProcess;
    TCHAR          szRawDatVar[
        PDE_DB_VAR_NAME_MAX LENGHT+1];
    TCHAR          szSendAct[
        PDE_DB_ARC_ACTIONNAME_MAX LENGHT+1];
    TCHAR          szRecAct[
        PDE_DB_ARC_ACTIONNAME_MAX LENGHT+1];
    DWORD          dwRecItems;
}
TLG_ARCHIV_STR;
```

Members

szName

Global parameter for the name of the log

szComment

Global parameter for the comment on the project

szServername

Global parameter reserved for future upgrades

dwTyp

Global parameter which identifies the log type.

TLG_ARCHIV_TYP_PROCESS	Process value log
TLG_ARCHIV_TYP_COMPRESS	Compressed log

dwAccessRead

Global parameter which identifies the authorization stage for read access (0...999).

dwAccessWrite

Global parameter which identifies the authorization stage for write access (0...999).

szArchivAction

Global parameter with the name of the action for general log processing.

fLocked

Global parameters which identifies a log as freed or locked.

dwRecordTyp

Parameter for circular and sequential logs which identifies the log type.

TLG_RECORD_TYP_CIRCULAR	Circular log
TLG_RECORD_TYP_FOLLOW	Sequential log

dwFillMessage

Parameter for circular, sequential and compressed logs which identifies when a filling level message is to be sent. Max. two filling level messages can be configured with a logic OR operation.

e.g.: 100% || 50%-90%)

TLG_FILL_MESSAGE_NO	No filling level message
TLG_FILL_MESSAGE_50	Filling level message at 50%
TLG_FILL_MESSAGE_60	Filling level message at 60%
TLG_FILL_MESSAGE_70	Filling level message at 70%
TLG_FILL_MESSAGE_80	Filling level message at 80%
TLG_FILL_MESSAGE_90	Filling level message at 90%
TLG_FILL_MESSAGE_100	Filling level message at 100%

dwRecordSize

Parameter for circular logs which defines the size of the cyclic buffer in data records.

dwStorage

Parameter for circular logs which identifies the log location.

TLG_STORAGE_MEMORY	Circular log in the main memory
TLG_STORAGE_HD	Circular log on hard disk

szCircularAction

Parameter for circular logs, name of the action for exporting the data of a circular log.

szCompressTime

Parameter for compressed logs, name of the timer object which specifies the compression time period.

dwSourceProcess

Parameter for compressed logs which identifies the processing type of the source log.

PDE_COMPSRC_CALC	Calculate
PDE_COMPSRC_CALCCOPY	Calculate and copy
PDE_COMPSRC_CALCDEL	Calculate and delete
PDE_COMPSRC_CALCCOPYDEL	Calculate, copy and delete

szRawDatVar

The parameter is reserved for future development and must be preset to 0.

szSendAct

The parameter is reserved for future development and must be preset to 0.

szRecAct

The parameter is reserved for future development and must be preset to 0.

dwRecItems

The parameter is reserved for future development and must be preset to 0.

Comment

The parameters of the TLG_ARCHIV_STR structure depend on the log type (process log or compressed log) and the type of logging (circular or sequential log).

Required files

pde_def.h

API functions

TLGReadArchiv (Page 2236)	Read log parameters

See also

TLGReadArchiv (Page 2236)

3.8.2.2 TLG_ARCHIVDataraw (RT Professional)

Declaration

```
typedef struct {
    SYSTEMTIME    stTime;
    double        doValue;
    DWORD         dwFlags;
}
TLG_ARCHIVDataraw;
```

Members

stTime

stTime contains the logging time.

doValue

doValue contains the existing value at the time stTime.

dwFlags

The Tag Logging sets flags for every value which is written into the log which provides information about the state of the tags.

The value must be converted to hexadecimal format for analysis of the flags. Whereby

- the left word (HighWord) contains flags from the data manager
- the left word (HighWord) quality codes, if PDE_RT_QUALITYCODE is set
- the right word (LowWord) status flags from the Tag Logging

Tag Logging status flags (LowWord):		
PDE_RT_DAYLIGHT	0x0001	summer time, status can be determined with GetTimeZomeInformation)
PDE_RT_SUBSTITUTION	0x0002	Substitute value
PDE_RT_TIME_BEVOR_JUMP	0x0004	Value prior to time jump
PDE_RT_TIME_BEHIND_JUMP	0x0008	Value after time jump
PDE_RT_TIME_OVERLAPPED	0x0010	Values during time overlap
PDE_RT_LOAD_SYSTEM	0x0020	For the first time after creating the log archived value
PDE_RT_RELOAD_SYSTEM	0x0040	First value after activating runtime
PDE_RT_CMPCOPY	0x0080	Compressed value
PDE_RT_TIME_CHANGED	0x0100	Time change took place
PDE_RT_HAND	0x0200	Manual tag supply
PDE_RT_ONCHANGEBACKUPVALUESTART	0x0400	
PDE_RT_ONCHANGEBACKUPVALUESTOP	0x0800	
PDE_RT_QUALITYCODE	0x1000	

data manager status flags (HighWord)		
DM_VARSTATE_NOT_ESTABLISHED	0x0001	Connection to partner not established
DM_VARSTATE_HANDSHAKE_ERROR	0x0002	Protocol error
DM_VARSTATE_HARDWARE_ERROR	0x0004	Network module faulty
DM_VARSTATE_MAX_LIMIT	0x0008	Configured high limit violated
DM_VARSTATE_MIN_LIMIT	0x0010	Configured low limit violated
DM_VARSTATE_MAX_RANGE	0x0020	High format limit violated
DM_VARSTATE_MIN_RANGE	0x0040	Low format limit violated
DM_VARSTATE_CONVERSION_ERROR	0x0080	Display conversion error in connection with MAX/ MIN_RANGE
DM_VARSTATE_STARTUP_VALUE	0x0100	Initialization value of tag
DM_VARSTATE_DEFAULT_VALUE	0x0200	Substitute value of tag
DM_VARSTATE_ADDRESS_ERROR	0x0400	Addressing error in the channel
DM_VARSTATE_INVALID_KEY	0x0800	Tag not found or does not exist
DM_VARSTATE_ACCESS_FAULT	0x1000	Access to tag not permitted
DM_VARSTATE_TIMEOUT	0x2000	Timeout no feedback from channel
DM_VARSTATE_SERVERDOWN	0x4000	Connection to server interrupted or server run down

	WinCC tag status	Quality code according to Profibus PA/OPC
DM_VARSTATE_SERVERDOWN (0x4000)	Server is down	Bad, out of service, 0x1C
DM_VARSTATE_HARDWARE_ERROR (0x0004)	Network module faulty	Bad, device failure, 0x0C
DM_VARSTATE_NOT_ESTABLISHED (0x0001)	Connection to partner not es- tablished	Bad, not connected, 0x08
DM_VARSTATE_TIMEOUT (0x2000)	Timeout no feedback from channel	Uncertain, last usable value, 0x44
DM_VARSTATE_HANDSHAKE_ERROR (0x0002)	Protocol error	Bad, no communication (no usable value), 0x18
DM_VARSTATE_ADDRESS_ERROR (0x0400)	Addressing error in the channel	Bad, configuration error, value not accep- ted, 0x04
DM_VARSTATE_INVALID_KEY (0x0800)	Tag not found or does not exist	Bad, configuration error, value not accep- ted, 0x04
DM_VARSTATE_ACCESS_FAULT (0x1000)	Access to tag not permitted	Bad, configuration error, value not accep- ted, 0x04
DM_VARSTATE_MAX_RANGE (0x0020)	High format limit violated	Uncertain, engineering unit range violation, high limit set, 0x56
DM_VARSTATE_MIN_RANGE (0x0040)	Low format limit violated	Uncertain, engineering unit range violation, low limit set, 0x55
DM_VARSTATE_CONVERSION_ERROR (0x0080)	Display conversion error	Uncertain, engineering unit range violation, set without limits, 0x54
DM_VARSTATE_MAX_LIMIT (0x0008)	Configured high limit violated	Oring simulation to the limit bit HIGH, 0x01 with the other tag status, e.g. gives "good (cascade), ok" 0xC2

3.8 Tag and log functions (RT Professional)

	WinCC tag status	Quality code according to Profibus PA/OPC
DM_VARSTATE_MIN_LIMIT (0x0010)	Configured low limit violated	Oring simulation to the limit bitLOW, 0x02 with the other tag status, e.g. gives "good (cascade), ok" 0xC1
DM_VARSTATE_STARTUP_VALUE (0x0100)	Initialization value of tag	Uncertain, initial value, 0x4C
DM_VARSTATE_DEFAULT_VALUE (0x0200)	Substitute value of tag	Uncertain, substitute-set, 0x48

Comment

If the current timer for transferring a SYSTEMTIME parameter is required, the GetLocalTime must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

Set the PDE_RT_DAYLIGHT flag if the appropriate time falls in summer time. You can determine the status of the flag with the GetTimeZoneInformation system function (TIME_ZONE_ID_DAYLIGHT).

Required files

pdertdef.h

pdert.h

dmdefs.h

API functions

TLGGetArchivDataEx (Page 2224)	Read out data of a log

See also

TLGGetArchivDataEx (Page 2224)

3.8.2.3 TLG_BACKUP_TABLE_INFO (RT Professional)

Declaration

```
typedef struct {
    TCHAR      szArchivName[ _MAX_PATH + 1 ];
    TCHAR      szFileName[ _MAX_PATH + 1 ];
    TCHAR      szComment[ _MAX_PATH + 1 ];
    DWORD      dwFormatFlags;
    DWORD      dwJobFlags;
    DWORD      dwSize;
    SYSTEMTIME sysFrom;
    SYSTEMTIME sysTo;
    DWORD      dwUserData;
}
TLG_BACKUP_TABLE_INFO;
```

Members

szArchivName

Name of the log from which the data is to be exported.

szFileName

Name of the export file with path and extension

szComment

Text of comment for export

dwFormatFlags

Format specifier:

TLG_BAKFMT_CSV:	CSV format (Comma Separated Value)

dwJobFlags

Job-specific identification, following are possible:

TLG_BACKUP_EXPORT:	Simple backup without deleting the source data
TLG_BACKUP_EVACUATE:	Export with deletion of the sources

dwSize

Size of the data to be exported

sysFrom

Start time from which the data is to be exported

sysTo

End time up to which the data is to be exported

dwUserData

Application-specific data

Required files

pdertdef.h

API functions

TLG_ENUMBACKUP_ENTRIES (Page 2254)	List backups (callback)

See also

TLG_ENUMBACKUP_ENTRIES (Page 2254)

3.8.2.4 TLG_CURVESCALEX (RT Professional)

Declaration

```
typedef struct {
    DWORD      dwDataTyp;
    DWORD      dwBufferSize;
    DWORD      dwRangeTyp;
    DWORD      dwAxisLocation;
    BOOL       fActualize;
    BOOL       fAutoRange;
    BOOL       fGridLinesBig;
    BOOL       fGridLinesFine;
    BOOL       fGridLinesBigVisible;
    BOOL       fGridLinesFineVisible;
    BOOL       fPercent;
    BOOL       fLimitRange;
    BOOL       fSubstitute;
    SYSTEMTIME stFrom;
    SYSTEMTIME stTo;
    double     doFrom;
    double     doTo;
    double     doGridBig;
    double     doGridFine;
    double     doShowDigits;
    double     doLimitUpper1;
    double     doLimitUpper2;
    double     doLimitUpper3;
    double     doLimitLower1;
    double     doLimitLower2;
    double     doLimitLower3;
    double     doDisplayRangeFrom;
    double     doDisplayRangeTo;
    double     doLimitRangeFrom;
    double     doLimitRangeTo;
    COLORREF   crColor;
    COLORREF   crColorTimeOverlapped;
    COLORREF   crColorTimeJump;
    COLORREF   crColorLimitUpper1;
    COLORREF   crColorLimitUpper2;
    COLORREF   crColorLimitUpper3;
    COLORREF   crColorLimitLower1;
    COLORREF   crColorLimitLower2;
    COLORREF   crColorLimitLower3;
    TCHAR      szSelectSQL[ TLG_MAX_SQL_SELECT ];
    TCHAR      szText[ TLG_MAX_STD_TEXT_NAME ];
    TCHAR      szFunction[ TLG_MAX_FUNCTION_NAME ];
    TCHAR      szDLL[ TLG_MAX_DLL_NAME ];
}
TLG_CURVESCALEX;
```

Members**dwDataType**

dwDataType determines which data type the trend is based on:

TLG_DATATYP_TIME-RANGE	0x00000001	for time range: The data originate from a time-controlled log (e.g.: process value log). The values specified in stFrom and stTo are valid. The fAutoRange flag then has no effect. Otherwise it applies: The data originate from a data-controlled log (e.g.: user log). The values specified in doFrom and doTo are valid.
TLG_DATATYP_USER-ARCHIV	0x00000002	irrelevant for trends: The data originate from a data-controlled user log. The values specified in doFrom and doTo are valid.
TLG_DATATYP_BLOCKDATA	0x00000004	Data is set by TLGSetTemplateData: The values in stFrom, stTo, doFrom and doTo are then irrelevant.

dwBufferSize

dwBufferSize corresponds to the number of measuring points of the trend and determines the size of the trend data buffer.

dwRangeTyp

The parameter is reserved for future development and must be preset to 0.

dwAxisLocation

The parameter is reserved for future development and must be preset to 0.

fActualize

The trend is displayed dynamically when fActualize = TRUE. Otherwise it is static.

fAutoRange

Autoscaling is activated if fAutoRange = TRUE.

If a time-controlled log exists (dwDataType = TLG_DATATYP_TIMERANGE), fAutoRange has no effect and the values specified in stFrom and stTo are valid.

fGridLinesBig

The parameter is reserved for future development and must be preset to 0.

fGridLinesFine

The parameter is reserved for future development and must be preset to 0.

fGridLinesBigVisible

The parameter is reserved for future development and must be preset to 0.

fGridLinesFineVisible

The parameter is reserved for future development and must be preset to 0.

fPercent

The parameter is reserved for future development and must be preset to 0.

fLimitRange

The parameter is reserved for future development and must be preset to 0.

fSubstitute

The parameter is reserved for future development and must be preset to 0.

stFrom

Start time of the display range. The parameter is only evaluated if
dwDataTy = TLG_TIME_RANGE.

stTo

End time of the display range. The parameter is only evaluated if
dwDataTy = TLG_TIME_RANGE.

If the current timer for transferring a SYSTEMTIME parameter is required, the GetLocalTime must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

doFrom

Start value of the display range. The parameter is only evaluated if
dwDataTy = TLG_DATA_RANGE.

doTo

End value of the display range. The parameter is only evaluated if in
dwDataTy = TLG_DATA_RANGE.

doGridBig

The parameter is reserved for future development and must be preset to 0.

doGridFine

The parameter is reserved for future development and must be preset to 0.

doShowDigits

Number of decimal places

doLimitUpper1

The parameter is reserved for future development and must be preset to 0.

doLimitUpper2

The parameter is reserved for future development and must be preset to 0.

doLimitUpper3

The parameter is reserved for future development and must be preset to 0.

doLimitLower1

The parameter is reserved for future development and must be preset to 0.

doLimitLower2

The parameter is reserved for future development and must be preset to 0.

doLimitLower3

The parameter is reserved for future development and must be preset to 0.

doDisplayRangeFrom

Indicates from which value a data-controlled display is to be made.

doDisplayRangeTo

Indicates up to which value a data-controlled display is to be made.

doLimitRangeFrom

The parameter is reserved for future development and must be preset to 0.

doLimitRangeTo

The parameter is reserved for future development and must be preset to 0.

crColor

The Windows-specific 32-bit value crColor determines the color in which the trend is to be displayed.

crColorTimeOverlapped

The parameter is reserved for future expansions.

crColorTimeJump

The parameter is reserved for future expansions.

crColorLimitUpper1

The parameter is reserved for future expansions.

crColorLimitUpper2

The parameter is reserved for future expansions.

crColorLimitUpper3

The parameter is reserved for future expansions.

crColorLimitLower1

The parameter is reserved for future expansions.

crColorLimitLower2

The parameter is reserved for future expansions.

crColorLimitLower3

The parameter is reserved for future expansions.

szSelectSQL

The parameter is reserved for future development and must be preset to NULL.

szText;

Labeling of the x-axis

szFunction

The parameter is reserved for future development and must be preset to NULL.

szDLL

The parameter is reserved for future development and must be preset to NULL.

Comment

TLG_CURVESCALEX is used by the TLG_TPLITEM_CURVE (Page 2179) structure.

Required files

pdertdef.h

See also

TLG_TPLITEM_CURVE (Page 2179)

3.8.2.5 TLG_CURVESCALEY (RT Professional)

Declaration

```

typedef struct {
    DWORD      dwDataTyp;
    DWORD      dwRangeTyp;
    DWORD      dwAxisLocation;
    DWORD      dwCurveForm;
    BOOL       fAutoRange;
    BOOL       fGridLinesBig;
    BOOL       fGridLinesFine;
    BOOL       fGridLinesBigVisible;
    BOOL       fGridLinesFineVisible;
    BOOL       fPercent;
    BOOL       fLimitRange;
    BOOL       fSubstitute;
    SYSTEMTIME stFrom;
    SYSTEMTIME stTo;
    double     doFrom;
    double     doTo;
    double     doGridBig;
    double     doGridFine;
    double     doShowDigitsF;
    double     doShowDigitsB;
    double     doLimitUpper1;
    double     doLimitUpper2;
    double     doLimitUpper3;
    double     doLimitLower1;
    double     doLimitLower2;
    double     doLimitLower3;
    double     doDisplayRangeFrom;
    double     doDisplayRangeTo;
    double     doLimitRangeFrom;
    double     doLimitRangeTo;
    COLORREF   crColor;
    COLORREF   crColorTimeOverlapped;
    COLORREF   crColorTimeJump;
    COLORREF   crColorLimitUpper1;
    COLORREF   crColorLimitUpper2;
    COLORREF   crColorLimitUpper3;
    COLORREF   crColorLimitLower1;
    COLORREF   crColorLimitLower2;
    COLORREF   crColorLimitLower3;
    TCHAR      szSelectSQL[ TLG_MAX_SQL_SELECT ];
    TCHAR      szText[ TLG_MAX_STD_TEXT_NAME ];
}
TLG_CURVESCALEY;

```


Members

dwDataType

The data type (time/user/block data) has no significance here.

dwRangeTyp

The parameter is reserved for future development and must be preset to 0.

dwAxisLocation

The parameter is reserved for future development and must be preset to 0.

dwCurveForm

The appearance of the trend is defined by dwCurveForm

TLG_CURVEFORM_STEP	View of the trend as a step trend
TLG_CURVEFORM_POINTS	View of the trend as a single point
TLG_CURVEFORM_DIRECT	Connect points of the trend linearly

fAutoRange

fAutoRange = TRUE adapts the display range automatically.

fGridLinesBig

fgridLinesBig = TRUE activates big grid lines.

fGridLinesFine

fgridLinesFine = TRUE activates fine grid lines.

fGridLinesBigVisible

If fgridLinesBigVisible = TRUE, the big grid lines are displayed.

fGridLinesFineVisible

If fgridLinesFineVisible = TRUE, the fine grid lines are displayed.

fPercent

The parameter is reserved for future development and must be preset to 0.

fLimitRange

The parameter is reserved for future development and must be preset to 0.

fSubstitute

The parameter is reserved for future development and must be preset to 0.

stFrom

The parameter is reserved for future development.

stTo

The parameter is reserved for future development.

doFrom

Lower limit of the display range on the y-axis.

doTo

Upper limit of the display range on the y-axis.

doGridBig

Spacing of the big grid lines.

doGridFine

Spacing of the fine grid lines.

doShowDigitsF

Places in display of values (before decimal point)

doShowDigitsB

Determines the number of decimal places

doLimitUpper1

The parameter is reserved for future development and must be preset to 0.

doLimitUpper2

The parameter is reserved for future development and must be preset to 0.

doLimitUpper3

The parameter is reserved for future development and must be preset to 0.

doLimitLower1

The parameter is reserved for future development and must be preset to 0.

doLimitLower2

The parameter is reserved for future development and must be preset to 0.

doLimitLower3

The parameter is reserved for future development and must be preset to 0.

doDisplayRangeFrom

Determines the value from which the display range begins.

doDisplayRangeTo

Determines the value up to which the display range extends.

doLimitRangeFrom

The parameter is reserved for future development and must be preset to 0.

doLimitRangeTo

The parameter is reserved for future development and must be preset to 0.

crColor

The Windows-specific 32-bit value crColor determines the color in which the trend is to be displayed.

crColorTimeOverlapped

The parameter is reserved for future development.

crColorTimeJump

The parameter is reserved for future development.

crColorLimitUpper1

The parameter is reserved for future development.

crColorLimitUpper2

The parameter is reserved for future development.

crColorLimitUpper3

The parameter is reserved for future development.

crColorLimitLower1

The parameter is reserved for future development.

crColorLimitLower2

The parameter is reserved for future development.

crColorLimitLower3

The parameter is reserved for future development.

szSelectSQL[TLG_MAX_SQL_SELECT];

The parameter is reserved for later upgrades and must be defaulted with ZERO.

szText[TLG_MAX_STD_TEXT_NAME];

Labeling of the y-axis

Comment

TLG_CURVESCALEY is used by the AUTOHOTSPOT structure.

Required files

pdertdef.h

See also

TLG_TPLITEM_CURVE (Page 2179)

3.8.2.6 TLG_GETARCHIVDATA (RT Professional)

Declaration

```
typedef struct {
    LPTSTR      lpszArchivName;
    LPTSTR      lpszVarName;
    SYSTEMTIME  stTime;
    double      doValue;
    DWORD       dwFlags;
}
TLG_GETARCHIVDATA;
```

Members

- lpszArchivName**
Pointer to the log name from which the data was read out.
- lpszVarName**
Pointer to the tag name whose values were read out.
- stTime**
x-value of the log value
- doValue**
y-value of the log value
- dwFlags**
The parameter is reserved for future development and must be preset to 0.

Required files

pdertdef.h

API functions

TLG_GETARCHIVDATA_CALLBACK (Page 2222)	Read out data of a log (callback)

See also

TLG_GETARCHIVDATA_CALLBACK (Page 2222)

3.8.2.7 TLG_IO_BACKUP_SELECT (RT Professional)

Declaration

```
typedef struct {  
    SYSTEMTIME    sysFrom;  
    SYSTEMTIME    sysTo;  
    LPTSTR        lpszSqlString;  
}  
TLG_IO_BACKUP_SELECT;
```

Members

sysFrom

System time of the first data record to be selected

sysTo

System time of the last data record to be selected

lpszSqlString

The parameter is reserved for future development and must be preset to NULL.

Required files

pdertdef.h

API functions

TLGGetBackupSize (Page 2257)	Determine size of an exported data record
TLGExport (Page 2255)	Export log data

See also

TLGGetBackupSize (Page 2257)

TLGExport (Page 2255)

3.8.2.8 TLG_PROT_CURVE_INFOS (RT Professional)

Declaration

```
typedef struct {
    TCHAR      szArchivName[ 128 + 1 ];
    TCHAR      szVariableName[ 128 + 1 ];
    TCHAR      szTextX[ 128 + 1 ];
    TCHAR      szTextY[ 128 + 1 ];
    SYSTEMTIME stFrom;
    SYSTEMTIME stTo;
    double     doFrom;
    double     doTo;
    DWORD      dwFlags;
    DWORD      dwCurveForm;
}
    TLG_PROT_CURVE_INFOS;
```

Members

szArchivName

Name of the log from which the data to be logged originates.

szVariableName

Name of the logging tag whose values are to be logged.

szTextX

Labeling of the x-axis

szTextY

Labeling of the y-axis

stFrom

Start time from which logging starts The parameter is only evaluated if dwFlags has TLG_PROTFLG_TIME_RANGE set.

stTo

End time up to which logging takes place. The parameter is only evaluated if dwFlags has TLG_PROTFLG_TIME_RANGE set.

If a current timer for transferring a SYSTEMTIME parameter is required, the GetLocalTime function must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

doFrom

Start time from which logging is to take place. The parameter is only evaluated if dwFlags has TLG_PROTFLG_DATA_RANGE set.

doTo

End time up to which logging is to take place. The parameter is only evaluated if dwFlags has TLG_PROTFLG_DATA_RANGE set.

dwFlags

dwFlags determines which data type the trend is based on:

TLG_PROTFLG_TIME_RANGE	The data originate from a time-controlled log (e.g.: process value log). The values specified in stFrom and stTo are valid.
TLG_PROTFLG_DATA_RANGE	The data originate from a data-controlled log (e.g.: user log). The values specified in doFrom and doTo are valid.

dwCurveForm

The appearance of the trend is defined by dwCurveForm

TLG_CURVEFORM_STEP	View of the trend as a step trend
TLG_CURVEFORM_POINTS	View of the trend as a single point
TLG_CURVEFORM_DIRECT	Connect points of the trend linearly

Comment

TLG_PROT_CURVE_INFOS is used by the structures TLG_TABLESCALE (Page 2172) and TLG_TPLITEM_INFO (Page 2180).

Required files

pdertdef.h

API functions

TLGDrawCurvesInDC (Page 2238)	Show trends

See also

TLGDrawCurvesInDC (Page 2238)

TLG_TABLESCALE (Page 2172)

TLG_TPLITEM_INFO (Page 2180)

3.8.2.9 TLG_SCAL_STR (RT Professional)

Declaration

```
typedef struct {  
    double    doScalVarUpper;  
    double    doScalVarLower;  
    double    doScalArcUpper;  
    double    doScalArcLower;  
    TCHAR     szStructName[PDE_DB_SCALE_MAX_LENHT];  
}  
TLG_SCAL_STR;
```

Members

doScalVarLower

Lower limit of the tag value

doScalVarUpper

Upper limit of the tag value

doScalArcLower

The parameter is reserved for future development and must be preset to 0.

doScalArcUpper

The parameter is reserved for future development and must be preset to 0.

szStructName

Name of the scaling structure for the tag limit values

Comment

The TLG_SCAL_STR structure is used within the TLG_VAR_STR (Page 2182) structure and is provided for later upgrades.

Required files

pde_def.h

See also

TLG_VAR_STR (Page 2182)

3.8.2.10 TLG_TABLE_INFO (RT Professional)

Declaration

```
typedef struct {
    DWORD    dwArchivTyp;
    TCHAR    szArchivName[ _MAX_PATH + 1 ];
    DWORD    dwSaveTyp;
}
TLG_TABLE_INFO;
```

Members

dwArchivTyp

dwArchivTyp identifies the log type:

TLG_ARCTYP_USER	User log
TLG_ARCTYP_PROCESS	Process value log
TLG_ARCTYP_COMPRESS	Compressed log

szArchivName

The form in which the name of the log is provided depends on the functions which use TLG_TABLE_INFO.

The enumeration of the logs by TLGEnumArchivs and TLGEnumArchivsSel supplies the log name as a table name, i.e. in the form "UA#ARCHIV#Archivname" for user logs and "PDE#HD#Archivname#Variablenname" respectively because a table entry for every tag exists here. Therefore every tag is enumerated.

The enumeration by TLGEnumArchivsEx supplies the "pure" log name.

dwSaveTyp

dwSaveTyp identifies the type of log:

TLG_ARCTYP_CIRCULAR	Circular log
TLG_ARCTYP_FOLLOW	Sequential log

Required files

pdertdef.h

API functions

TLG_ENUMTABLES (Page 2217)	List logs (callback)

See also

TLG_ENUMTABLES (Page 2217)

3.8.2.11 TLG_TABLESCALE (RT Professional)**Declaration**

```
typedef struct {
    BOOL        fActualize;
    BOOL        fVisible;
    BOOL        fModify;
    BOOL        fCommon;
    DWORD       dwDataTyp;
    SYSTEMTIME  stFrom;
    SYSTEMTIME  stTo;
    double      doFrom;
    double      doTo;
    double      doShowDigits;
    double      doLimitUpper1;
    double      doLimitUpper2;
    double      doLimitUpper3;
    double      doLimitLower1;
    double      doLimitLower2;
    double      doLimitLower3;
    OLORREF     crColor;
    COLORREF    crColorTimeOverlapped;
    COLORREF    crColorTimeJump;
    COLORREF    crColorLimitUpper1;
    COLORREF    crColorLimitUpper2;
    COLORREF    crColorLimitUpper3;
    COLORREF    crColorLimitLower1;
    COLORREF    crColorLimitLower2;
    COLORREF    crColorLimitLower3;
    TCHAR       szSelectSQL[ TLG_MAX_SQL_SELECT ];
}
TLG_TABLESCALE;
```

Members**fActualize**

TRUE	The trend view is dynamic.
FALSE	The trend view is static.

fVisible

TRUE	The column is visible.
FALSE	The column is invisible.

fModify

TRUE	The entries in the column can be edited.
FALSE	The entries in the column cannot be edited.

fCommon

TRUE	All columns of the table have a common time column.
FALSE	The columns of the table have no common time column.

dwDataTyp

The parameter is reserved for future development and must be preset to 0.

stFrom

Start time of the display range.

stTo

End time of the display range.

If a current timer for transferring a SYSTEMTIME parameter is required, the GetLocalTime function must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

doFrom

The parameter is reserved for future development and must be preset to 0.

doTo

The parameter is reserved for future development and must be preset to 0.

doShowDigits

Number of decimal places to be output.

doLimitUpper1

The parameter is reserved for future development and must be preset to 0.

doLimitUpper2

The parameter is reserved for future development and must be preset to 0.

dpoLimitUper3

The parameter is reserved for future development and must be preset to 0.

doLimitLower1

The parameter is reserved for future development and must be preset to 0.

doLimitLower2

The parameter is reserved for future development and must be preset to 0.

doLimitLower3

The parameter is reserved for future development and must be preset to 0.

crColor

The Windows-specific 32-bit value crColor defines the color used in the table column.

crColorTimeOverlapped

The parameter is reserved for future development.

crColorTimeJump

The parameter is reserved for future development.

crColorLimitUpper1

The parameter is reserved for future development.

crColorLimitUpper2

The parameter is reserved for future development.

crColorLimitUpper3

The parameter is reserved for future development.

crColorLimitLower1

The parameter is reserved for future development.

crColorLimitLower2

The parameter is reserved for future development.

crColorLimitLower3

The parameter is reserved for future development.

szSelectSQL

The parameter is reserved for later upgrades and must be defaulted with ZERO.

Comment

TLG_TABLESCALE is used by the TLG_TEMPLATEITEM_INFO (Page 2175) structure.

Required files

pdertdef.h

See also

TLG_PROT_CURVE_INFOS (Page 2166)

TLG_TPLITEM_INFO (Page 2180)

TLG_TEMPLATEITEM_INFO (Page 2175)

TLG_TPLITEM_TABLE (Page 2181)

3.8.2.12 TLG_TEMPLATEITEM_INFO (RT Professional)

Declaration

```

typedef struct {
    TCHAR                szTemplateName[
                        TLG_MAX_TEMPLATEITEM_NAME+1 ];
    TCHAR                szTemplateName[
                        TLG_MAX_TEMPLATE_NAME + 1 ];
    TCHAR                szArchivName[ 128 + 1 ];
    TCHAR                szVariableName[ 128 + 1 ];
    TCHAR                szDMVariableName[ 128 + 1 ];
    DWORD               dwReadAccessLevel;
    DWORD               dwWriteAccessLevel;
    DWORD               dwArchivTyp;
    TCHAR                szTimeNameRange[ 128 + 1 ];
    DWORD               dwTemplateItemType;
    DWORD               dwTemplateType;
    BOOL                fVisible;
    TLG_TPLITEM_INFO    tplInfo;
}
TLG_TEMPLATEITEM_INFO;

```

Members

szTemplateItemName

Name of the trend or column

szTemplateName

Name of the trend or table window template

szArchivName

Name of the log in which the log tag linked with the curve or column is logged.

szVariableName

Name of the log tag linked with the trend or column

szDMVariableName

Name of the data manager tag

dwReadAccessLevel

In the enumeration of trends or columns dwReadAccessLevel contains the user authorization level for read access.

dwWriteAccessLevel

In the enumeration of trends or columns dwWriteAccessLevel contains the user authorization level for write access.

dwArchivTyp

dwArchivTyp identifies the log type:

TLG_ARCTYP_USER	User log
TLG_ARCTYP_PROCESS	Process value log
TLG_ARCTYP_COMPRESS	Compressed log

szTimeNameRange

Name of the time object which is to be used to determine a time range from a start time.

dwTemplateItemTyp

dwTemplateItemTyp must correspond to the value specified in dwTemplateTyp:

TLG_TEMPLATEITEM_ALL	Trend and table templates
TLG_TEMPLATEITEM_CURVE	Trend template
TLG_TEMPLATEITEM_TABLE	Table template

dwTemplateTyp

dwTemplateTyp identifies the type of window template

TLG_TEMPLATE_CURVE	Trend window template
TLG_TEMPLATE_TABLE	Table window template

fVisible

TRUE	The column or trend is visible.
FALSE	The column or trend is invisible.

tplInfo

Structure of the TLG_TPLITEM_INFO (Page 2180) type with the properties of a trend or column template.

Required files

pdertdef.h

API functions

TLG_TABLESCALE (Page 2170)	Table column (properties) (structure)

See also

TLG_TPLITEM_INFO (Page 2180)
 TLG_TABLESCALE (Page 2170)
 TLGInsertTemplateItem (Page 2240)

3.8.2.13 TLG_TIME_STR (RT Professional)**Declaration**

```
typedef struct {
    DWORD      dwBasis;
    DWORD      dwFactor;
    TCHAR      szTimeName[
                PDE_DB_TIMENAME_MAX_LENHT + 1 ];
}
    TLG_TIME_STR;
```

Members**dwBasis**

Timebase The cycle time is given by multiplying the time factor and timebase.

TLG_TBASE_500MS	500ms
TLG_TBASE_SEC	1 second
TLG_TBASE_MIN	1 minute
TLG_TBASE_HOUR	1 hour
TLG_TBASE_DAY	1 day

dwFactor

Time factor The cycle time is given by multiplying the time factor and timebase.

szTimeName

Name of the time object

Comment

Times are to be understood as acquisition and logging cycles which can be freely assigned. A time object can be used simultaneously as an acquisition cycle and a logging cycle.

Required files

pde_def.h

API functions

TLGReadTime (Page 2251)	Determine time object parameter

See also

TLGReadTime (Page 2251)

3.8.2.14 TLG_TIMEDATA (RT Professional)**Declaration**

```
typedef struct {
    TCHAR    szTimeName[ _MAX_PATH + 1 ];
    DWORD    dwTimeBase;
    DWORD    dwTimeValue;
}
TLG_TIMEDATA;
```

Members**szTimeName**

Name of the time object

dwTimeBase

Timebase The cycle time is given by multiplying the time factor and timebase.

TLG_TBASE_500MS	500ms
TLG_TBASE_SEC	1 second
TLG_TBASE_MIN	1 minute
TLG_TBASE_HOUR	1 hour
TLG_TBASE_DAY	1 day

dwTimeValue

Time factor The cycle time is given by multiplying the time factor and timebase.

Required files

pdertdef.h

API functions

TLG_ENUMTIMES_CALLBACK (Page 2250)	List time objects (callback)

See also

TLG_ENUMTIMES_CALLBACK (Page 2250)

3.8.2.15 TLG_TPLITEM_CURVE (RT Professional)**Declaration**

```
typedef struct {  
    TLG_CURVESCALEX    csx;  
    TLG_CURVESCALEY    csy;  
}  
TLG_TPLITEM_CURVE;
```

Members**csx**

The TLG_CURVESCALEX (Page 2155) structure contains the data for scaling the X-axis of a trend.

csy

The TLG_CURVESCALEY (Page 2160) structure contains the data for scaling the Y-axis of a trend.

Description

TLG_TPLITEM_CURVE is used by the TLG_TPLITEM_INFO (Page 2180) structure.

Required files

pdertdef.h

See also

TLG_TPLITEM_INFO (Page 2180)

TLG_CURVESCALEX (Page 2155)

TLG_CURVESCALEY (Page 2160)

3.8.2.16 TLG_TPLITEM_INFO (RT Professional)

Declaration

```
typedef union {  
    TLG_TPLITEM_CURVE    tplCurve;  
    TLG_TPLITEM_TABLE    tplTable;  
}  
TLG_TPLITEM_INFO;
```

Members

tplCurve

The properties of a trend are read from the TLG_TPLITEM_CURVE (Page 2177) structure.

tplTable

The properties of a column are read from the TLG_TPLITEM_TABLE (Page 2181) structure.

Comment

TLG_TPLITEM_INFO is used by the TLG_TEMPLATEITEM_INFO (Page 2173) structure.

Since a tag can be displayed both in a trend and in a table, both structures TLG_TPLITEM_CURVE and TLG_TPLITEM_TABLE can be assigned values.

Required files

pdertdef.h

See also

TLG_PROT_CURVE_INFOS (Page 2166)

TLG_TABLESCALE (Page 2170)

TLG_TEMPLATEITEM_INFO (Page 2173)

TLG_TPLITEM_CURVE (Page 2177)

TLG_TPLITEM_TABLE (Page 2181)

3.8.2.17 TLG_TPLITEM_TABLE (RT Professional)

Declaration

```
typedef struct {  
    TLG_TABLESCALE    ts;  
}  
TLG_TPLITEM_TABLE;
```

Members

ts

TLG_TABLESCALE (Page 2170) with the properties of a table column.

Description

TLG_TPLITEM_TABLE is used by the TLG_TPLITEM_INFO (Page 2178) structure.

For organizational reasons, TLG_TABLESCALE cannot be used directly in the TLG_TPLITEM_INFO structure.

Required files

pdertdef.h

See also

TLG_TPLITEM_INFO (Page 2178)

TLG_TABLESCALE (Page 2170)

3.8.2.18 TLG_VAR_STR (RT Professional)

Declaration

```

typedef struct {
    TCHAR          szVarName [PDE_DB_VAR_NAME_MAX_LENIGHT + 1];
    TCHAR          szProcName [PDE_DB_VAR_NAME_MAX_LENIGHT + 1];
    DWORD          dwVarType;
    DWORD          dwArchivStyle;
    TCHAR          szWriteBackTo [PDE_DB_VAR_WRITEBACK_MAX_LENIGHT + 1];
    DWORD          dwSupply;
    BOOL           fLocked;
    TCHAR          szComment [PDE_DB_VAR_COMMENT_MAX_LENIGHT + 1];
    TCHAR          szRecordCycle [PDE_DB_VAR_CYCLENAME_MAX_LENIGHT + 1];
    TCHAR          szArchivCycle [PDE_DB_VAR_CYCLENAME_MAX_LENIGHT + 1];
    DWORD          dwMultiple;
    DWORD          dwValueFlow;
    DWORD          dwValueFollow;
    DWORD          dwSaveByFault;
    DWORD          dwArchivTrigger;
    TCHAR          szTextHighSignal [PDE_DB_VAR_SIGNALTEX_MAX_LENIGHT + 1];
    TCHAR          szTextLowSignal [PDE_DB_VAR_SIGNALTEX_MAX_LENIGHT + 1];
    DWORD          dwValProcess;
    TCHAR          szFuncValProcess [PDE_DB_VAR_FUNCNAME_MAX_LENIGHT + 1];
    TCHAR          szDLLValProcess [PDE_DB_VAR_DLLNAME_MAX_LENIGHT + 1];
    TCHAR          szFuncStartEvent [PDE_DB_VAR_FUNCNAME_MAX_LENIGHT + 1];
    TCHAR          szDLLStartEvent [PDE_DB_VAR_DLLNAME_MAX_LENIGHT + 1];
    TCHAR          szFuncStopEvent [PDE_DB_VAR_FUNCNAME_MAX_LENIGHT + 1];
    TCHAR          szDLLStopEvent [PDE_DB_VAR_DLLNAME_MAX_LENIGHT + 1];
    TCHAR          szFuncDynamic [PDE_DB_VAR_FUNCNAME_MAX_LENIGHT + 1];
    TCHAR          szDLLDynamic [PDE_DB_VAR_DLLNAME_MAX_LENIGHT + 1];
    TCHAR          szUnitDirect [PDE_DB_VAR_UNITDIR_MAX_LENIGHT + 1];
    TCHAR          szUnitStruct [PDE_DB_VAR_UNITSTR_MAX_LENIGHT + 1];
    DWORD          dwRecItems;
    TCHAR          szSourceArchiv [PDE_DB_ARC_NAME_MAX_LENIGHT + 1];
    TCHAR          szSourceVarName [PDE_DB_VAR_NAME_MAX_LENIGHT + 1];
    TCHAR          szRawdataName [PDE_DB_VAR_NAME_MAX_LENIGHT + 1];
    TCHAR          szRawConvDLLName [TLG_DB_DLLNAME_MAX_LENIGHT + 1];
    DWORD          dwRawDataIndex;
    DWORD          dwRawDataFormat;
    TLG_SCAL_STR    ScaleStruct;
    TLG_RECORD_STR  RecordStruct;
} TLG_VAR_STR;

```

Comment

The assignment of the parameters of the TLG_VAR_STR structure depends on the logging type (acyclic, cyclic-selective or cyclic-continuous) and the tag type (binary or analog).

Members

szVarName

Global parameter for the name of the log tag or the tag group

The characters : ? " ' \ * % and blank may not be used in log tag names, otherwise the error TLG_API_NAME_WRONG_CHAR is output.

szProcName

Global parameter for the name of the process tag

dwVarType

Global parameter which identifies the log tag type.

TLG_VAR_TYP_ANALOG	0x00010001	analog tag: dwArchivTrigger is set internally to 0L (irrelevant)
TLG_VAR_TYP_BIN	0x00010002	binary tag: dwValProcess is set internally to 0L (irrelevant)
TLG_VAR_TYP_COMPRESS	0x00010004	Tag for compressed logs: dwSupply is set internally to TLG_SUPPLY_BY_SYSTEM, dwSaveByFault is set internally to TLG_SAVE_LAST_VALUE
TLG_VAR_TYP_PROCESS	0x00010008	process-controlled tag
	invalid value	Error message: TLG_API_VARTYPE_MISMATCH

dwArchivStyle

Global parameter which identifies the logging type.

TLG_STY_ACYCLIC	0x00800001	Acyclic logging
TLG_STY_CYCLIC_CON	0x00800002	Cyclic-continuous logging
TLG_STY_CYCLIC_SEL	0x00800004	Cyclic-selective logging
TLG_STY_ON_CHANGE	0x00800008	Detection for change
	invalid value	Error message TLG_API_ERR_ARCHIVSTYLE

szWriteBackTo

Global parameter into which the name of the log tag is to be written back.

dwSupply

Global parameter which identifies the tag supply type.

TLG_SUPPLY_BY_HAND	0L - FALSE	Manual tag input
TLG_SUPPLY_BY_SYSTEM	1L - TRUE	Supply from system (process coupling or internal tag)
	invalid value	Error message TLG_API_INVALID_PARAM

If dwVarType = TLG_VAR_TYP_COMPRESS, TLG_SUPPLY_BY_SYSTEM is always set internally.

fLocked

Global parameter which identifies the handling of the logging at system start.

TRUE	disabled
FALSE	enabled

szComment

Global parameter for the comment on the log tag

szRecordCycle

Name of the time object (e.g.: "1 minute") which specifies the acquisition cycle.

The parameter is relevant for the cyclic-selective, cyclic-continuous acquisition of analog and binary tags.

szArchivCycle

Name of the time object (e.g.: "1 minute") which specifies the logging cycle. The logging cycle is given by multiplying dwMultiple by szArchivCycle.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

dwMultiple

Identifies the multiplication factor. The logging cycle is given by multiplying dwMultiple by szArchivCycle.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

dwValueFlow

Identifies the number of flow values

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

dwValueFollow

Identifies the number of return values.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

dwSaveByFault

Identifies the memory behavior in case of a fault.

The parameter is relevant for analog and binary tags of all logging types.

TLG_SAVE_LAST_VALUE	1L	Save last valid value
TLG_SAVE_SUBST_VALUE	2L	Save substitute value
	invalid value	Error message TLG_API_INVALID_PARAM

If dwVarType = TLG_VAR_TYP_COMPRESS, TLG_SAVE_LAST_VALUE is always set internally.

dwArchivTrigger

Identifies the trigger behavior of the logging

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of binary tags.

TLG_TRIG_CHANGE	1L	Logging for every signal change
TLG_TRIG_CHANGE_01	2L	Logging for signal change 0 -> 1
TLG_TRIG_CHANGE_10	3L	Logging for signal change 1 -> 0
TLG_TRIG_ALLWAYS	4L	The value is logged in every logging cycle even without a signal change.
	invalid value	Error message TLG_API_INVALID_PARAM

If dwVarType = TLG_VAR_TYP_ANALOG, 0L is always set internally (irrelevant).

szTextHighSignal

Text for signal state 1.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of binary tags.

szTextLowSignal

Text for the signal state 0.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of binary tags.

dwValProcess

Identifies the processing of the log value to be saved over the acquired values within the logging cycle.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog tags.

TLG_VAL_ACTUAL	1L	Actual value
TLG_VAL_AVERAGE	2L	Average value
TLG_VAL_SUM	3L	Sum
TLG_VAL_MIN	4L	Minimum value
TLG_VAL_MAX	5L	Maximum value
TLG_VAL_DLL	6L	Function or DLL
	invalid value	Error message TLG_API_INVALID_PARAM

If dwVarType = TLG_VAR_TYP_BIN , 0L is always set internally (irrelevant).

szFuncValProcess

Name of the action or the DLL function which specifies the processing of the log value to be saved.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags and only relevant if dwValProcess = TLG_VAL_DLL.

szDLLValProcess

Name of the DLL

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags and only relevant if dwValProcess = TLG_VAL_DLL.

szFuncStartEvent

Name of the action or the DLL function which specifies the start event.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

szDLLStartEvent

Name of the DLL.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

szFuncStopEvent

Name of the action or the DLL function which specifies the stop event.

The parameter is relevant for the cyclic-selective logging of analog and binary tags.

szDLLStopEvent

Name of the DLL

The parameter is relevant for the cyclic-selective logging of analog and binary tags.

szFuncDynamic

Name of the action or the DLL function which specifies the dynamic switching of the detection and/or logging cycle.

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

szDLLDynamic

Name of the DLL

The parameter is relevant for the cyclic-selective, cyclic-continuous logging of analog and binary tags.

szUnitDirect

Identifies the unit of the log tag.

The parameter is relevant for analog tags of all logging types.

szUnitStruct

Name of the structure that contains the unit of the log tag. NULL if the unit is configured directly in szUnitDirect.

The parameter is relevant for analog tags of all logging types.

dwReclItems

The parameter is reserved for future development and must be preset to 0.

szSourceArchiv

Name of the source archive.

The parameter is relevant for the cyclic-continuous logging.

szSourceVarName

Name of the log tag in the source log

The parameter is relevant for the cyclic-continuous logging.

szRawdataName

Name of the raw data tag

The parameter is relevant for process-controlled tags.

szRawConvDLLName

Name of the normalization DLL

The parameter is relevant for process-controlled tags.

dwRawDataIndex

The number of the raw data tag corresponds to the log ID of the PLC in connection with S7PMC.

The parameter is relevant for process-controlled variables.

dwRawDataFormat

Format of the raw data tag

The parameter is relevant for process-controlled tags.

ScaleStruct

TLG_SCAL_STR (Page 2168) Structure with the limits of the tag.

RecordStruct

The parameter is reserved for future development.

Required files

pde_def.h

API functions

TLGReadVariable (Page 2207)	Determine tag parameter

See also

TLGReadVariable (Page 2207)

TLG_SCAL_STR (Page 2168)

3.8.2.19 TLG_VARIABLE_INFO (RT Professional)**Declaration**

```
typedef struct {
    DWORD    dwVariableTyp;
    TCHAR    szVariableName[ _MAX_PATH + 1 ];
}
TLG_VARIABLE_INFO;
```

Members**dwVariableType**

dwVariableType identifies the type of tag. Possible values are:

TLG_VAR_TYP_ANALOG	analog tag
TLG_VAR_TYP_BIN	binary tag
TLG_VAR_TYP_COMPRESS	tag for compressed logs
TLG_VAR_TYP_PROCESS	process-controlled tag

szVariableName

Name of the tag

Required files

pdertdef.h

pde_typ.h

API functions

TLG_ENUMVARIABLES (Page 2206)	List tags (callback)

See also

TLG_ENUMVARIABLES (Page 2206)

3.8.3 General functions (RT Professional)**3.8.3.1 TLGCSCConnect (RT Professional)****Description**

The function establishes a connection with the WinCC project database.

Declaration

```

BOOL TLGCSCConnect (
    HWND          hwndParent,
    LPCMN_ERROR    lpError );

```

Parameters

hwndParent

Window handle of the parent window

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection established

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Related functions

TLGCSCConnectEx (Page 2190)	Set up connection to project database
TLGCSDisConnect (Page 2191)	Set up connection to project database

Examples

Enum all acquisition and logging times (Page 2262) "TL01.cpp"

Read parameters of time object (Page 2276) "TL01.cpp"

Enum logs (Page 2265) "TL01.cpp"

Read log (Page 2272) "TL01.cpp"

See also

TLGCSCConnectEx (Page 2190)

Enum logs (Page 2265)

TLGOpenProject (Page 2198)

Read parameters of time object (Page 2276)

Enum all acquisition and logging times (Page 2262)

Read log (Page 2272)

TLGCSDisConnect (Page 2191)

3.8.3.2 TLGCSConnectEx (RT Professional)

Description

The function establishes a connection with the WinCC project database.

Declaration

```
BOOL TLGCSConnectEx (  
    HWND          hwndParent,  
    DWORD         dwMode,  
    LPCMN_ERROR    lpError );
```

Parameters

hwndParent

The parameter is reserved for later upgrades and must be defaulted with ZERO.

dwMode

The parameter is reserved for future development.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection established

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Related functions

TLGCSCConnect (Page 2186)	Set up connection to project database

See also

TLGCSCConnect (Page 2186)

TLGOpenProject (Page 2198)

3.8.3.3 TLGCSDisConnect (RT Professional)**Description**

The function terminates an established connection to the WinCC project database. The call is required so that the DLL can be cleanly unloaded again.

Note

The call may not be used in the destructor of an application (EXE, DLL, OCX etc.) because this can lead to a frozen call and subsequent program crash due to specific Microsoft-based mechanisms.

Declaration

```
BOOL TLGCSDisConnect (
    LPCMN_ERROR    lpError );
```

Parameters**lpError**

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Connection terminated.

FALSE

Error

Required files

pdecsccli.h
pdecsccli.lib
pdecsccli.dll

Related functions

TLGCSCConnect (Page 2186)	Set up connection to project database

Examples

Enum all acquisition and logging times (Page 2262) "TL01.cpp"
Read parameters of time object (Page 2276) "TL01.cpp"
Enum logs (Page 2265) "TL01.cpp"
Read log (Page 2272) "TL01.cpp"

See also

TLGCSCConnect (Page 2186)
Enum all acquisition and logging times (Page 2262)
Enum logs (Page 2265)
Read log (Page 2272)
Read parameters of time object (Page 2276)

3.8.3.4 TLGChangeLanguage (RT Professional)**Description**

The currently used data language can be switched with this function.

Declaration

```
BOOL TLGChangeLanguage (  
    DWORD          dwLanguage,  
    PCMN_ERROR     lpError );
```

Parameters**dwLanguage**

Code of the language to be used in future

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Data language switched

FALSE

Error

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_ERR_SUPPLY	An error has occurred

Required files

pdertcli.h
pde_glob.h
pdertcli.lib
pdertcli.dll

3.8.3.5 TLGConnect (RT Professional)**Description**

The function initializes the log system and establishes a connection with the Tag Logging Runtime.

Declaration

```

BOOL TLGConnect (
    HWND          hwndParent,
    PCMN_ERROR    lpError );

```

Parameters

hwndParent

The parameter is reserved for future development and must be preset to NULL.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Connection established

FALSE

Error

Comment

The call generates an invisible window for the communication which is only deleted with the TLGDisconnect function. This may be important when Windows functions are also used.

Error messages

TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed

Required files

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

Related functions

TLGDisconnect (Page 2195)	Terminate connection

Examples

Enumerate logs (Page 2270) "TL02.cpp"

See also

Enumerate logs (Page 2270)

TLGDisconnect (Page 2195)

Edit curve template - Example 1 (Page 2259)

3.8.3.6 TLGDisconnect (RT Professional)**Description**

An existing connection to the Tag Logging Runtime is disconnected with this function.

Declaration

```
BOOL TLGDisconnect (  
    PCMN_ERROR    lpError );
```

Parameters**lpError**

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Connection terminated

FALSE

Error

Comment

Note

The call may not be used in the destructor of an application (EXE, DLL, OCX...) because this can lead to jamming of the call and thus the program due to specific Microsoft-based mechanisms.

Error messages

TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed

Required files

pdertcli.h
pde_glob.h
pdertcli.lib
pdertcli.dll

Related functions

TLGConnect (Page 2191)	Establish connection

Examples

Enumerate logs (Page 2270) "TL02.cpp"

See also

Enumerate logs (Page 2270)
TLGConnect (Page 2191)
Edit curve template - Example 1 (Page 2259)
Enum all acquisition and logging times (Page 2262)

3.8.4 Project management functions (RT Professional)

3.8.4.1 TLGCloseProject (RT Professional)

Description

Closes the current project. All the data objects belonging to this project are closed.
The function is not longer significant and always supplies TRUE as a return value.

Declaration

```
BOOL TLGCloseProject (  
    HANDLE      hProject,  
    LPCMN_ERROR lpoes );
```

Parameters

hProject

Handle of a project opened with TLGOpenProject.

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Project closed

FALSE

Error

Required files

pdecsccli.h
pdecsccli.lib
pdecsccli.dll

Examples

Read parameters of time object (Page 2276) "TL01.cpp"
Enum logs (Page 2265) "TL01.cpp"
Read log (Page 2272) "TL01.cpp"

See also

Read parameters of time object (Page 2276)
Enum logs (Page 2265)
Read log (Page 2272)

3.8.4.2 TLGOpenProject (RT Professional)

Description

Establishes a connection with the WinCC project database.

Declaration

```
BOOL TLGOpenProject (  
    HANDLE*      lphProject,  
    LPTSTR       lpszProjectName,  
    HWND         hwndParent,  
    LPCMN_ERROR   lpoes );
```

Parameters

lphProject

Address of a memory area in which the handle is to be saved.

lpszProjectName

Name of the project to be opened.

The project path to be specified here can be determined with one of the following API functions:

- DMEnumOpenedProjects
- DMGetRuntimeProject
- TLGEnumProject

If a different project path than the currently open project is specified the TLG_API_PROJECT_NAME_NOT_EXIST error is returned. The TLG_API_NO_PROJECT_EXIST error is returned if no project is open.

hwndParent

Handle of the parent window

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Project opened

FALSE

Error

Comment

If no connection was established with the WinCC project data base by TLGCSCConnect first, the connection error TLG_API_NO_CONNECTION is returned.

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_NO_PROJECT_EXIST	No project exists
TLG_API_PROJECT_NAME_NOT_EXIST	Project name not found

Required files

pdecsccli.h
pdecsccli.lib
pdecsccli.dll

Related functions

TLGCSCConnect (Page 2186)	Set up connection to project database
TLGCSCConnectEx (Page 2188)	Set up connection to project database
TLGEnumProject (Page 2200)	List projects

Examples

Read parameters of time object (Page 2276) "TL01.cpp"

Enum logs (Page 2265) "TL01.cpp"

Read log (Page 2272) "TL01.cpp"

Enum tags of a log (Page 2268) "TL02.cpp"

See also

TLGEnumProject (Page 2200)

TLGCSCConnect (Page 2186)

TLGCSCConnectEx (Page 2188)

Read parameters of time object (Page 2276)

Enum logs (Page 2265)

Read log (Page 2272)

Enum tags of a log (Page 2268)

3.8.4.3 TLGEnumProject (RT Professional)

Description

The function determines the names of the open projects.

Declaration

```
BOOL TLGEnumProject (
    TLG_ENUM_PROJECT_NAME_CALLBACK    lpCallbackFunc,
    PVOID                             lpUser,
    LPCMN_ERROR                       lpoes );
```

Parameters

lpCallbackFunc

Pointer to your callback function which is called for every open project.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Projects listed

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Related functions

TLG_ENUM_PROJECT_NAME_CALLBACK (Page 2201)	List projects (callback)

See also

TLG_ENUM_PROJECT_NAME_CALLBACK (Page 2201)

TLGOpenProject (Page 2196)

3.8.4.4 TLG_ENUM_PROJECT_NAME_CALLBACK (RT Professional)**Description**

In order to be able to evaluate the projects listed by the system, you must provide a callback function of the TLG_ENUM_PROJECT_NAME_CALLBACK type.

Declaration

```

BOOL ( * TLG_ENUM_PROJECT_NAME_CALLBACK) (
    LPTSTR    lpszName,
    PVOID     lpUser );

```

Parameters**lpszName**

The lpszName pointer refers to the name of the first project.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

The return values depend on your implementation.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

pdecsccli.h

Related functions

TLGEnumProject (Page 2198)	List projects

See also

TLGEnumProject (Page 2198)

3.8.5 Functions for processing tags (RT Professional)**3.8.5.1 TLGEnumVariables (RT Professional)****Description**

The function determines all the tag names of a log.

Declaration

```
BOOL TLGEnumVariables (  
    HANDLE                hProject,  
    LPTSTR                lpszArchivName,  
    TLG_ENUM_VARIABLE_NAME_CALLBACK lpCallbackFunc,  
    PVOID                 lpUser,  
    LPCMN_ERROR           lpoes );
```

Parameters**hProject**

Handle of the project in which the log is located.

lpszArchivName

Pointer to the name of the log

lpCallbackFunc

Pointer to your callback function which is called for every tag entry in the log.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Log tags listed

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Related functions

TLG_ENUM_VARIABLE_NAME_CALLBACK (Page 2203)	List tags (callback)

See also

TLG_ENUM_VARIABLE_NAME_CALLBACK (Page 2203)

TLGReadVariable (Page 2207)

3.8.5.2 TLG_ENUM_VARIABLE_NAME_CALLBACK (RT Professional)**Description**

In order to be able to evaluate the tags listed by the system, you must provide a callback function of the TLG_ENUM_VARIABLE_NAME_CALLBACK type.

Declaration

```

BOOL ( * TLG_ENUM_VARIABLE_NAME_CALLBACK ) (
    LPTSTR    lpzName,
    PVOID     lpUser );

```

Parameters

lpszName

The lpszName pointer refers to the name of the first tag.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

pdecsccli.h

Related functions

TLGEnumVariables (Page 2200)	List tags

See also

TLGEnumVariables (Page 2200)

3.8.5.3 TLGEnumVariablesEx (RT Professional)

Description

The function determines all the tag names of a log.

Declaration

```

BOOL TLGEnumVariablesEx (
    LPCTSTR          lpszArchiveName,
    TLG_ENUMVARIABLES lpfnCallback,
    LPVOID           lpUser,
    PCMN_ERROR       lpError );

```

Parameters

lpszArchiveName

Pointer to the name of the log whose tags are to be listed.

lpfnCallback

Pointer to your callback function which is called for every tag of the log.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Tags listed

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

pdertcli.h

pdertcli.lib

pdertcli.dll

Related functions

TLG_ENUMVARIABLES (Page 2206)	List tags (callback)

Examples

Enum tags of a log (Page 2268) "TL02.cpp"

See also

TLG_ENUMVARIABLES (Page 2206)

Enum tags of a log (Page 2268)

3.8.5.4 TLG_ENUMVARIABLES (RT Professional)

Description

In order to be able to evaluate the data of a tag listed by the system, you must provide a callback function of the TLG_ENUMVARIABLES type.

Declaration

```
BOOL ( * TLG_ENUMVARIABLES) (
    PTLG_VARIABLE_INFO    lpvi,
    LPVOID                 lpUser );
```

Parameters

lpvi

Address of a structure of TLG_VARIABLE_INFO (Page 2185) type with the data of a tag.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value**TRUE**

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

pdertcli.h

pde_glob.h

pdertdef.h

Related functions

TLGEnumVariablesEx (Page 2202)	List tags

Examples

Enum tags of a log (Page 2268) "TL02.cpp"

See also

TLGEnumVariablesEx (Page 2202)

TLG_VARIABLE_INFO (Page 2185)

Enum tags of a log (Page 2268)

TLGEnumArchivsEx (Page 2214)

3.8.5.5 TLGReadVariable (RT Professional)**Description**

Read the parameters of a tag

Declaration

```
BOOL TLGReadVariable (  
    HANDLE          hProject,  
    LPTSTR          lpszArchivName,  
    LPTSTR          lpszVariableName,  
    PTLG_VAR_STR    lpVariable,  
    LPCMN_ERROR     lpoes );
```

Parameters

hProject

Handle of the project in which the tag to be processed is located.

lpszArchivName

Pointer to the name of the log

lpszVariableName

Pointer to the name of a tag of the log

lpVariable

Address of the TLG_VAR_STR (Page 2180) structure from which the data of the tag are read out.

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Parameters read

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Related functions

TLGEnumVariables (Page 2200)	List tags
TLGReadVariable6	Determine tag parameter

Examples

Read log (Page 2272) "TL01.cpp"

See also

TLG_VAR_STR (Page 2180)

TLGEnumVariables (Page 2200)

Read log (Page 2272)

3.8.6 Functions for processing logs (RT Professional)**3.8.6.1 TLGEnumArchives (RT Professional)****Description**

The function determines the names of all logs in the project hproject. All types of logs can be processed with this function.

Declaration

```

BOOL TLGEnumArchives (
    HANDLE                hProject,
    TLG_ENUM_ARCHIV_CALLBACK lpCallbackFunc,
    PVOID                 lpUser,
    LPCMN_ERROR            lpoes );

```

Parameters**hProject**

Handle of the project in which the logs are located.

lpCallbackFunc

Pointer to your callback function which is called for every log.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Logs listed

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Related functions

TLGReadArchiv (Page 2236)	Read log parameters
TLG_ENUM_ARCHIV_CALLBACK (Page 2210)	List logs (callback)

Examples

Enum logs (Page 2265) "TL01.cpp"

See also

TLG_ENUM_ARCHIV_CALLBACK (Page 2210)

Enum logs (Page 2265)

TLGReadArchiv (Page 2236)

3.8.6.2 TLG_ENUM_ARCHIV_CALLBACK (RT Professional)

Description

In order to be able to evaluate the logs listed by the system, you must provide a callback function of the TLG_ENUM_ARCHIV_CALLBACK type.

Declaration

```
BOOL ( * TLG_ENUM_ARCHIV_CALLBACK) (
    LPTSTR    lpzName,
    PVOID     lpUser );
```

Parameters

lpzName

The lpzName pointer refers to the name of the log.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Comment

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

pdeccli.h

Related functions

TLGEnumArchives (Page 2207)	List logs

Examples

Enum logs (Page 2265) "TL01.cpp"

See also

TLGEnumArchives (Page 2207)

Enum logs (Page 2265)

3.8.6.3 TLGEnumArchivs (RT Professional)

Description

Enumerates the existing logs Unlike TLGEnumArchivsSel the logs to be listed can only be limited by the log type.

The information of a log is provided in structures of the TLG_TABLE_INFO type with the callback function. The log name is transferred here as a table name, i.e. in the form "UA#ARCHIV#ArchivName" or "PDE#HD#Archivname#Variablenname".

Declaration

```
BOOL TLGEnumArchivs (
    DWORD          dwArchivTyp,
    TLG_ENUMTABLES lpfnCallback,
    LPVOID         lpUser,
    PCMN_ERROR     lpError );
```

Parameters

dwArchivTyp

dwArchivTyp identifies the log type:

TLG_ARCTYP_USER	User log
TLG_ARCTYP_PROCESS	Process value log
TLG_ARCTYP_COMPRESS	Compressed log
NULL	All logs are enumerated.

lpfnCallback

Pointer to your callback function which is called for every log.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Logs listed

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_ERR_SUPPLY	An error has occurred

Required files

pdertcli.h
 pde_glob.h
 pdertdef.h
 pdertcli.lib
 pdertcli.dll

Related functions

TLGEnumArchivsEx (Page 2214)	List logs
TLGEnumArchivsSel (Page 2215)	List logs
TLG_ENUMTABLES (Page 2217)	List logs (callback)

Examples

Enumerate logs (Page 2270) "TL02.cpp"

See also

TLGEnumArchivsEx (Page 2214)
 TLGEnumArchivsSel (Page 2215)

TLG_ENUMTABLES (Page 2217)

Enumerate logs (Page 2270)

3.8.6.4 TLGEnumArchivsEx (RT Professional)

Description

Enumerates the existing logs. Unlike TLGEnumArchivsSel the logs to be listed can only be limited by the log type (user, process value, compressed log).

The information of a log is provided in structures of the TLG_TABLE_INFO type with the callback function. Unlike TLGEnumArchivs not the table name of the log but the "pure" log name is transferred.

Declaration

```

BOOL TLGEnumArchivsEx (
    DWORD          dwArchivTyp,
    TLG_ENUMTABLES lpfnCallback,
    LPVOID         lpUser,
    PCMN_ERROR     lpError );

```

Parameters

dwArchivTyp

dwArchivTyp identifies the log type:

TLG_ARCTYP_U_USER	User log
TLG_ARCTYP_PROCESS	Process value log
TLG_ARCTYP_COMPRESS	Compressed log
NULL	All logs are enumerated.

lpfnCallback

Pointer to your callback function which is called for every log.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Logs are listed

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project

Required files

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

Related functions

TLG_ENUMTABLES (Page 2217)	List logs (callback)
TLGEnumArchivs (Page 2210)	List logs
TLGEnumArchivsSel (Page 2215)	List logs

See also

TLGEnumArchivsSel (Page 2215)

TLGEnumArchivs (Page 2210)

TLG_ENUMTABLES (Page 2217)

3.8.6.5 TLGEnumArchivsSel (RT Professional)**Description**

The function enumerates the existing log types. Unlike TLGEnumArchivs the logs to be listed can be limited by the logging type (circular log, sequential log) in addition to by the log type.

The information of a log is provided in structures of the `TLG_TABLE_INFO` type with the callback function. The log name is transferred here as a table name, i.e. in the form "UA#ARCHIV#ArchivName" or "PDE#HD#Archivname#Variablenname". All tags of the log are enumerated. This form is required for example in `TLGBackup`.

Declaration

```

BOOL TLGEnumArchivsSel (
    DWORD          dwArchivTyp,
    DWORD          dwSaveTyp,
    TLG_ENUMTABLES lpfnCallback,
    LPVOID         lpUser,
    PCMN_ERROR     lpError );

```

Parameters

dwArchivTyp

dwArchivTyp identifies the log type:

TLG_ARCTYP_USER	User log
TLG_ARCTYP_PROCESS	Process value log
TLG_ARCTYP_COMPRESS	Compressed log
NULL	All logs are enumerated.

dwSaveTyp

dwSaveTyp identifies the type of log:

TLG_ARCTYP_ALL	Circular and sequential logs
TLG_ARCTYP_CIRCULAR	Circular logs
TLG_ARCTYP_FOLLOW	Sequential logs

lpfnCallback

Pointer to your callback function which is called for every log.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpError

Pointer to the data of the extended error message in the `CMN_ERROR` structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The logs are listed.

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid or wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project

Required files

pdertcli.h
 pde_glob.h
 pdertdef.h
 pdertcli.lib
 pdertcli.dll

Related functions

TLG_ENUMTABLES (Page 2217)	List logs (callback)
AUTOHOTSPOT	List logs

See also

TLGEnumArchivsEx (Page 2212)
 TLGEnumArchivs (Page 2210)
 TLG_ENUMTABLES (Page 2217)
 TLG_TABLE_INFO (Page 2169)

3.8.6.6 TLG_ENUMTABLES (RT Professional)**Description**

In order to be able to evaluate the logs listed by the system, you must provide a callback function of the TLG_ENUMTABLES type.

Declaration

```
BOOL TLG_ENUMTABLES (
    LPTSTR          lpTableName,
    PTLG_TABLE_INFO lpti,
    PVOID lpUser );
```

Parameters

lpTableName

Pointer to the name of the log

lpti

Information about the database table is contained in the TLG_TABLE_INFO (Page 2169) structure.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Comment

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

pdertcli.h
pde_glob.h
pdertdef.h

pdertcli.lib

pdertcli.dll

Related functions

TLGEnumArchivs (Page 2210)	List logs
TLGEnumArchivsSel (Page 2213)	List logs

Examples

Enumerate archives "TL02.cpp"

See also

TLGEnumArchivs (Page 2210)

TLGEnumArchivsSel (Page 2213)

TLG_TABLE_INFO (Page 2169)

TLGEnumArchivsEx (Page 2212)

3.8.6.7 TLGFreeMemory (RT Professional)

Description

A memory area allocated by TLGGetArchivDataEx is freed with this function.

Declaration

```
BOOL TLGFreeMemory (  
    LPVOID lpMemory );
```

Parameters

lpMemory

You get the pointer to the memory area to be freed when calling the TLGGetArchivDataEx function in the ppTLGData parameter.

Note

The pointer must be initialized before calling TLGGetArchivDataEx() with NULL and checked before calling TLGFreeMemory() for not equal to NULL.

Return value**TRUE**

Memory is freed

FALSE

Error

Note

The pointer must be initialized before calling TLGGetArchivDataEx() with NULL and checked before calling TLGFreeMemory() for not equal to NULL.

Required files

pdertcli.h

pdertcli.lib

pdertcli.dll

Related functions

TLGGetArchivDataEx (Page 2224)	Read out data of a log

See also

TLGGetArchivDataEx (Page 2224)

3.8.6.8 TLGGetArchivData (RT Professional)**Description**

The function reads out the data from a log between two times. Unlike TLGGetArchivDataEx the values of the log tags are provided by a callback function.

A maximum 10,000 data records can be read with one call. If there are more data records between start time and end time, you only get the first 10,000.

You get the following data records by adding a millisecond to the time of the last data record and using this as a start time in a new call for TLGGetArchivData.

Declaration

```

BOOL TLGGetArchivData (
    LPTSTR                lpszArchivName,
    LPTSTR                lpszVarName,
    SYSTEMTIME            stStart,
    SYSTEMTIME            stStop,
    TLG_GETARCHIVDATA_CALLBACK lpfnCallback,
    PVOID                lpUser,
    DWORD                dwFlags,
    PCMN_ERROR            lpError );

```

Parameters

lpszArchivName

Pointer to the name of the log from which data is to be read.

lpszVarName

Pointer to the name of the log tag whose values are to be read.

stStart

Start time from which data should be read.

stStop

End time up to which the data should be read

lpfnCallback

Pointer to your callback function which is called for every measuring point to be read out.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

dwFlags

The parameter is reserved for future development and must be preset to 0.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Log data determined

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NODEFAULTSERVER	No standard server configured
TLG_API_NOSERVER	No standard server configured and no local server available

Required files

pdertcli.h
pde_glob.h
pdertdef.h
pdertcli.lib
pdertcli.dll

Related functions

TLG_GETARCHIVDATA_CALLBACK (Page 2222)	Read out data of a log (callback)
TLGGetArchivDataEx (Page 2224)	Read out data of a log

See also

TLG_GETARCHIVDATA_CALLBACK (Page 2222)

TLGGetArchivDataEx (Page 2224)

3.8.6.9 TLG_GETARCHIVDATA_CALLBACK (RT Professional)**Description**

In order to evaluate the log data listed by the system, you must provide a callback function of the TLG_GETARCHIVDATA_CALLBACK type.

Declaration

```

BOOL ( * TLG_GETARCHIVDATA_CALLBACK) (
    PTLG_GETARCHIVDATA    lpGAD,
    PVOID                 lpUser );

```

Parameters

lpGAD

The system reserves temporary storage of the length of the TLG_GETARCHIVDATA structure per log. The lpGAD pointer refers to the beginning of the first element.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only move data if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, e.g. GetMessage
 - Runtime API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

pdertcli.h
pde_glob.h
pdertdef.h

Related functions

TLGGetArchivData (Page 2218)	Read out data of a log
TLG_GETARCHIVDATA (Page 2164)	Log data of process value logs (structure) Log data of compressed logs (structure)

See also

TLGGetArchivData (Page 2218)

TLG_GETARCHIVDATA (Page 2164)

3.8.6.10 TLGGetArchivDataEx (RT Professional)**Description**

Reads out the data from a log between two times. Unlike TLGGetArchivData the values of the log tags are saved in the memory.

Note

The assigned memory in ppTlgData must be freed by TLGFreeMemory.

Declaration

```

BOOL TLGGetArchivDataEx (
    LPCTSTR                lpszArchivName,
    LPCTSTR                lpszVarName,
    SYSTEMTIME*            pstStart,
    SYSTEMTIME*            pstStop,
    PTLG_ARCHIVDATARAW*    ppTlgData,
    DWORD*                 pdwNumberOfData,
    DWORD*                 pdwFlags,
    PCMN_ERROR             lpError );

```

Parameters**lpszArchivName**

Pointer to the log name from which the data are read out.

lpszVarName

Pointer to the name of the log tag whose values are read.

stStart

Start time from which data are read.

stStop

End time up to which the data are read

If a current timer for transferring a SYSTEMTIME parameter is required, use the GetLocalTime function and not GetSystemTime. There is generally a considerable time difference between these two functions.

ppTlgData

Address of a pointer in which the address of the data of the log tag is stored. The data field is allocated by TLGGetArchivDataEx and gets structures of the TLG_ARCHIVDATARAW (Page 2150) type.

Note

The assigned memory must be freed again by TLGFreeMemory.

The pointer must be initialized before calling TLGGetArchivDataEx() with NULL and checked before calling TLGFreeMemory() for not equal to NULL.

pdwNumberOfData

Pointer to number of data records

- Before the call: Maximum number of data records to be read
- After the call: Number of read data records with the values of the log tags (of structures of the TLG_ARCHIVDATARAW type).

Maximum 10,000 data records can be read with a call. If there are more data records between start time and end time, you only get the first 10000. You get the next ones by adding a millisecond to the time of the last data record and using this as a start time in a new call for TLGGetArchivDataEx.

dwFlags

The parameter is reserved for future development and must be preset to 0.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Log data determined

FALSE

Error

Note

The memory assigned by TLGGetArchivDataEx in ppTlgData must be freed again by TLGFreeMemory.

The pointer must be initialized before calling TLGGetArchivDataEx() with NULL and checked before calling TLGFreeMemory() for not equal to NULL.

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NODEFAULTSERVER	No standard server configured
TLG_API_NOSERVER	No standard server configured and no local server available

Required files

pdertcli.h
pde_glob.h
pdertdef.h
pdertcli.lib
pdertcli.dll

Related functions

TLGGetArchivData (Page 2218)	Read out data of a log
TLGFreeMemory (Page 2217)	Free memory

See also

TLGFreeMemory (Page 2217)
TLGGetArchivData (Page 2218)
TLG_ARCHIVDATARAW (Page 2150)

3.8.6.11 TLGGetClosestTime (RT Professional)**Description**

The function determines the adjacent logging times at a given time.

Declaration

```

BOOL TLGGetClosestTime (
    LPCTSTR      lpszArchivName,
    LPCTSTR      lpszVarName,
    SYSTEMTIME*   pstTime,
    BOOL         bPrevious,
    PCMN_ERROR    lpError );

```

Parameters

lpszArchivName

Pointer to the name of the log

lpszVarName

Pointer to the name of the tag

pstTime

pstTime is used as an input and output parameter. When calling TLGGetClosestTime, pstTime contains the system time for which the adjacent logging times are to be determined. After the successful function call, pstTime contains the closest logging time.

bPrevious

bPrevious identifies whether a previous or next logging time is to be determined at the transferred system time.

TRUE	A previous logging time is determined.
FALSE	A next logging time is determined.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

pstTime contains an adjacent logging time.

FALSE

Error or no adjacent logging time could be determined.

Comment

If bPrevious is specified as TRUE and pstTime contains an existing value, exactly this time may be returned and not the previous time. If bPrevious is specified as FALSE, the next time is always supplied.

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_VARIABLE_NAME_NOT_EXIST	No tag of this name found
TLG_API_TIME_NAME_NOT_EXIST	No time object of this name found
TLG_API_NO_TIME_EXIST	No time object exists

Required files

pdertcli.h
pde_glob.h
pdertcli.lib
pdertcli.dll

Related functions

TLGGetClosestTimeEx (Page 2228)	Determine logging time

See also

TLGGetClosestTimeEx (Page 2228)

3.8.6.12 TLGGetClosestTimeEx (RT Professional)**Description**

The function determines the adjacent logging times at a given time.

Declaration

```

BOOL TLGGetClosestTimeEx (
    LPCTSTR      lpszArchivName,
    LPCTSTR      lpszVarName,
    SYSTEMTIME*   pstTime,
    BOOL          bPrevious,
    BOOL          bIgnoreInvalid,
    PCMN_ERROR    lpError );

```

Parameters

lpszArchivName

Pointer to the name of the log

lpszVarName

Pointer to the name of the tag

pstTime

pstTime is used as an input and output parameter. When calling TLGGetClosestTime, pstTime contains the system time for which the adjacent logging times are determined. After the successful function call, pstTime contains the closest logging time.

bPrevious

bPrevious identifies whether a previous or next logging time is to be determined at the transferred system time.

TRUE	A previous logging time is determined.
FALSE	A next logging time is determined.

blgnoreInvalid

Indicates whether values with the connection fault flag are to be ignored in the time determination.

TRUE	The values with connection fault are ignored and skipped.
FALSE	All values are used for the time determination so that the same function exists as in TLGGetClosestTime().

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

pstTime contains an adjacent logging time.

FALSE

Error or no adjacent logging time could be determined.

Comment

If bPrevious is specified as TRUE and pstTime contains an existing value, exactly this time may be returned and not the previous time. If bPrevious is specified as FALSE, the next time is always supplied.

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_VARIABLE_NAME_NOT_EXIST	No tag of this name found
TLG_API_TIME_NAME_NOT_EXIST	No time object of this name found
TLG_API_NO_TIME_EXIST	No time object exists

Required files

pdertcli.h
pde_glob.h
pdertcli.lib
pdertcli.dll

Related functions

TLGGetClosestTime (Page 2224)	Determine logging time

See also

TLGGetClosestTime (Page 2224)

3.8.6.13 TLGInsertArchivData (RT Professional)**Description**

The function offers the possibility of inserting any data in an existing log on the hard disk.

If you perform TlGInsertArchivData for values which were created during summer time, you have to allow for the time shift. Set the PDE_RT_DAYLIGHT flag when calling these values.

Without this flag, reading calls, e.g. TLGGetArchivData or TLGGetClosestTime return the default time.

Note

Since the correctness of the data to be inserted is not checked, there is a danger of the log being destroyed.

Declaration

```

BOOL TLGInsertArchivData (
    LPCTSTR          lpszArchivName,
    LPCTSTR          lpszVarName,
    PTLG_ARCHIVDATARAW pTlgData,
    DWORD            dwNumberOfData,
    DWORD            dwFlags,
    PCMN_ERROR        lpError );

```

Parameters

lpszArchivName

Pointer to the name of the log

lpszVarName

Pointer to the name of the tag

pTlgData

Pointer to the first of the structures of the TLG_ARCHIVDATARAW type with the tag values to be inserted in the log.

dwNumberOfData

Number of tag values (of the TLG_ARCHIVDATARAW structures) to be inserted.

dwFlags

TLG_API_FLG_FAST_INSERT	If this flag is set, the data are written much faster in optimized mode. However, a prerequisite for using this performance optimization is that the timestamps of the written data records are sorted chronologically and are always more up to date than the data records already in the log. When inserting older data records between existing data records, this flag may not be used. If values are inserted which are older than existing ones or ones which are not sorted chronologically, the older values are rejected without error messages.
TLG_API_FLG_REPLACE_INSERT	If this flag is set, the data in the log is overwritten. The "TLG_API_FLG_REPLACE_INSERT" flag cannot be used together with the "TLG_API_FLG_FAST_INSERT" flag.
TLG_API_FLG_DETAILED_ERRORCODES	If this flag is set, detailed error codes are output.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Tag values inserted in log

FALSE

Error

Error codes

If a log which is located in the main memory is specified under `lpszArchivName`, an error message `TLG_API_ERR_SUPPLY` is output.

Only logs on the hard disk can be addressed with this function. If write errors occurred for individual values, a more exact error description is returned in `CMN_ERROR`. Due to the limited length of the `CMN_ERR.szErrorText` the number of detail errors is limited to max. 57.

If a log which is located in the main memory is specified under `lpszArchivName`, an error message `TLG_API_ERR_SUPPLY` is output.

<code>dwError1</code>	<code>TLG_API_ERR_SUPPLY</code>
<code>dwError2</code>	0
<code>dwError3</code>	Line number in the source code
<code>dwError4</code>	Number of errors (max. 57)
<code>dwError5</code>	Number of data to be written (<code>dwNumberOfData</code>)
<code>szErrorText</code>	TLG-API: an error occurred, line xxx; Pos:0=0xXX,1=0xXX,2=0xXX, ... (, 56=0xXX)

When the `TLG_API_FLG_DETAILED_ERRORCODES` flag is set in `dwFlags`, the following detailed error codes are output.

Error code	Description	Value (with flag)	Value (without flag)
OK	No error	0x00	0x00
Error	General error status	0x01	0x40
Record already exists	The data record already exists and will not be overwritten.	0x02	0x80
Not allowed	Manual value input is disabled.	0x04	0x800
Limit violated	Value outside the limits	0x08	0x1000
No server	Problem with the server connection	0x10	0x10
Tag not found	Logging tag not found	0x20	0x40
Invalid flag	Prohibited combination of flags <code>TLG_API_FLG_REPLACE_INSERT</code> and <code>TLG_API_FLG_FAST_INSERT</code>	0x40	0x40
Time stamp is in the future	Values with a time stamp in the future are not allowed	0x80	0x40
Log is RAM-based	Write operations to logs in the main memory are not allowed	0x100	0x40

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NODEFAULTSERVER	No standard server configured
TLG_API_NOSERVER	No standard server configured and no local server available

Required files

pdertcli.h
 pde_glob.h
 pdertdef.h
 pdertcli.lib
 pdertcli.dll

3.8.6.14 TLGLockArchiv (RT Professional)

Description

Locks or frees a complete log. New data are not logged if a log is locked.

Declaration

```

BOOL TLGLockArchiv (
    HWND          hwnd,
    LPTSTR        lpszArchivName,
    BOOL          fLocked,
    PCMN_ERROR    lpError );

```

Parameters

hwnd

Window handle of the PDE runtime window

lpszArchivName

Pointer to the name of the log

fLocked

TRUE:	Lock log
FALSE:	Free log

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Log locked/freed

FALSE

Error

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NO_ARCHIV_EXIST	No log available
TLG_API_NODEFAULTSERVER	No standard server configured
TLG_API_NOSERVER	No standard server configured and no local server available

Required files

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

3.8.6.15 TLGLockVariable (RT Professional)**Description**

Locks or frees a tag. A locked tag is no longer updated and no longer logged.

Declaration

```

BOOL TLGLockVariable (
    HWND          hwnd,
    LPTSTR        lpszArchivName,
    LPTSTR        lpszVarName,
    BOOL          fLocked,
    PCMN_ERROR    lpError );

```

Parameters

hwnd

Window handle of the PDE runtime window

lpszArchivName

Pointer to the name of the log

To select a certain server the "ServerPrefix::" can be prefixed to the log name.

lpszVarName

Pointer to the name of the tag

fLocked

TRUE:	Lock tag
FALSE:	Free tag

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Tag freed / locked

FALSE

Error

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NO_ARCHIV_EXIST	No log available

TLG_API_NODEFAULTSERVER	No standard server configured
TLG_API_NOSERVER	No standard server configured and no local server available

Required files

pdertcli.h
pde_glob.h
pdertcli.lib
pdertcli.dll

3.8.6.16 TLGReadArchiv (RT Professional)

Description

Read the parameters of an existing log in the hProject project. All types of logs can be processed with this function.

Declaration

```
BOOL TLGReadArchiv (  
    HANDLE          hProject,  
    LPTSTR          lpszArchivName,  
    PTLG_ARCHIV_STR lpArchiv,  
    LPCMN_ERROR     lpoes );
```

Parameters

hProject

Handle of the project in which the log to be processed is located.

lpszArchivName

Name of the log to be processed.

lpArchiv

Address of the TLG_ARCHIV_STR (Page 2147) structure in which the parameters of the log are to be saved.

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Log contents read

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Related functions

TLGEnumArchives (Page 2207)	List logs
TLGReadArchiv6	Read log parameters

Examples

Read log (Page 2272)"TL01.cpp"

See also

TLGEnumArchives (Page 2207)

Read log (Page 2272)

TLG_ARCHIV_STR (Page 2147)

3.8.7 Functions for trend and table views (RT Professional)**3.8.7.1 TLGCloseWindow (RT Professional)****Description**

An application window is closed with the function.

Declaration

```
BOOL TLGCloseWindow (  
    HWND          hwnd,  
    PCMN_ERROR    lpError );
```

Parameters

hwnd

Handle to the parent window of the application to be closed.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Application window closed

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply

Required files

pdertcli.h
pde_glob.h
pdertcli.lib
pdertcli.dll

3.8.7.2 TLGDrawCurvesInDC (RT Professional)

Description

Trends can be shown in an application window with this function.

Declaration

```

BOOL TLGDrawCurvesInDC (
    HDC          hDC,
    PRECT        lprect,
    PTLG_PROT_CURVE_INFOS lpci,
    DWORD        dwNumberOfCurves,
    LPCMN_ERROR  lpError );

```

Parameters

hDC

Handle to the parent window in which the trend is to be output

lprect

Pointer to the Windows-specific structure of the RECT type with the size data of the window.

lpci

Pointer to the first structure of the TLG_PROT_CURVE_INFOS (Page 2166) type with information about how the trends are to be output.

dwNumberOfCurves

Number of the structures (number of trends to be shown) transferred in lpci.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Trends shown

FALSE

Error

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NOT_SUPPORTED	This function is not supported

Required files

pdertcli.h
pde_glob.h
pdertdef.h
pdertcli.lib
pdertcli.dll

See also

TLG_PROT_CURVE_INFOS (Page 2166)

3.8.7.3 TLGInsertTemplateItem (RT Professional)

Description

The function inserts a new entry for an existing window template, either a trend template or a table template.

Declaration

```
BOOL TLGInsertTemplateItem (  
    LPTSTR                lpszTemplateName,  
    PTLG_TEMPLATEITEM_INFO lpptii,  
    LPCMN_ERROR            lpError );
```

Parameters

lpszTemplateName

Pointer to the name of the window template to which a trend or table template is to be added.

lpptii

Pointer to a structure of the TLG_TEMPLATEITEM_INFO (Page 2173) type with the data of the trend or table template to be added.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Window template extended

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NOT_SUPPORTED	This function is not supported

Required files

pdertcli.h
pde_glob.h
pdertdef.h
pdertcli.lib
pdertcli.dll

Examples

Edit curve template - Example 1 (Page 2259) "TL02.cpp"

See also

TLG_TEMPLATEITEM_INFO (Page 2173)
Edit curve template - Example 1 (Page 2259)

3.8.7.4 TLGPressToolBarButton (RT Professional)**Description**

You can trigger functions linked with the toolbar buttons with this function.

Declaration

```

BOOL TLGPressToolBarButton (
    LPTSTR      lpszTemplate,
    DWORD       dwButtonID,
    PCMN_ERROR  lpError );

```

Parameters

lpszTemplate

Name of the template

dwButtonID

Code number for the button to be triggered:

Constants for trends and tables	
TLG_BASIC_BTN_HELP	Open online help
TLG_BASIC_BTN_DLG	Open parameter assignment dialog
TLG_BASIC_BTN_FIRST	Show first data records of a log.
TLG_BASIC_BTN_PREV	Scroll back in the log.
TLG_BASIC_BTN_NEXT	Scroll forward in the log.
TLG_BASIC_BTN_LAST	Show last data records of a log.
TLG_BASIC_BTN_STARTSTOP	Activate or deactivate the updated view of a trend or table.
TLG_BASIC_BTN_PREV_ITEM	Previous trend in the foreground or table column first.
TLG_BASIC_BTN_NEXT_ITEM	Next trend in the foreground or table column first.
TLG_BASIC_BTN_ARC_VAR_SELECT	Opens the log tag selection dialog.
TLG_BASIC_BTN_ITEM_SELECT	Opens the selection dialog for trends or table columns.
TLG_BASIC_BTN_TIME_SELECT	Opens the selection dialog for the display time range.

Constants for tables	
TLG_TABLE_BTN_EDIT	Activate or deactivate the edit function.

Constants for trends	
TLG_CURVE_BTN_ZOOMIN	Activate the zoom function for enlarging the section.
TLG_CURVE_BTN_ZOOMOUT	Deactivate the zoom function.
TLG_CURVE_BTN_1_TO_1	Switch to normal trend view.
TLG_CURVE_BTN_LINEAL	Activate or deactivate the ruler.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
IDS_API_ERROR_WINDOW_NOT_FOUND	Button (window) could not be found
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NO_TYP_CHANGE_ALLOWED	Type change is not allowed

Required files

pdertcli.h
 pde_glob.h
 pdertdef.h
 pdertcli.lib
 pdertcli.dll

3.8.7.5 TLGSetRulerWindowVisible (RT Professional)**Description**

The data window of the read line is shown or hidden with this function.

Declaration

```
BOOL TLGSetRulerWindowVisible (
    LPTSTR      lpszTemplateName,
    BOOL        bShowRulerWindow,
    LPCMN_ERROR lpError );
```

Parameters**lpszTemplateName**

Pointer to the name of the window template

bShowRulerWindow

TRUE:	The read line with data window is shown.
FALSE:	The read line without data window is shown.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The parameter for the visibility of the read line data window was re-entered.

FALSE

Error

Comment

The function is not effective in the f(x) trend view and table view until the appropriate image with the object is displayed.

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
IDS_API_ERROR_WINDOW_NOT_FOUND	Button (window) could not be found
TLG_API_INVALID_PARAM	invalid/wrong parameter supply

Required files

pdertcli.h
pde_glob.h
pdertdef.h
pdertcli.lib
pdertcli.dll

3.8.7.6 TLGShowWindow (RT Professional)**Description**

The function influences the display form of the window.

Declaration

```
BOOL TLGShowWindow (  
    HWND          hwnd,  
    DWORD          dwFlags,  
    PCMN_ERROR     lpError );
```

Parameters**hwnd**

Window handle of the runtime window

dwFlags

The same flags apply here as in ::ShowWindow(HWND hwnd, int nCmdShow) as they are defined in WinUser.h (SW_SHOW, SW_HIDE, ...).

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Application window shown

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply

Required files

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

WinUser.h

3.8.8 Functions for processing time systems (RT Professional)**3.8.8.1 TLGEnumTime (RT Professional)****Description**

Enumerates the names of all detection and logging times of a project

Declaration

```
BOOL TLGEnumTime (
    HANDLE                hProject,
    TLG_ENUM_TIME_NAME_CALLBACK lpCallbackFunc,
    PVOID                lpUser,
    LPCMN_ERROR           lpoes );
```

Parameters

- hProject**
Handle of the project in which the time objects to be listed are located.
- lpCallbackFunc**
Pointer to your callback function which is called for every existing time object.
- lpUser**
Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.
- lpoes**
Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

- TRUE**
Time objects listed
- FALSE**
Error

Required files

- pdecsccli.h
- pdecsccli.lib
- pdecsccli.dll

Related functions

TLG_ENUM_TIME_NAME_CALLBACK (Page 2247)	List time objects (callback)

Examples

Enum all acquisition and logging times (Page 2262) "TL01.cpp"

See also

TLG_ENUM_TIME_NAME_CALLBACK (Page 2247)

Enum all acquisition and logging times (Page 2262)

3.8.8.2 TLG_ENUM_TIME_NAME_CALLBACK (RT Professional)

Description

In order to be able to evaluate the time objects listed by the system, you must provide a callback function of the TLG_ENUM_TIME_NAME_CALLBACK type.

Declaration

```
BOOL ( * TLG_ENUM_TIME_NAME_CALLBACK) (  
    LPTSTR    lpzName,  
    PVOID     lpUser );
```

Parameters

lpzName

The lpzName pointer refers to the name of the first time object.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

pdeccli.h

Related functions

TLGEnumTime (Page 2243)	List time objects

Examples

Enum all acquisition and logging times (Page 2262) "TL01.cpp"

See also

Enum all acquisition and logging times (Page 2262)

TLGEnumTime (Page 2243)

3.8.8.3 TLGEnumTimes (RT Professional)**Description**

Enumerates all detection and logging times.

Declaration

```
BOOL TLGEnumTimes(  
    TLG_ENUMTIMES_CALLBACK    lpfnCallback,  
    LPVOID                    lpUser,  
    PCMN_ERROR                 lpError );
```

Parameters

lpfnCallback

Pointer to your callback function

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Time objects listed

FALSE

Error

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project

Required files

pdertcli.h
pde_glob.h
pdertdef.h
pdertcli.lib
pdertcli.dll

Related functions

TLG_ENUMTIMES_CALLBACK (Page 2250)	List time objects (callback)

See also

TLG_ENUMTIMES_CALLBACK (Page 2250)

3.8.8.4 TLG_ENUMTIMES_CALLBACK (RT Professional)

Description

In order to be able to evaluate the time objects listed by the system, you must provide a callback function of the TLG_ENUMTIMES_CALLBACK type.

Declaration

```
BOOL ( * TLG_ENUMTIMES_CALLBACK) (
    PTLG_TIMEDATA    lpTime,
    PVOID             lpUser );
```

Parameters

lpTime

The system reserves temporary storage of the length of the TLG_TIMEDATA (Page 2176) structure per time object. The lpTime pointer refers to the beginning of the first element.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
 - API functions from the same DLL
 - Enumerations that call other enumerations
-

Required files

pdertcli.h

pde_glob.h

pdertdef.h

Related functions

TLGEnumTimes (Page 2246)	List time objects

See also

TLGEnumTimes (Page 2246)

TLG_TIMEDATA (Page 2176)

3.8.8.5 TLGReadTime (RT Professional)**Description**

Read the parameters of an already created time object

Declaration

```

BOOL TLGReadTime (
    HANDLE          hProject,
    LPTSTR          lpszTimeName,
    PTLG_TIME_STR   lpTime,
    LPCMN_ERROR      lpoes );

```

Parameters**hProject**

Handle of the project in which the time object to be processed is located.

lpszTimeName

Pointer to the name of the time object

lpTime

Address of the TLG_TIME_STR (Page 2175) structure with the data of the time object.

lpoes

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Parameters of the time object read

FALSE

Error

Required files

pdecsccli.h

pdecsccli.lib

pdecsccli.dll

Examples

Read parameters of time object (Page 2276) "TL01.cpp"

See also

TLG_TIME_STR (Page 2175)

Read parameters of time object (Page 2276)

3.8.9 Functions for saving and restoring (RT Professional)

3.8.9.1 TLGEnumBackupEntries (RT Professional)

Description

This function is no longer supported and supplies FALSE and the error code TLG_API_NOT_SUPPORTED as a return value.

Declaration

```
BOOL TLGEnumBackupEntries (
    LPTSTR                lpszArchivName,
    TLG_ENUMBACKUP_ENTRIES lpfnCallback,
    LPVOID                 lpUser,
    PCMN_ERROR             lpError );
```

Parameters

lpArchivName

Pointer to the name of the log

lpfnCallback

Pointer to your callback function which is called for every existing backup.

lpUser

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Backups listed

FALSE

Error

Error messages

TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NOT_SUPPORTED	This function is not supported

Required files

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

Related functions

TLG_ENUMBACKUP_ENTRIES (Page 2254)	List backups (callback)

See also

TLG_ENUMBACKUP_ENTRIES (Page 2254)

3.8.9.2 TLG_ENUMBACKUP_ENTRIES (RT Professional)

Description

The TLGEnumBackupEntries function is no longer supported. In order to be able to evaluate the backups listed by the system, you must provide a callback function of the TLG_ENUMBACKUP_ENTRIES type.

Declaration

```
BOOL ( * TLG_ENUMBACKUP_ENTRIES) (
    PTLG_BACKUP_TABLE_INFO    lpbti,
    PVOID                     lpUser );
```

Parameters

lpbti

The system reserves temporary storage of the length of the TLG_BACKUP_TABLE_INFO (Page 2153) structure per backup. The lpbti pointer refers to the beginning of the first element.

lpUser

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value

TRUE

The enumeration is continued.

FALSE

The enumeration is cancelled.

Comment

Note

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

Required files

pdertcli.h
pde_glob.h
pdertdef.h

Related functions

TLGEnumBackupEntries (Page 2250)	List backups

See also

TLGEnumBackupEntries (Page 2250)
TLG_BACKUP_TABLE_INFO (Page 2153)

3.8.9.3 TLGExport (RT Professional)

Description

Parts of a log can be exported with the function. The data records to be exported are selected by the TLG_IO_BACKUP_SELECT structure.

Declaration

```

BOOL TLGExport (
    LPTSTR                lpszArchivName,
    LPTSTR                lpszFileName,
    PTLG_IO_BACKUP_SELECT lpibs,
    DWORD                 dwJobFlags,
    DWORD                 dwFormatFlags,
    PCMN_ERROR             lpError );

```

Parameters

lpszArchivName

Pointer to the name of the log The name must be specified made up in the form "log name\tag name" zusammengesetzt anzugeben.

The parts of the name can be determined with TLGEnumArchivsEx and TLGEnumVariablesEx.

lpszFileName

Pointer to the name of the file into which the data are to be exported.

lpibs

Address of the transfer structure TLG_IO_BACKUP_SELECT (Page 2165) for the selection parameters.

dwJobFlags

The parameter is provided for later upgrades and must be supplied with 0L.

dwFormatFlags

Format specifier:

TLG_BAKFMT_CSV:	CSV format (Comma Separated Value)

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Log data exported

FALSE

Error

Comment

This function is only useful for sequential logs, circular logs and compressed logs.

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project

Required files

pdertcli.h
 pde_glob.h
 pdertdef.h
 pdertcli.lib
 pdertcli.dll

See also

TLG_IO_BACKUP_SELECT (Page 2165)

3.8.9.4 TLGGetBackupSize (RT Professional)**Description**

The function is no longer supported and supplies FALSE and the error code TLG_API_NOT_SUPPORTED as a return value.

Declaration

```

BOOL TLGGetBackupSize (
    LPTSTR          lpszArchivName,
    DWORD*          lpdwSizeOfTable,
    PTLG_IO_BACKUP_SELECT lpibs,
    DWORD           dwJobFlags,
    DWORD           dwFormatFlags,
    PCMN_ERROR      lpError );

```

Parameters**lpszArchivName**

Pointer to the name of the log

lpdwSizeOfTable

Address of a DWORD in which the size of the selected data in bytes is written.

lpibs

Address of the transfer structure TLG_IO_BACKUP_SELECT (Page 2165) for the selection parameters.

dwJobFlags

Job-specific identification, following are possible:

TLG_BACKUP_EXPORT:	Simple backup without deleting the source data
TLG_BACKUP_EVACUATE:	Export with deletion of the sources

dwFormatFlags

Format specifier:

TLG_BAKFMT_CSV:	CSV format (Comma Separated Value)-

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Size of data determined

FALSE

Error

Error messages

TLG_API_ERR_SUPPLY	An error has occurred
TLG_API_INVALID_PARAM	invalid/wrong parameter supply
TLG_API_NO_INTERFACE	Access to lower level COM interfaces failed
TLG_API_NO_CONNECTION	No connection established with an activated project
TLG_API_NOT_SUPPORTED	This function is not supported

Required files

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

See also

TLG_IO_BACKUP_SELECT (Page 2165)

3.8 Tag and log functions (RT Professional)

```

// scripts
BOOL ret = FALSE;
CMN_ERROR Error;
TCHAR szText[255];
HWND hwndParent = NULL;
TLG_TEMPLATEITEM_INFO TemplateItem;
memset(&Error, 0, sizeof(CMN_ERROR));
ret = TLGConnect(hwndParent, &Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGInsertTemplateItem: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
}
else
{
    memset(&TemplateItem, 0, sizeof(TLG_TEMPLATEITEM_INFO));
    _tcsncpy_s(TemplateItem.szArchivName, _countof(TemplateItem.szArchivName),
        _T("PLSMWA"), _TRUNCATE); // archive name
    _tcsncpy_s(TemplateItem.szTemplateName, _countof(TemplateItem.szTemplateName),
        _T("KV_PID1"), _TRUNCATE); // template for curve window
    _tcsncpy_s(TemplateItem.szVariableName, _countof(TemplateItem.szVariableName),
        _T("VPID1_X"), _TRUNCATE); // DM variable name
    _tcsncpy_s(TemplateItem.szDMVariableName, _countof(TemplateItem.szDMVariableName),
        _T("VPID1_X"), _TRUNCATE);
    _tcsncpy_s(TemplateItem.szTemplateItemName,
        _countof(TemplateItem.szTemplateItemName), _T("KPID3_X"), _TRUNCATE); // curve name
    TemplateItem.dwTemplateType = TLG_TEMPLATE_CURVE;
    TemplateItem.dwTemplateItemType = TLG_TEMPLATEITEM_CURVE;
    TemplateItem.dwReadAccessLevel = 1;
    TemplateItem.dwWriteAccessLevel = 1;
    TemplateItem.dwArchivTyp = TLG_ARCTYP_PROCESS; // _USER, _PROCESS, _COMPRESS
    _tcsncpy_s(TemplateItem.szTimeNameRange, _countof(TemplateItem.szTimeNameRange),
        _T("500 ms"), _TRUNCATE);
    TemplateItem.fVisible = TRUE;
    TemplateItem.tplInfo.tplCurve.csx.dwDataType = TLG_DATATYP_TIMERANGE;
    TemplateItem.tplInfo.tplCurve.csx.fAutoRange = TRUE;
    TemplateItem.tplInfo.tplCurve.csx.fActualize = TRUE;
    TemplateItem.tplInfo.tplCurve.csx.dwBufferSize = 10;
    TemplateItem.tplInfo.tplCurve.csy.dwDataType = TLG_DATATYP_BLOCKDATA;
    TemplateItem.tplInfo.tplCurve.csy.fAutoRange = TRUE;
    TemplateItem.tplInfo.tplCurve.csy.crColor = COLOR_RED;
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGInsertTemplateItem(TemplateItem.szTemplateName, &TemplateItem, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGInsertTemplateItem: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("InsertItem:
TemplateItemName= %s  "), TemplateItem.szTemplateItemName);
    }
}

```

```

    }
    ODKTrace(szText);
    // Disconnect
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGDisconnect(&Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGDisconnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

[TLGInsertTemplateItem \(Page 2238\)](#)

[TLGConnect \(Page 2191\)](#)

[TLGDisconnect \(Page 2193\)](#)

3.8.10.2 Enum all acquisition and logging times (RT Professional)

Example

```

/ =====
// =====
// : Modul with examples to TagLogging-API
// *****
// Copyright (C) 1995/96 SIEMENS AG, AUT 913 All rights reserved
// *****
#include "stdafx.h" // if MFC classes
#include "stdlib.h"
#include "TL01.h"
#include "dm01.h"
// IMPLEMENTATION
// =====
// 4. Callback
// =====
//{{ODK_EXAMPLE}Enum all acquisition and archiving times (TLG)}
//{{FUNCTION}TLGCSCConnect (TLG)}
//{{FUNCTION}TLGCSCDisconnect (TLG)}
//{{FUNCTION}TLGEnumTime (TLG)}
//{{FUNCTION}TLG_ENUM_TIME_NAME_CALLBACK (TLG)}
//{{FUNCTION}(END)}
// =====
// Function: TLGEnumTime(void) ODK TL CS
// =====
// : Enum all acquisition and archiving times
// :
// =====
BOOL MyTLGEnumTimeNameCallback (LPTSTR szTime, PVOID lpUser)
{
    TCHAR szText[255];
    sprintf(szText, "...Enum Time %s", szTime);
    ODKTrace(szText);
    //printf("%s\r\n", szText);
    return TRUE;
}

void MyTLGEnumTime(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    TCHAR szText[255];
    HANDLE hProject;
    CMN_ERROR Error;
    PVOID pUser = NULL;
    sprintf(szText, "TLGEnumTime");
    ODKTrace(szText);
    //printf("%s\r\n", szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGCSCConnect(NULL, &Error);
    if (FALSE == ret)

```

```

{
    sprintf(szText, "Error in TLGCSConnect: E1= 0x%08lx ; E2= 0x%08lx ; %s",
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    //printf("%s\r\n",szText);
}
else
{
    sprintf(szText, " TLGCSConnect");
    ODKTrace(szText);
    //printf("%s\r\n",szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
    if (FALSE == ret)
    {
        sprintf(szText, "Error in TLGOpenProject: E1= 0x%08lx ; E2= 0x%08lx ; %s",
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        //printf("%s\r\n",szText);
    }
    else //TLGOpenProject OK
    {
        sprintf(szText, " TLGOpenProject");
        ODKTrace(szText);
        //printf("%s\r\n",szText);
        // read projected archiving times
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGEnumTime(hProject, MyTLGEnumTimeNameCallback, pUser, &Error);
        if (FALSE == ret)
        {
            sprintf(szText, "Error in TLGEnumTime: E1= 0x%08lx ; E2= 0x%08lx ; %s",
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            sprintf(szText, " TLGEnumTime");
        }
        ODKTrace(szText);
        //printf("%s\r\n",szText);
        //TLGCloseProject();
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGCloseProject(hProject, &Error);
        if(FALSE == ret)
        {
            sprintf(szText, "Error in TLGCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s",
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            sprintf(szText, " TLGCloseProject");
        }
        ODKTrace(szText);
        //printf("%s\r\n",szText);
    }
    // Disconnect
    memset(&Error,0,sizeof(CMN_ERROR));
}

```

```
ret = TLGCSDisConnect(&Error);
if (FALSE == ret)
{
    sprintf(szText, "Error in TLGCSDisConnect: E1= 0x%08lx ; E2= 0x%08lx ; %s",
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    sprintf(szText, " TLGCSDisConnect");
}
ODKTrace(szText);
//printf("%s\r\n", szText);
}
}
//{{ODK_EXAMPLE} (END) }
```

See also

[TLGCSCConnect \(Page 2186\)](#)

[TLG_ENUM_TIME_NAME_CALLBACK \(Page 2245\)](#)

[TLGEnumTime \(Page 2243\)](#)

[TLGDisconnect \(Page 2193\)](#)

[TLGCSDisConnect \(Page 2189\)](#)

3.8.10.3 Enum logs (RT Professional)

Example

```
//{{ODK_EXAMPLE}Enum archives (TLG)}
//{{FUNCTION}TLGCSCConnect (TLG)}
//{{FUNCTION}TLGCSDisConnect (TLG)}
//{{FUNCTION}TLGOpenProject (TLG)}
//{{FUNCTION}TLGCloseProject (TLG)}
//{{FUNCTION}TLGEnumArchives (TLG)}
//{{FUNCTION}TLG_ENUM_ARCHIV_CALLBACK (TLG)}
//{{FUNCTION}(END)}
// =====
// Function: TLGEnumArchives(void) ODK TL CS
// =====
// : Enum archives
// :
// =====
BOOL MyTLGEnumArchivCallback(LPTSTR szArchives, PVOID lpUser)
{
    lpUser;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Enum Archive=%s "),szArchives);
    ODKTrace(szText);
    return TRUE;
}

void MyTLGEnumArchives(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HANDLE hProject;
    PVOID pUser = NULL;
    TCHAR szText[255];
    TCHAR szArchivName[255];
    _tcsncpy_s(szArchivName, _countof(szArchivName), _T("PLSMWA"), _TRUNCATE);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    //TLGConnect
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSCConnect(NULL, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSCConnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        //TLGOpenProject
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
        if (FALSE == ret)
    }
}
```

3.8 Tag and log functions (RT Professional)

```

    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGOpenProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        // TLGEnumArchives
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = TLGEnumArchives(hProject, MyTLGEnumArchivCallback, pUser, &Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGEnumArchives: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGEnumArchives"));
        }
        ODKTrace(szText);
    }
    //CloseProject
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGCloseProject(hProject, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCloseProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGCloseProject"));
    }
    ODKTrace(szText);
}
// Disconnect
memset(&Error, 0, sizeof(CMN_ERROR));
ret = TLGCSDisConnect(&Error);
if (FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSDisConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
    Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGCSDisConnect"));
}
    ODKTrace(szText);
}
//{{ODK_EXAMPLE} (END) }

```


See also

[TLGCSCConnect \(Page 2186\)](#)

[TLGOpenProject \(Page 2196\)](#)

[TLGCloseProject \(Page 2194\)](#)

[TLGEnumArchives \(Page 2207\)](#)

[TLG_ENUM_ARCHIV_CALLBACK \(Page 2208\)](#)

[TLGCSDisConnect \(Page 2189\)](#)

3.8.10.4 Enum tags of a log (RT Professional)

Example

```

//{{ODK_EXAMPLE}Enum variables of a archive (TLG)}
//{{FUNCTION}TLG_ENUM_PROJECT_NAME_CALLBACK (TLG)}
//{{FUNCTION}TLGEnumVariablesEx (TLG)}
//{{FUNCTION}TLG_ENUMVARIABLES (TLG)}
//{{FUNCTION}(END)}
// =====
// Function: MyTLGEnumVariablesEx(void) ODK TL CS
// =====
// : Edit archive variables
// :
// =====
BOOL MyTLGEnumVariablesExCallback(PTLG_VARIABLE_INFO lpvi, LPVOID lpUser)
{
    lpUser;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Enum Variables %d %s "), lpvi->dwVariableTyp, lpvi->szVariableName);
    ODKTrace(szText);
    return TRUE;
}

void MyTLGEnumVariablesEx(void)
{
    BOOL ret = FALSE;
    TCHAR szText[255];
    CMN_ERROR Error;
    HANDLE hProject = NULL;
    TCHAR szArchivName[255];
    HWND hwndParent = NULL;
    VOID* pUser = NULL;
    _tcsncpy_s(szArchivName, _countof(szArchivName), _T("PLSMWA"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGConnect(hwndParent, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGConnect: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGConnect"));
        ODKTrace(szText);

        // Info
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = TLGEnumVariablesEx (szArchivName, MyTLGEnumVariablesExCallback,
            pUser, &Error );
    }
}

```

```

    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGEnumVariablesEx: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGEnumVariablesEx: OK"));
    }
    ODKTrace(szText);

    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGDisconnect(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGDisconnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}

```

See also

TLGOpenProject (Page 2196)

TLGEnumVariablesEx (Page 2202)

TLG_ENUMVARIABLES (Page 2204)

3.8.10.5 Enumerate logs (RT Professional)

Example

```

//{{ODK_EXAMPLE}Enumerate archives (TLG)}
//{{FUNCTION}TLGConnect (TLG)}
//{{FUNCTION}TLGDisconnect (TLG)}
//{{FUNCTION}TLGEnumArchivs (TLG)}
//{{FUNCTION}TLG_ENUMTABLES (TLG)}
//{{FUNCTION}(END)}
// =====
// Function: MyTLGEnumArchivs(void) ODK TL RT
// =====
// : Enumerate archives
// =====
BOOL MyTLGEnumTablesCallback(LPTSTR lpTableName, PTLG_TABLE_INFO lpti, PVOID lpUser)
{
    lpUser;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TableName=%s"), lpTableName);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T(" ...ArchivTyp=%d"), lpti->dwArchivTyp);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T(" ...ArchivName=%s"), lpti->szArchivName);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T(" ...dwSaveTyp=%d"), lpti->dwSaveTyp);
    ODKTrace(szText);
    return TRUE;
}

void MyTLGEnumArchivs(void)
{
    BOOL ret = FALSE;
    //DWORD StartTime = 0;
    //DWORD StopTime = 0;
    void* lpUser = NULL;
    CMN_ERROR Error;
    TCHAR szText[255];
    HWND hwndParent = NULL;
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGConnect(hwndParent, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGConnect: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGConnect"));
    }
}

```

```

ODKTrace(szText);

memset(&Error,0,sizeof(CMN_ERROR));
ret = TLGEnumArchivs(TLG_ARCTYP_PROCESS, MyTLGEnumTablesCallback, lpUser, &Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGEnumArchivs:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGEnumArchivs: OK"));
}
ODKTrace(szText);
memset(&Error,0,sizeof(CMN_ERROR));
ret = TLGDisconnect(&Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGDisconnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
}
}
}
//{{ODK_EXAMPLE}(END)}

```

See also

TLGConnect (Page 2191)

TLGDisconnect (Page 2193)

TLGEnumArchivs (Page 2210)

3.8.10.6 Read log (RT Professional)

Example

```

//{{ODK_EXAMPLE}Read archive (TLG)}
//{{FUNCTION}TLGCSConnect (TLG)}
//{{FUNCTION}TLGCSDisconnect (TLG)}
//{{FUNCTION}TLGOpenProject (TLG)}
//{{FUNCTION}TLGCloseProject (TLG)}
//{{FUNCTION}TLGReadArchiv (TLG)}
//{{FUNCTION}TLGReadVariable (TLG)}
//{{FUNCTION}(END)}
// =====
// Function: TLGReadArchiv(void) ODK TL CS
// =====
// : Read archive
// :
// =====
void MyTLGReadArchiv(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HANDLE hProject;
    //PVOID pUser = NULL;
    TCHAR szText[255];
    TCHAR szArchivName[255];
    TCHAR szVarName[255];
    TLG_ARCHIV_STR TLGArchivStruct;
    TLG_VAR_STR TLGVarStruct;
    _tcsncpy_s(szArchivName, _countof(szArchivName), _T("PLSMWA"), _TRUNCATE);
    _tcsncpy_s(szVarName, _countof(szVarName), _T("X"), _TRUNCATE);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSConnect(NULL, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSConnect: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
    }
}

```

```

{
    // read
    memset(&TLGArchivStruct,0,sizeof(TLG_ARCHIV_STR));
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGReadArchiv(hProject, szArchivName, &TLGArchivStruct, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGReadArchiv:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGReadArchiv"));
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  ArcName=%s Comments=
%s dwTyp=0x%04x dwRecSize=%d"),
            TLGArchivStruct.szName,
            TLGArchivStruct.szComment,
            TLGArchivStruct.dwTyp,
            TLGArchivStruct.dwRecordSize);
        ODKTrace(szText);
    }
    memset(&TLGVarStruct, 0, sizeof(TLG_VAR_STR));
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGReadVariable(hProject, szArchivName, szVarName, &TLGVarStruct, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGReadVariable: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGReadVariable"));
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  szVarName=%s"),
            TLGVarStruct.szVarName);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  szProcName=%s"),
            TLGVarStruct.szProcName);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  dwVarType=0x%04X"),
            TLGVarStruct.dwVarType);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  dwArchivStyle=0x
%04X"), TLGVarStruct.dwArchivStyle);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  szWriteBackTo=%s"),
            TLGVarStruct.szWriteBackTo);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  dwSupply=0x%04X"),
            TLGVarStruct.dwSupply);
        ODKTrace(szText);
    }
}

```

3.8 Tag and log functions (RT Professional)

```

        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    fLocked=%d"),
TLGVarStruct.fLocked);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    szComment=%s"),
TLGVarStruct.szComment);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    szRecordCycle=%s"),
TLGVarStruct.szRecordCycle);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    szArchivCycle=%s"),
TLGVarStruct.szArchivCycle);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    dwMultiple=0x%04X"),
TLGVarStruct.dwMultiple);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    dwValueFlow=0x
%04X"), TLGVarStruct.dwValueFlow);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    dwValueFollow=0x
%04X"), TLGVarStruct.dwValueFollow);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    dwSaveByFault=0x
%04X"), TLGVarStruct.dwSaveByFault);
        ODKTrace(szText);
    }
}
//CloseProject
memset(&Error,0,sizeof(CMN_ERROR));
ret = TLGCloseProject(hProject,&Error);
if(FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCloseProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGCloseProject"));
}
    ODKTrace(szText);
}
// Disconnect
memset(&Error,0,sizeof(CMN_ERROR));
ret = TLGCSDisConnect(&Error);
if (FALSE == ret)
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSDisConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
}
else
{
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGCSDisConnect"));
}
    ODKTrace(szText);
}

```



```
// {{ODK_EXAMPLE} (END) }
```

See also

TLGOpenProject (Page 2196)
TLGCloseProject (Page 2194)
TLGCSCConnect (Page 2186)
TLGReadVariable (Page 2205)
TLGReadArchiv (Page 2234)
TLGCSDisConnect (Page 2189)

3.8.10.7 Read parameters of time object (RT Professional)

Example

```

//{{ODK_EXAMPLE}Read parameters of time object (TLG)}
//{{FUNCTION}TLGCSCConnect (TLG)}
//{{FUNCTION}TLGCSDisConnect (TLG)}
//{{FUNCTION}TLGReadTime (TLG)}
//{{FUNCTION}TLGOpenProject (TLG)}
//{{FUNCTION}TLGCloseProject (TLG)}
//{{FUNCTION}(END)}
// =====
// Function: TLGReadTime(void) ODK TL CS
// =====
// : Reads the parameters of an already existing time object.
// :
// =====
void MyTLGReadTime(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    TCHAR szText[255];
    HANDLE hProject;
    CMN_ERROR Error;
    TLG_TIME_STR TLGTimeStruct;
    TCHAR szTime[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGReadTime"));
    ODKTrace(szText);
    MyDMenuOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSCConnect(NULL, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSCConnect: E1= 0x
%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            // read properties of selected time
            memset(&Error,0,sizeof(CMN_ERROR));
            memset(&TLGTimeStruct, 0, sizeof(TLG_TIME_STR));

```

```

    _tcsncpy_s(szTime, _countof(szTime), _T("500 ms"), _TRUNCATE);
    ret = TLGReadTime(hProject, szTime, &TLGTimeStruct, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGReadTime:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...szTimeName = %s
"), TLGTimeStruct.szTimeName);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...dwBasis = %10d
Millisec"), TLGTimeStruct.dwBasis);
        ODKTrace(szText);
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...dwFactor = %10d
"), TLGTimeStruct.dwFactor);
        ODKTrace(szText);
    }
    // save changes permanent
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGSaveProject(hProject, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGSaveProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGSaveProject"));
    }
    ODKTrace(szText);
    //TLGCloseProject();
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = TLGCloseProject(hProject, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGCloseProject"));
    }
    ODKTrace(szText);
}
// Disconnect
memset(&Error, 0, sizeof(CMN_ERROR));
ret = TLGCSDisConnect(&Error);
if (FALSE == ret)
{

```

3.9 Recipe functions (RT Professional)

```

        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSDisConnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
        Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("    TLGCSDisConnect"));
    }
    ODKTrace(szText);
}
}
//{{ODK_EXAMPLE}(END)}

```

See also

[TLGOpenProject \(Page 2196\)](#)
[TLGCloseProject \(Page 2194\)](#)
[TLGCSCConnect \(Page 2186\)](#)
[TLGReadTime \(Page 2249\)](#)
[TLGCSDisConnect \(Page 2189\)](#)

3.9 Recipe functions (RT Professional)

3.9.1 Basics (RT Professional)

3.9.1.1 Overview of the functions (RT Professional)

Overview

uaSetLocalEvents (Page 2289)	Sets the local events.
uaArchiveClose (Page 2301)	Close connection to current user archive
uaArchiveDelete (Page 2302)	Delete data records in the user archive
uaArchiveExport (Page 2303)	Export user archive
uaArchiveGetCount (Page 2314)	Read the number of data records
uaArchiveGetFieldLength (Page 2323)	Read the current field length
uaArchiveGetFieldName (Page 2324)	Read the current field name
uaArchiveGetFields (Page 2325)	Determine number of fields
uaArchiveGetFieldType (Page 2326)	Read the current field type
uaArchiveGetFieldValueDate (Page 2327)	Read date and time from current data field
uaArchiveGetFieldValueDouble (Page 2328)	Read double value from current data field
uaArchiveGetFieldValueLong (Page 2330)	Read long value from current data field

uaArchiveGetFieldValueString (Page 2331)	Read string from current data field
uaArchiveGetFilter (Page 2316)	Read filter setting
uaArchiveGetID (Page 2304)	Read ID of the current data field
uaArchiveGetName (Page 2305)	Read name of archive
uaArchiveGetSort (Page 2307)	Read sorting of the current data field
uaArchiveImport (Page 2308)	Import user archive
uaArchiveInsert (Page 2317)	Insert data record into user archive
uaArchiveMoveFirst (Page 2318)	Move to first data record
uaArchiveMoveLast (Page 2319)	Move to last data record
uaArchiveMoveNext (Page 2320)	Move to next data record
uaArchiveMovePrevious (Page 2321)	Move to previous data record
uaArchiveOpen (Page 2309)	Create connection to current user archive
uaArchiveReadTagValuesByName (Page 2344)	Read tag values based on the name
uaArchiveReadTagValues (Page 2343)	Read tag values
uaArchiveRequery (Page 2338)	Reload to current user archive
uaArchiveSetFieldValueDate (Page 2332)	Write date and time to current data field
uaArchiveSetFieldValueDouble (Page 2333)	Write double value to current data field
uaArchiveSetFieldValueLong (Page 2334)	Write long value to current data field
uaArchiveSetFieldValueString (Page 2336)	Write string to current data field
uaArchiveSetFilter (Page 2339)	Set filter
uaArchiveSetSort (Page 2341)	Set up sorting criterion
uaArchiveUpdate (Page 2310)	Update data record into user archive
uaArchiveWriteTagValuesByName (Page 2347)	Write data record to tag based on the name
uaArchiveWriteTagValues (Page 2345)	Write data record to tag
uaConnect (Page 2290)	Establish connection to a user archive (runtime).
uaDisconnect (Page 2292)	Close connection to user archive
uaGetArchive (Page 2311)	Read archive configuration
uaGetField (Page 2337)	Read field configuration
uaGetLastError (Page 2286)	Query last error in user archive
AUTOHOTSPOT	Query last COM function error that occurred
uaGetNumArchives (Page 2313)	Find number of configured user archives
uaGetNumFields (Page 2322)	Determine number of fields
uaIsActive (Page 2288)	Query whether an archive is open in runtime
uaOpenArchives (Page 2313)	Query how many user archives are open in runtime
uaOpenViews (Page 2342)	Query how many user archive views are open in runtime
uaQueryArchiveByName (Page 2295)	Set up connection to user archive based on the name (runtime)
uaQueryArchive (Page 2293)	Set up connection to user archive (runtime)
UaQueryConfiguration (Page 2297)	Query connection to the configuration
uaReleaseArchive (Page 2298)	Close connection to user archive (runtime)
uaReleaseConfiguration (Page 2299)	Close connection to the configuration

See also

uaUsers (Page 2289)

3.9.1.2 Structure of recipes (RT Professional)

Introduction

dwKeyField

0	1	2			n
Record-number	Field 1	Field 2			Field n
1					
2					
3					
m					

hRecord

User archives are individual tables in the database that can be structured by the user in any manner, except for the first column. The data types are limited to char, double, int, and DateTime fields.

The GetField and SetField functions act on an internal hRecord data record that can be written to the archive with Insert and Update.

If the user archive is modified directly via the database (ODBC/SQL), an online synchronization is not performed.

Configuration of User Archives

The first step is to configure the user archives. This configuration can be carried out with the User Archives editor, the User Archives script functions, or the User Archives API functions. The script functions are identical to the API functions in this case.

Configuration with the User Archives API functions

The UAQueryConfiguration function supplies a handle (UAHCONFIG) for the configuration functions. This handle can be used to call the configuration functions UASetArchive, UAAAddArchive, UASetField, UAAAddField, etc. The "UAReleaseConfiguration" function completes the configuration of User Archives.

These functions are not available in multiclient projects.

Establishing connection to User Archives

Next, standard function UAConnect must be called in order to establish a connection to the User Archives component. UAConnect creates the UAHCONNECT handle, which enables archives (but not views) to be opened and closed.

Information functions in runtime mode

The UAIsActive function indicates whether an archive is active and open in runtime. UAUsers returns the number of logged on users, UAOpenArchives the number of open archives, and UAOpenViews the number of open views.

Opening of runtime functions

A configured user archive is required for runtime mode. The UAQueryArchive and UAQueryArchiveByName provide a handle for the runtime functions. Once the archive is opened with the UAArchiveOpen function, the User Archives runtime functions can be used.

Functions for runtime mode

The UAArchiveMoveNext, UAArchiveMovePrevious, UAArchiveMoveFirst, and UAArchiveMoveLast functions are used to move the cursor. The hArchive handle ensures a unique assignment to a data record in the user archive. This assignment also allows indirect addressing, as is required for screen masks, for example.

The UAArchiveUpdate function saves the temporary data record in the archive and in doing so overwrites the data record on which the cursor is currently positioned. This data record must be read beforehand using the UAArchiveMoveNext, UAArchiveMovePrevious, UAArchiveMoveFirst, or UAArchiveMoveLast function.

Closing connection to User Archives

The UAArchiveClose function is used to close a user archive. The UAReleaseArchive function closes the connection to the current archive and the UADisconnect function then terminates the connection to the User Archives component.

3.9.1.3 Dependencies in call sequence of API functions (RT Professional)

Dependencies in call sequence of runtime functions

The User Archives function `UAConnect` creates the `UAHCONNECT` handle. This handle is required for opening and closing archives (not views). Therefore, the `UAConnect` function must be called first in order to obtain the `UAHCONNECT` handle. This handle can then be used to call the API functions for opening and closing archives (not views), as listed below. The `UADisconnect` function must be called to complete the configuration.

The `UAQueryArchive` and `UAQueryArchiveByName` functions create the `UAHARCHIVE` handle. This handle is a requirement for the User Archives API function `UAArchiveOpen`, which opens the archive for runtime mode.

Handles for runtime archive functions

`UAConnect` Handle `UAHCONNECT`

--->

Required by `UAlsActive`

`UAUsers`

`UAArchiveOpen`

`UAArchiveViews`

`UAQueryArchive` ---> Handle `UAHARCHIVE`

`UAQueryArchiveByName` ---> Handle `UAHARCHIVE`

Required by:

`UAArchiveOpen` --->

Required for

`UAArchiveDelete`

`UAArchiveExport`

`UAArchiveGetCount`

`UAArchiveGetFieldLength`

`UAArchiveGetFields`

`UAArchiveGetFieldType`

`UAArchiveGetFieldValueDate`

`UAArchiveGetFieldValueDouble`

`UAArchiveGetFieldValueFloat`

`UAArchiveGetFieldValueLong`

`UAArchiveGetFieldValueString`

`UAArchiveGetFieldName`

`UAArchiveGetFilter`

`UAArchiveGetID`

`UAArchiveGetName`

`UAArchiveGetSort`


```

    UAArchiveImport
    UAArchiveInsert
    UAArchiveMoveFirst
    UAArchiveMoveLast
    UAArchiveMoveNext
    UAArchiveMovePrevious
    UAArchiveReadTagValues
    UAArchiveReadTagValuesBy-
    Name
    UAArchiveRequery
    UAArchiveSetFieldValueDate
    UAArchiveSetFieldValueDouble
    UAArchiveSetFieldValueFloat
    UAArchiveSetFieldValueLong
    UAArchiveSetFieldValueString
    UAArchiveSetFilter
    UAArchiveSetSort
    UAArchiveUpdate
    UAArchiveWriteTagValues
    UAArchiveWriteTagValuesByName
    <--- UAArchiveClose

    <---
    UAReleaseArchive

    <---    UADisconnect

```

3.9.1.4 Error messages (RT Professional)

Overview

The following error messages can be queried with API function `UAGetLastError`.

<code>UA_ERROR_SUCCESS</code>	0	No error, everything is OK.
<code>UA_ERROR_GENERIC</code>	1	Error in configuration function.
<code>UA_ERROR_CONNECT_FAILED</code>	100	No runtime connection established (not runtime).
<code>UA_ERROR_OPEN_FAILED</code>	101	The archive cannot be opened.
<code>UA_ERROR_CLOSE_FAILED</code>	102	The archive cannot be closed, or the handle is invalid.
<code>UA_ERROR_REQUERY_FAILED</code>	103	The reloading to archive with, for example, a new filter/new sorting could not be performed.
<code>UA_ERROR_MOVE_FAILED</code>	104	Movement to the first, last, next, or previous data record could not be performed.
<code>UA_ERROR_INSERT_FAILED</code>	105	A data record could not be inserted at the specified position (check position).

3.9 Recipe functions (RT Professional)

UA_ERROR_UPDATE_FAILED	106	The specified data record could not be updated (locked, or no data record exists at this position).
UA_ERROR_DELETE_FAILED	107	The data record(s) could not be deleted.
UA_ERROR_IMPORT_FAILED	108	The import of an archive failed (check format, size).
UA_ERROR_EXPORT_FAILED	109	The export of the user archive was not successful (check space).
UA_ERROR_READ_FAILED	110	Tags could not be read.
UA_ERROR_WRITE_FAILED	111	Tags could not be written
UA_ERROR_GET_FAILED	112	Archive/field description and/or data record values could not be read.
UA_ERROR_SET_FAILED	113	Archive/field description and/or data record values could not be written.
UA_ERROR_INVALID_NAME	200	Invalid name.
UA_ERROR_INVALID_TYPE	201	Incorrect type specified
UA_ERROR_INVALID_NUMRECS	202	Number of data records cannot be received
UA_ERROR_INVALID_COMMTYPE	203	Incorrect connection (None / Raw / Direct) selected
UA_ERROR_INVALID_LENGTH	204	The specified (string) length is not received
UA_ERROR_INVALID_PRECISION	205	The specified number of digits to the right of the decimal point will not be received.
UA_ERROR_NULL_POINTER	1000	A NULL pointer was passed
UA_ERROR_INVALID_POINTER	1001	The specified pointer is invalid
UA_ERROR_INVALID_HANDLE	1002	The specified handle is invalid
UA_ERROR_INVALID_INDEX	1003	The specified index is invalid
UA_ERROR_SERVER_UNKNOWN	1004	The specified server in use is invalid.

3.9.1.5 Constants (RT Professional)

Configuration constants:

Archive types

UA_ARCHIVETYPE_UNKNOWN	0	Unknown archive type
UA_ARCHIVETYPE_UNLIMITED	1	Circular archive
UA_ARCHIVETYPE_LIMITED	2	Archive with fixed number of data records

Archive flags

UA_ARCHIVEFLAG_ACCESS	0x00000001	An additional column showing the time of the last access is also created
UA_ARCHIVEFLAG_USER	0x00000002	An additional column showing the user name is also created

Communication types

UA_COMMTYPE_NONE	1	No tag connection
UA_COMMTYPE_RAW	2	Communication via raw data tag
UA_COMMTYPE_DIRECT	3	Communication via DM tag

Field types

UA_FIELDTYPE_INTEGER	1	Field type INT
UA_FIELDTYPE_DOUBLE	2	Field type double
UA_FIELDTYPE_FLOAT	3	Field type float
UA_FIELDTYPE_STRING	6	Field type string/byte array null-terminated
UA_FIELDTYPE_DATETIME	8	Field type for date and time

Field flags

UA_FIELDFLAG_UNIQUE	0x00000001	Field must contain a unique value
UA_FIELDFLAG_NOTNULL	0x00000002	Field must contain a value
UA_FIELDFLAG_INDEX	0x00000010	Field should be supported by index

Maximum length

UA_MAXLEN_NAME	20	Maximum length of the archive/field name
UA_MAXLEN_ALIAS	50	Maximum length of the second supplemental name/comment
UA_MAXLEN_PLCID	8	Maximum length of the communication device name
UA_MAXLEN_DMVARNAME	128	Maximum length of the data manager tag name
UA_MAXLEN_VALUE	255	Maximum length for string tag

Parameter constants:

Actions

UA_ACTION_GENERIC	0	Configuration
UA_ACTION_INSERT	1	Insert
UA_ACTION_UPDATE	2	Synchronize/overwrite
UA_ACTION_DELETE	3	Delete

Move values

UA_MOVE_FIRST	0	Move to first data record
UA_MOVE_NEXT	1	Move to next data record
UA_MOVE_PREVIOUS	-1	Move to previous data record
UA_MOVE_LAST	0x80000000	Move to last data record

Import/export file types

UA_FILETYPE_DEFAULT	0	CSV is the default file type
UA_FILETYPE_CSV	1	(comma separated value) file type

3.9.2 General functions (RT Professional)**3.9.2.1 uaGetLastError (RT Professional)****Description**

Many functions return a Boolean value. TRUE indicates error-free function processing. If the value FALSE is returned, the error can be read in the last function used by using the uaGetLastError function.

uaGetLastError always takes you back to the last error. If you know precisely which function contains an error, you must evaluate the return after each call of a function and call uaGetLastError in the event of an error.

Example of functions with return values

```
if ( uaArchiveGetFieldValueLong ( hArchive, Index, &IntValue ) == TRUE )
    printf( "Field Value = %u\n", IntValue );
else
    printf("Error calling uaArchiveGetFieldValueLong: %d / %08lx\n", uaGetLastError(),
    uaGetLastHResult());
```

Example of functions with no return values

```
uaArchiveGetFilter(hArchive, pszFilter, cMaxLen);
    INT nUAError = uaGetLastError ( );
    if ( UA_ERROR_SUCCESS != nUAError)
    {
        printf( "Filter = [%s]\n", pszFilter );
    }
    else
    {
        printf("Error calling uaArchiveGetFilter: %d, hr=0x%08lx\n", nUAError,
        uaGetLastHResult());
    }
    INT uaGetLastError()
```

Return value

Error status of the last function executed. Error constants and predefined items are determined in CCUACAPI.H.

UA_ERROR_SUCCESS	No error, everything is OK.
UA_ERROR_GENERIC	Error in configuration function.
UA_ERROR_CONNECT_FAILED	No Runtime connection established.
UA_ERROR_OPEN_FAILED	The archive cannot be opened.
UA_ERROR_CLOSE_FAILED	The archive cannot be closed, or the handle is invalid.
UA_ERROR_REQUERY_FAILED	The archive reset relationship with for example, a new filter/new sorting could not be established.
UA_ERROR_MOVE_FAILED	The new First, Last, Next or Previous positioning could not be performed.
UA_ERROR_INSERT_FAILED	The record cannot be inserted at the given position. Check the position.
UA_ERROR_UPDATE_FAILED	The record stated cannot be updated. The record is locked or no record is available in this position.
UA_ERROR_DELETE_FAILED	The record(s) cannot (all) be deleted.
UA_ERROR_IMPORT_FAILED	The import of an archive was not successful. Check the archive format and size.
UA_ERROR_EXPORT_FAILED	The user archive export was not successful. Check the size of the memory cell.
UA_ERROR_READ_FAILED	Tag reading was not performed.
UA_ERROR_WRITE_FAILED	Writing to tags was not performed.
UA_ERROR_GET_FAILED	Reading the archive / field description and/or record values was not performed.
UA_ERROR_SET_FAILED	Archive / field description and/or record value writing was not performed.
UA_ERROR_INVALID_NAME	Invalid name.
UA_ERROR_INVALID_TYPE	Invalid type.
UA_ERROR_INVALID_NUMRECS	Invalid record number.
UA_ERROR_INVALID_COMMTYPE	Wrong link to None / Raw / Direct selected.
UA_ERROR_INVALID_LENGTH	The specified (String) length is invalid.
UA_ERROR_INVALID_PRECISION	The specified number of decimal points is invalid.
UA_ERROR_NULL_POINTER	A zero pointer was specified.
UA_ERROR_INVALID_POINTER	The specified pointer is invalid.
UA_ERROR_INVALID_HANDLE	The specified handle is invalid.
UA_ERROR_INVALID_INDEX	The specified index is invalid.
UA_ERROR_SERVER_UNKNOWN	The server specified as used is unknown.

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

3.9.2.2 **uaIsActive** (RT Professional)

Description

Check that no recipe is in use in an active Runtime.

Declaration

```
BOOL uaIsActive (  
    UAHCONNECT hConnect )
```

Parameters

hConnect

Handle on recipe in Runtime. This handle is generated using uaConnect.

Return value

TRUE

Recipe in use in active Runtime.

FALSE

Recipe not in use in active Runtime.

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaConnect (Page 2290)	Generate handle

See also

uaConnect (Page 2290)

3.9.2.3 uaUsers (RT Professional)

Description

Provides the number for all users, connected to a recipe by uaConnect. Here, consider calls by users, e.g. through scripts that are also contained within WinCC.

Declaration

```
LONG uaUsers (  
    UAHCONNECT hConnect )
```

Parameters

hConnect

Handle on recipe in Runtime. This handle is generated using uaConnect.

Return value

Number of active connections

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaConnect (Page 2290)	Generate handle

See also

uaConnect (Page 2290)

3.9.2.4 uaSetLocalEvents (RT Professional)

Description

Sets local events.

Declaration

```
void uaSetLocalEvents (
    UAHCONNECT hConnect
    BOOL bLocalEvents )
```

Parameters

hConnect

Handle on recipe in Runtime. This handle is generated using uaConnect.

bLocalEvents

Local Event

Return value

No return value

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaConnect (Page 2290)	Generate handle

See also

uaConnect (Page 2290)

3.9.3 Functions for generating connections (RT Professional)

3.9.3.1 uaConnect (RT Professional)

Description

Generates a connection to a recipe in Runtime.

Declaration

```
BOOL uaConnect (
    UAHCONNECT* phConnect )
```

Parameters

phConnect

Pointer to the recipe handle.

Return value

TRUE

Connection to recipe generated

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Comments

The supplied handle is required for the functions uaDisconnect, ualsActive, uaUsers, uaOpenViews, uaOpenArchives, uaQueryArchive and uaQueryArchiveByName.

Related functions

uaDisconnect (Page 2292)	Terminate connection

See also

ualsActive (Page 2286)
uaUsers (Page 2287)
uaSetLocalEvents (Page 2287)
uaDisconnect (Page 2292)
uaQueryArchive (Page 2293)
uaQueryArchiveByName (Page 2295)

[uaOpenArchives \(Page 2313\)](#)

[uaOpenViews \(Page 2342\)](#)

[Overview of the functions \(Page 2276\)](#)

3.9.3.2 uaDisconnect (RT Professional)

Description

When a connection is made to a recipe in Runtime, the connection is terminated.

Declaration

```
BOOL uaDisconnect (  
    UAHCONNECT hConnect )
```

Parameters

hConnect

Handle on recipe in Runtime. This handle is generated using uaConnect.

Return value

TRUE

Connection terminated.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaConnect (Page 2288)	Generate handle

See also

uaConnect (Page 2288)
Overview of the functions (Page 2276)

3.9.3.3 uaQueryArchive (RT Professional)

Description

Makes a connection to a recipe. uaQueryArchive generates the handle phArchive.

Declaration

```
BOOL uaQueryArchive (
    UAHCONNECT      hConnect,
    LONG            lArchive,
    UAHARCHIVE*     phArchive )
```

Parameters

hConnect

Handle on recipe in Runtime. This handle is generated using uaConnect.

lArchive

ID of the archive to be connected.

phArchive

Pointer to the connected recipe handle.

Return value

TRUE

Connection established

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Comments

If only an archive index is available, the ID of the archive can be determined with the configuration function `uaGetArchive`.

Related functions

uaConnect (Page 2288)	Generate handle

See also

[uaConnect \(Page 2288\)](#)
[uaReleaseArchive \(Page 2298\)](#)
[uaArchiveClose \(Page 2301\)](#)
[uaArchiveDelete \(Page 2302\)](#)
[uaArchiveExport \(Page 2303\)](#)
[uaArchiveGetID \(Page 2304\)](#)
[uaArchiveGetName \(Page 2305\)](#)
[uaArchiveGetSort \(Page 2307\)](#)
[uaArchiveImport \(Page 2308\)](#)
[uaArchiveOpen \(Page 2309\)](#)
[uaArchiveUpdate \(Page 2310\)](#)
[uaArchiveGetCount \(Page 2314\)](#)
[uaArchiveGetFilter \(Page 2316\)](#)
[uaArchiveInsert \(Page 2317\)](#)
[uaArchiveMoveFirst \(Page 2318\)](#)
[uaArchiveMoveLast \(Page 2319\)](#)
[uaArchiveMoveNext \(Page 2320\)](#)
[uaArchiveMovePrevious \(Page 2321\)](#)
[uaArchiveGetFieldLength \(Page 2323\)](#)
[uaArchiveGetFields \(Page 2325\)](#)
[uaArchiveGetFieldType \(Page 2326\)](#)
[uaArchiveGetFieldName \(Page 2324\)](#)
[uaArchiveGetFieldValueDate \(Page 2327\)](#)
[uaArchiveGetFieldValueDouble \(Page 2328\)](#)
[uaArchiveGetFieldValueLong \(Page 2330\)](#)
[uaArchiveSetFieldValueDate \(Page 2332\)](#)
[uaArchiveSetFieldValueDouble \(Page 2333\)](#)

uaArchiveSetFieldValueLong (Page 2334)
uaArchiveSetFieldValueString (Page 2336)
uaArchiveRequery (Page 2338)
uaArchiveSetFilter (Page 2339)
uaArchiveSetSort (Page 2341)
uaArchiveReadTagValues (Page 2343)
uaArchiveReadTagValuesByName (Page 2344)
uaArchiveWriteTagValues (Page 2345)
uaArchiveWriteTagValuesByName (Page 2347)
Overview of the functions (Page 2276)

3.9.3.4 uaQueryArchiveByName (RT Professional)

Description

Makes a connection to a recipe by name. uaQueryArchiveByName generates the handle phArchive.

Declaration

```
BOOL uaQueryArchiveByName (
    UAHCONNECT      hConnect,
    LPCSTR           pszName,
    UAHARCHIVE*      phArchive )
```

Parameters

hConnect

Handle on recipe in Runtime. This handle is generated using uaConnect.

pszName

Recipe name.

phArchive

Pointer to the connected recipe handle.

Return value

TRUE

Connection established

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Comments

If only an archive index is available, the name of the archive can be determined with the configuration function `uaGetArchive`.

Related functions

uaConnect (Page 2288)	Generate handle

See also[uaConnect \(Page 2288\)](#)[uaReleaseArchive \(Page 2298\)](#)[uaArchiveClose \(Page 2301\)](#)[uaArchiveDelete \(Page 2302\)](#)[uaArchiveExport \(Page 2303\)](#)[uaArchiveGetID \(Page 2304\)](#)[uaArchiveGetName \(Page 2305\)](#)[uaArchiveGetSort \(Page 2307\)](#)[uaArchiveImport \(Page 2308\)](#)[uaArchiveOpen \(Page 2309\)](#)[uaArchiveUpdate \(Page 2310\)](#)[uaArchiveGetCount \(Page 2314\)](#)[uaArchiveGetFilter \(Page 2316\)](#)[uaArchiveInsert \(Page 2317\)](#)[uaArchiveMoveFirst \(Page 2318\)](#)[uaArchiveMoveLast \(Page 2319\)](#)[uaArchiveMoveNext \(Page 2320\)](#)[uaArchiveMovePrevious \(Page 2321\)](#)[uaArchiveGetFieldLength \(Page 2323\)](#)

uaArchiveGetFields (Page 2325)
uaArchiveGetFieldType (Page 2326)
uaArchiveGetFieldName (Page 2324)
uaArchiveGetFieldValueDate (Page 2327)
uaArchiveGetFieldValueDouble (Page 2328)
uaArchiveGetFieldValueLong (Page 2330)
uaArchiveSetFieldValueDate (Page 2332)
uaArchiveSetFieldValueDouble (Page 2333)
uaArchiveSetFieldValueLong (Page 2334)
uaArchiveSetFieldValueString (Page 2336)
uaArchiveRequery (Page 2338)
uaArchiveSetFilter (Page 2339)
uaArchiveSetSort (Page 2341)
uaArchiveReadTagValues (Page 2343)
uaArchiveReadTagValuesByName (Page 2344)
uaArchiveWriteTagValues (Page 2345)
uaArchiveWriteTagValuesByName (Page 2347)
Overview of the functions (Page 2276)

3.9.3.5 UaQueryConfiguration (RT Professional)

Description

Generates the connection to a recipe for the configuration.

Declaration

```
BOOL uaQueryConfiguration (  
    UAHCONFIG*      phConfig )
```

Parameters

phConfig

Pointer to receptor handle.

Return value

TRUE

Connection established

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

See also

[uaReleaseConfiguration](#) (Page 2299)

[uaGetArchive](#) (Page 2311)

[uaGetNumArchives](#) (Page 2313)

[uaGetNumFields](#) (Page 2322)

[uaGetField](#) (Page 2337)

[Overview of the functions](#) (Page 2276)

3.9.3.6 **uaReleaseArchive** (RT Professional)

Description

Breaks off the connection to the currently connected recipe.

Declaration

```
BOOL uaReleaseArchive (  
    UAHARCHIVE      hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using [uaQueryArchive](#) or [uaQueryArchiveByName](#).

Return value

TRUE

Connection terminated.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Comment

To successfully terminate the connection you must set the hArchive handle to NULL . If you use a handle that is no longer valid the error message UA_ERROR_INVALID_HANDLE is generated. This avoids unnecessary memory overloading.

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.3.7 uaReleaseConfiguration (RT Professional)

Description

Terminates a connection to a recipe after configuring it.

Declaration

```
BOOL uaReleaseConfiguration (  
    UAHCONFIG    hConfig,  
    BOOL         bSave )
```

Parameters

hConfig

Handle on recipe. This handle is generated using uaQueryConfiguration.

bSave

Establishes that changes to a configuration are saved before terminating the connection.

TRUE	Save changes
FALSE	Cancel changes

Note

bSave = TRUE should only be used, if a recipe is not used in an active WinCC Runtime! Check if Runtime is active by using the ualsActive function.

Return value

TRUE

Connection terminated.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

UaQueryConfiguration (Page 2295)	Generate handle

See also

UaQueryConfiguration (Page 2295)

Overview of the functions (Page 2276)

3.9.4 Functions for recipe processing (RT Professional)

3.9.4.1 uaArchiveClose (RT Professional)

Description

Closes an open recipe.

Declaration

```
BOOL uaArchiveClose (
    UAHARCHIVE      hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Recipe closed.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle
uaArchiveOpen (Page 2309)	Open recipe

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
uaArchiveOpen (Page 2309)
Overview of the functions (Page 2276)

3.9.4.2 uaArchiveDelete (RT Professional)

Description

Deletes data from a recipe. The configured recipe remains selected.

Declaration

```
BOOL uaArchiveDelete (
    UAHARCHIVE    hArchive,
    LPCSTR         pszWhere )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszWhere

String with the SQL selection for the data entries to be deleted. This string corresponds to the SQL instruction DELETE FROM <archive> WHERE pszWhere.

Note

If pszWhere is empty, the entire user archive will be deleted.

Return value

TRUE

Data deleted.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.4.3 uaArchiveExport (RT Professional)

Description

Exports data from a recipe to CSV file.

Declaration

```
BOOL uaArchiveExport (
    UAHARCHIVE    hArchive,
    LPCSTR        pszDestination,
    LONG          lType,
    LONG          lOptions )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszDestination

Target file name. When calling the function on a client, the path specified refers to the server.

IType

Target file data format. Two formats are available:

UA_FILETYPE_DEFAULT = 0	The preset CSV file format
UA_FILETYPE_CSV = 1	CSV file format

IOptions

The parameter is reserved for future development and must be preset to 0.

Return value**TRUE**

Data exported.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle
uaArchiveImport (Page 2308)	Import data

See also

uaArchiveImport (Page 2308)

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.4.4 uaArchiveGetID (RT Professional)**Description**

Reads the recipe ID

Return Value:

"ID" of the user archive

Declaration

```
LONG uaArchiveGetID (
    UAHARCHIVE      hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

ID of the recipe

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.4.5 uaArchiveGetName (RT Professional)

Description

Reads the recipe name.

Declaration

```
VOID uaArchiveGetName (
    UAHARCHIVE      hArchive,
    LPSTR            pszName,
    LONG             cMaxLen )
```

Parameters

- hArchive**
Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.
- pszName**
Pointer to the recipe name buffer.
- cMaxLen**
Maximum Length

Return value

No return value

Required files

- ccuacapi.h
- ccuacapi.lib
- ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

Example

```
char tank [40];
uaArchiveGetName( hArchive, tank, 39 );
```


See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.4.6 uaArchiveGetSort (RT Professional)

Description

Reads the recipe sort parameters.

Declaration

```
VOID uaArchiveGetSort (
    UAHARCHIVE      hArchive,
    LPSTR            pszSort,
    LONG             cMaxLen )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszSort

Sort parameters as SQL statement.

cMaxLen

Maximum Length

Return value

No return value

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.4.7 uaArchiveImport (RT Professional)**Description**

Imports data from a CSV file to a recipe. The structure of the recipe must be identical to the imported CSV archive.

Declaration

```

BOOL uaArchiveImport (
    UAHARCHIVE      hArchive,
    LPCSTR          pszSource,
    LONG             lType,
    LONG             lOptions )

```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszSource

File name using the data to be imported.

lType

Source file data format. Two formats are available:

UA_FILETYPE_DEFAULT = 0	The preset CSV file format
UA_FILETYPE_CSV = 1	CSV file format

lOptions

The parameter is reserved for future development and must be preset to 0.

Return value

TRUE

Data imported.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle
uaArchiveExport (Page 2301)	Export files.

See also

uaArchiveExport (Page 2301)

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.4.8 uaArchiveOpen (RT Professional)

Description

Opens a recipe. Calling uaArchiveOpen is required to read or write in a recipe. For example calling the functions uaArchiveMoveNext, uaArchiveDelete or uaArchiveUpdate.

Declaration

```
BOOL uaArchiveOpen (
    UAHARCHIVE hArchive )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value**TRUE**

Recipe open.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle
uaArchiveClose (Page 2299)	Close recipe

See also

uaArchiveClose (Page 2299)

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.4.9 uaArchiveUpdate (RT Professional)**Description**

Updates an open recipe. All changed recipe data will be applied in the database. The recipe structure remains unchanged.

Declaration

```
BOOL uaArchiveUpdate (
U    AHARCHIVE hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Recipe data updated.

FALSE

In the case of a consistency violation, the error message Update_failed = 106 is displayed. This kind of consistency violation exists, for example, if a field has the "Field must contain a value" property, but there is no value in the field.

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.4.10 uaGetArchive (RT Professional)

Description

Reads the recipe configuration.

Declaration

```
BOOL uaGetArchive (
    UAHCONFIG          hConfig,
    Long               lArchive,
    UACONFIGARCHIVE*   pArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryConfiguration.

lArchive

Recipe index. Value: 0... uaGetNumArchives() -1

pArchive

Point on the configuration data buffer.

Return value

TRUE

Successful access.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

UaQueryConfiguration (Page 2295)	Generate handle

See also

UaQueryConfiguration (Page 2295)

Overview of the functions (Page 2276)

3.9.4.11 uaGetNumArchives (RT Professional)

Description

Reads the number of recipes currently configured.

Declaration

```
LONG uaGetNumArchives (  
    UAHCONFIG      hConfig )
```

Parameters

hConfig

Handle on recipe. This handle is generated using uaQueryConfiguration.

Return value

Number of recipes configured. If an error occurs -1 is output.

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

UaQueryConfiguration (Page 2295)	Generate handle

See also

UaQueryConfiguration (Page 2295)
Overview of the functions (Page 2276)

3.9.4.12 uaOpenArchives (RT Professional)

Description

Communicates the number of recipes in the open Runtime.

Declaration

```
LONG uaOpenArchives (
    UAHCONNECT hConnect )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaConnect.

Return value

Number of recipes open.

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaConnect (Page 2288)	Generate handle

See also

uaConnect (Page 2288)
Overview of the functions (Page 2276)

3.9.5 Functions for recipe element processing (RT Professional)

3.9.5.1 uaArchiveGetCount (RT Professional)

Description

Reads the number of data records.

Return Value:

Number of data records. If "0" the archive is empty or an error has occurred. A query using "uaGetLastError" is required.

Declaration

```
LONG uaArchiveGetCount(  
    UAHARCHIVE      hArchive,  
    LONG*           plCount )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

plCount

Pointer to a tag in which the number of records should be stored.

Return value

Number of data records. If 0 the archive is empty or an error has occurred. A query using uaGetLastError is recommended.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.5.2 uaArchiveGetFilter (RT Professional)

Description

Reads the selection criteria for the current data record.

Declaration

```
VOID uaArchiveGetFilter (
    UAHARCHIVE      hArchive,
    LPSTR            pszFilter,
    LONG             cMaxLen )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszFilter

Selection parameters as SQL statement.

cMaxLen

Maximum Length

Return value

No return value

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.5.3 uaArchiveInsert (RT Professional)

Description

Inserts the local data record buffer into the recipe. In order for meaningful records to be stated in the new data file, before calling uaArchiveInsert the fields of the local data record buffers must be written using the uaArchiveSetFieldValue...function.

The "ID" field must be written using uaArchiveSetFieldValueLong function if an archive has no ID or if it is written as "0".

Declaration

```
BOOL uaArchiveInsert (
    UAHARCHIVE    hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Data record inserted.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.5.4 uaArchiveMoveFirst (RT Professional)**Description**

Goes to the first data record.

Declaration

```
BOOL uaArchiveMoveFirst (
    UAHARCHIVE      hArchive )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value**TRUE**

Successful jump.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.5.5 uaArchiveMoveLast (RT Professional)

Description

Goes to the last data record.

Declaration

```
BOOL uaArchiveMoveLast (
    UAHARCHIVE      hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful jump.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.5.6 uaArchiveMoveNext (RT Professional)**Description**

Goes to the next data record.

Declaration

```
BOOL uaArchiveMoveNext (  
    UAHARCHIVE      hArchive )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value**TRUE**

Successful jump.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.5.7 uaArchiveMovePrevious (RT Professional)

Description

Goes to the previous data record.

Declaration

```
BOOL uaArchiveMovePrevious (  
    UAHARCHIVE      hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Successful jump.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.5.8 uaGetNumFields (RT Professional)**Description**

Supplies the number of the configured fields. The "ID", "Last User" and "Last Access" fields are not included. In the configuration calls the index is stated using "0 ... uaGetNumFields() -1".

Declaration

```
LONG uaGetNumFields (
    UAHCONFIG      honfig,
    long           lArchive )
```

Parameters**hConfig**

Handle on recipe. This handle is generated using uaQueryConfiguration.

lArchive

Recipe index. Value: 0... uaGetNumArchives() -1

Return value

Number of the configured fields. If an error occurs -1 is output.

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

UaQueryConfiguration (Page 2295)	Generate handle

See also

UaQueryConfiguration (Page 2295)
Overview of the functions (Page 2276)

3.9.6 Functions for field processing (RT Professional)

3.9.6.1 uaArchiveGetFieldLength (RT Professional)

Description

Reads the length of a field in the current data record.

Declaration

```
LONG uaArchiveGetFieldLength(  
    UAHARCHIVE    hArchive,  
    LONG          lField )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

Return value

Length of the current field.

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.6.2 uaArchiveGetFieldName (RT Professional)**Description**

Reads the name of a field in the current data record.

Declaration

```
VOID uaArchiveGetFieldName (
    UAHARCHIVE    hArchive,
    LONG           lField,
    LPCSTR         pszName,
    LONG           cMaxLen )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

pszName

Field Name

cMaxLen

Maximum Length

Return value

TRUE

Name read.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

AUTOHOTSPOT	Generate handle
uaQueryArchive (Page 2291)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.6.3 uaArchiveGetFields (RT Professional)

Description

Reads the number of the configured data fields. The "ID", "Last User" and "Last Access" fields are included. In Runtime calls the index of configured fields is indicated using 1 ... N. The "ID" field has a "0" index. The "Last User" and "Last Access" fields are appended at the end of the configured fields.

Declaration

```
LONG uaArchiveGetFields (  
    UAHARCHIVE      hArchive )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

Number of the configured fields.

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.6.4 uaArchiveGetFieldType (RT Professional)**Description**

Reads the type of a field in the current data record.

Declaration

```
LONG uaArchiveGetFieldType (  
    UAHARCHIVE    hArchive,  
    LONG           lField )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

IField

The field number, where iField = 1 addresses the first configured field. IField = 0 addresses the "ID" field.

Return value

Type of the current field. The symbolic definitions of the field types are:

UA_FIELDTYPE_INTEGER	
UA_FIELDTYPE_FLOAT	
UA_FIELDTYPE_DOUBLE	
UA_FIELDTYPE_STRING	
UA_FIELDTYPE_DATETIME	

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.6.5 uaArchiveGetFieldValueDate (RT Professional)

Description

Reads the date and time of a field in the current data record.

Declaration

```
BOOL uaArchiveGetFieldValueDate (
    UAHARCHIVE      hArchive,
    LONG             lField,
    LPSYSTEMTIME     pstDateTime )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

IField

The field number, where iField = 1 addresses the first configured field. IField = 0 addresses the "ID" field.

pstDateTime

Pointer to a tag of the SYSTEMTIME type.

Return value

TRUE

Change date and time.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.6.6 uaArchiveGetFieldValueDouble (RT Professional)

Description

Reads a value of the Double data type from a field in the current data record.

Declaration

```
BOOL uaArchiveGetFieldValueDouble (
    UAHARCHIVE      hArchive,
    LONG             lField,
    double*          pdValue )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

pdValue

Pointer to the tag for the field value.

Return value

TRUE

Value read.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.6.7 uaArchiveGetFieldValueLong (RT Professional)

Description

Reads a value of the Long data type from a field in the current data record.

Declaration

```
BOOL uaArchiveGetFieldValueLong (
    UAHARCHIVE      hArchive,
    LONG             lField,
    LONG*            pdValue )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

pdValue

Pointer to the tag for the field value.

Return value

TRUE

Value read.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.6.8 uaArchiveGetFieldValueString (RT Professional)

Description

Reads a value of the String data type from a field in the current data record.

Declaration

```
BOOL uaArchiveGetFieldValueString (
    UAHARCHIVE      hArchive,
    LONG             lField,
    LPSTR            pszString,
    LONG             cMaxLen )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

pszString

Pointer to the tag for the field value.

pdValue

Maximum length of the string.

Return value

TRUE

Value read.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.6.9 uaArchiveSetFieldValueDate (RT Professional)**Description**

Writes the date and time into a field of the current data record.

Declaration

```
BOOL uaArchiveSetFieldValueDate (
    UAHARCHIVE      hArchive,
    LONG             lField,
    LPSYSTEMTIME     pstDateTime )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

pstDateTime

Date and Time.

Return value

TRUE

Value written.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.6.10 uaArchiveSetFieldValueDouble (RT Professional)

Description

Writes a value from the Double data type from a field in the current data record.

Declaration

```
BOOL uaArchiveSetFieldValueDouble (
    UAHARCHIVE    hArchive,
    LONG           lField,
    double         dValue )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

IField

The field number, where iField = 1 addresses the first configured field. IField = 0 addresses the "ID" field.

dValue

Value to be written.

Return value**TRUE**

Value written.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

Overview of the functions (Page 2276)

3.9.6.11 uaArchiveSetFieldValueLong (RT Professional)**Description**

Writes a value from the Long data type from a field in the current data record.

Declaration

```
BOOL uaArchiveSetFieldValueLong (
    UAHARCHIVE hArchive,
    LONG lField,
    LONG dValue )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

dValue

Value to be written.

Return value

TRUE

Value written.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.6.12 uaArchiveSetFieldValueString (RT Professional)

Description

Writes a value from the String data type from a field in the current data record.

Declaration

```
BOOL uaArchiveSetFieldValueString (
    UAHARCHIVE    hArchive,
    LONG          lField,
    LPCSTR        pszString )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

pszString

String to be written.

Return value

TRUE

Value written.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
Overview of the functions (Page 2276)

3.9.6.13 uaGetField (RT Professional)

Description

Reads the field configuration.

Declaration

```
BOOL uaGetField (
    UAHCONFIG      hConfig,
    Long           lArchive,
    Long           lField,
    UACONFIGFIELD* pField )
```

Parameters

hConfig

Handle on recipe. This handle is generated using uaQueryConfiguration.

lArchive

Recipe index. Value: 0... uaGetNumArchives() -1

lField

The field number, where lField = 1 addresses the first configured field. lField = 0 addresses the "ID" field.

pField

Point on the configuration data buffer.

Return value

TRUE

Configuration read.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

UaQueryConfiguration (Page 2295)	Generate handle

See also

UaQueryConfiguration (Page 2295)
Overview of the functions (Page 2276)

3.9.7 Filter and sort functions. (RT Professional)

3.9.7.1 uaArchiveRequery (RT Professional)

Description

After calling for uaArchiveSetFilter and uaArchiveSetSort you must reload the recipe using uaArchiveRequery.

Note

You can use the uaArchiveSetSort and uaArchiveSetFilter functions, without opening the recipe with uaArchiveOpen. In this case you must not make a call using uaArchiveRequery.

Declaration

```
BOOL uaArchiveRequery(  
    UAHARCHIVE hArchive )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

Return value

TRUE

Recipe reloaded.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Comment

You can also make a call using `uaArchiveRequery`, if you have inserted Runtime values in the recipe view.

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

[uaQueryArchive \(Page 2291\)](#)

[uaQueryArchiveByName \(Page 2293\)](#)

[Overview of the functions \(Page 2276\)](#)

3.9.7.2 uaArchiveSetFilter (RT Professional)

Description

Sets the selection parameters for the recipe. You can also use the function, without opening the recipe using `uaArchiveOpen`.

If you have opened a user archive using `uaArchiveOpen`, you must reload the recipe after filtering using `uaArchiveRequery`.

Declaration

```
VOID uaArchiveSetFilter (
    UAHARCHIVE    hArchive,
    LPSTR          pszFilter )
```

Parameters

hArchive

Handle on recipe. This handle is generated using `uaQueryArchive` or `uaQueryArchiveByName`.

pszFilter

Selection parameters as SQL parameters.

Return value

TRUE

Selection parameters changed.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

[uaQueryArchive \(Page 2291\)](#)
[uaQueryArchiveByName \(Page 2293\)](#)
[Overview of the functions \(Page 2276\)](#)

3.9.7.3 uaArchiveSetSort (RT Professional)

Description

Sets the sort for a recipe. You can also use the function, without opening the recipe using uaArchiveOpen.

If you have opened a recipe using uaArchiveOpen, you must reload the recipe after sorting using uaArchiveRequery.

Declaration

```
BOOL uaArchiveSetSort (
    UAHARCHIVE    hArchive,
    LPSTR          pszSort )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszSort

Sort parameters as SQL statement.

Return value

TRUE

Sort successful.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

- uaQueryArchive (Page 2291)
- uaQueryArchiveByName (Page 2293)
- Overview of the functions (Page 2276)

3.9.8 Functions for recipe view processing (RT Professional)

3.9.8.1 uaOpenViews (RT Professional)

Description

Gives the number of recipes in the open Runtime view.

Declaration

```
LONG uaOpenViews (
    UAHCONNECT      hConnect )
```

Parameters

hArchive
Handle on recipe. This handle is generated using uaConnect.

Return value

Number of currently open views.

Required files

- ccuacapi.h
- ccuacapi.lib
- ccuacapi.dll

Related functions

uaConnect (Page 2288)	Generate handle

See also

uaConnect (Page 2288)
Overview of the functions (Page 2276)

3.9.9 Functions for variables processing (RT Professional)

3.9.9.1 uaArchiveReadTagValues (RT Professional)

Description

Reads the current value from the field tag.

Declaration

```
BOOL uaArchiveReadTagValues (
    UAHARCHIVE    hArchive,
    LONG*         pnFields,
    LONG          cFields,
    LONG          lOptions )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pnFields

The parameter is reserved for future development and must be preset to 0.

cFields

The parameter is reserved for future development and must be preset to 0.

lOptions

The parameter is reserved for future development and must be preset to 0.

Return value

TRUE

Data read.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaArchiveWriteTagValues (Page 2345)	Writing values
uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)
uaQueryArchiveByName (Page 2293)
uaArchiveWriteTagValues (Page 2345)
Overview of the functions (Page 2276)

3.9.9.2 uaArchiveReadTagValuesByName (RT Professional)**Description**

Reads the current values from the tags.

Declaration

```
BOOL uaArchiveReadTagValuesByName (
    UAHARCHIVE    hArchive,
    LPCSTR        pszFields,
    LONG          lOptions )
```

Parameters**hArchive**

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszFields

The parameter is reserved for future development and must be preset to NULL.

lOptions

The parameter is reserved for future development and must be preset to 0.

Return value

TRUE

Data read.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaArchiveWriteTagValuesByName (Page 2347)	Reading values
uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

uaArchiveWriteTagValuesByName (Page 2347)

Overview of the functions (Page 2276)

3.9.9.3 uaArchiveWriteTagValues (RT Professional)

Description

Writes the values of the current data record to the tag.

Declaration

```
BOOL uaArchiveWriteTagValues (
    UAHARCHIVE      hArchive,
    LONG*            pnFields,
    LONG             cFields,
    LONG             lOptions )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pnFields

The parameter is reserved for future development and must be preset to 0.

cFields

The parameter is reserved for future development and must be preset to 0.

lOptions

The parameter is reserved for future development and must be preset to 0.

Return value

TRUE

Data written.

FALSE

Error

Required files

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

Related functions

uaArchiveReadTagValues (Page 2341)	Reading values
uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

uaArchiveReadTagValues (Page 2341)

Overview of the functions (Page 2276)

3.9.9.4 uaArchiveWriteTagValuesByName (RT Professional)

Description

Writes the values of the current data record to the tag.

Declaration

```
BOOL uaArchiveWriteTagValuesByName (
    UAHARCHIVE    hArchive,
    LPCSTR        pszFields,
    LONG          lOptions )
```

Parameters

hArchive

Handle on recipe. This handle is generated using uaQueryArchive or uaQueryArchiveByName.

pszFields

The parameter is reserved for future development and must be preset to NULL.

lOptions

The parameter is reserved for future development and must be preset to 0.

Return value

TRUE

Data written.

FALSE

Error

Required files

ccuacapi.h
ccuacapi.lib
ccuacapi.dll

Related functions

uaArchiveReadTagValuesByName (Page 2342)	Reading values
uaQueryArchive (Page 2291)	Generate handle
uaQueryArchiveByName (Page 2293)	Generate handle

See also

uaQueryArchive (Page 2291)

uaQueryArchiveByName (Page 2293)

uaArchiveReadTagValuesByName (Page 2342)

Overview of the functions (Page 2276)

3.10 Alarm functions (RT Professional)**3.10.1 Basics (RT Professional)****3.10.1.1 Overview of the functions (RT Professional)****Overview**

MSG_SERVICE_NOTIFY_PROCPPlus (Page 2397)	Send and receive service (callback)
MSRTActivateMProtPlus (Page 2400)	Pause or continue protocol service
MSRTCheckWinFilterPlus (Page 2437)	Check the alarm filter conditions
MSRTCreateMsgInstancePlus (Page 2407)	Create alarm
MSRTCreateMsgInstanceWithCommentPlus (Page 2406)	Create alarm
AUTOHOTSPOT	Set comment text parameters
AUTOHOTSPOT	Set alarm filter parameters
MSRTDialogMsgLockPlus (Page 2410)	Set parameters for disabling alarms
MSRTEnumArchivDataPlus (Page 2389)	List alarms of a log
MSRTEnumBackupListPlus (Page 2449)	List export list entries
MSRTEnumGroupMsgPlus (Page 2431)	List alarm group individual alarms
MSRTEnumProtDataPlus (Page 2401)	List report data
MSRTExportPlus (Page 2451)	Export alarms from sequential logs
MSRTGetClassInfoPlus (Page 2415)	Query alarm class information
MSRTGetCommentInstancePlus (Page 2443)	Query comment text
MSRTGetFilterDataPlus (Page 2435)	Query alarm filter conditions
MSRTGetInfotextPlus (Page 2446)	Query info text
MSRTGetLastMsgWithCommentPlus (Page 2417)	Query last log alarm
MSRTGetMsgActualPlus (Page 2418)	Determine number of alarms
MSRTGetMsgCSDataPlus (Page 2420)	Query alarm configuration data
MSRTGetMsgPriorityPlus (Page 2421)	Query priority of the alarm
MSRTGetMsgQuitPlus (Page 2423)	Query alarms that require acknowledgment
MSRTGetMsgTextPlus (Page 2424)	Query alarm text block

MSRTGetSelectedMsgPlus (Page 2426)	Determine alarm number
MSRTLoopInAlarmPlus (Page 2430)	Loop-in-alarm
MSRTMsgWinCommandPlus (Page 2441)	Perform control function in alarm window
MSRTPrintMProtPlus (Page 2403)	Print alarm sequence report of pending alarms
MSRTQuitGroupPlus (Page 2434)	Acknowledge alarm group
MSRTQuitHornPlus (Page 2396)	Acknowledge audible signal device
MSRTResetMsgPlus (Page 2428)	Acknowledge alarm
MSRTSetCommentInstancePlus (Page 2445)	Specify comment text
MSRTSetInfotextPlus (Page 2448)	Specify info text
MSRTSetMsgFilterPlus (Page 2439)	Set alarm filter
MSRTSetMsgWinFilterPlus (Page 2440)	Set alarm filter for an alarm window
MSRTStartMsgServicePlus (Page 2391)	Start send and receive service (callback)
MSRTStopMsgServicePlus (Page 2394)	Stop send and receive service

See also

MSRTEnumMsgRTDataPlus (Page 2414)

MSRTLckGroupPlus (Page 2433)

3.10.1.2 Overview of structures (RT Professional)**Overview**

MSG_BACKUP_STRUCT_PLUS (Page 2385)	Export list
MSG_CLASS_STRUCT_PLUS (Page 2376)	Alarm class
MSG_COMMENT_INSTANCE_STRUCT_PLUS (Page 2383)	Comment
MSG_CSDATA_STRUCT_PLUS (Page 2378)	Single alarm
MSG_FILTER_STRUCT_PLUS (Page 2369)	Alarm filter
MSG_INFOTEXT_STRUCT_PLUS (Page 2384)	Info text
MSG_RTCREATE_STRUCT_PLUS (Page 2363)	Create an alarm in the runtime system
MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2365)	Single alarm
MSG_RTDATA_INSTANCE_STRUCT_PLUS (Page 2373)	Single alarm
MSG_RTGROUPENUM_STRUCT_PLUS (Page 2381)	Single alarms of an alarm group
MSG_RTGROUPSET_STRUCT_PLUS (Page 2382)	Alarm group data
MSG_RTLOCK_STRUCT_PLUS (Page 2387)	Alarm lock
MSG_TEXTVAL256_STRUCT_PLUS (Page 2388)	Associated text values

3.10.1.3 Error messages (RT Professional)

Overview

The following error messages can be returned by the API functions in the CMN_ERROR error structure:

Alarm Logging CS

MSG_ERR_CS_CONNECT	0x00001001	No connection to the Alarm Logging CS or to the project.
MSG_ERR_CS_PROJECT	0x00001002	Incorrect or invalid project
MSG_ERR_CS_PROJECTNAME	0x00001003	Incorrect or invalid project name
MSG_ERR_CS_PARAM	0x00001004	Incorrect parameter; dwError2 contains the number of the incorrect parameter.
MSG_ERR_CS_NODATA	0x00001005	No data found
MSG_ERR_CS_STRUCTFIELD	0x00001006	Error in a structure field; dwError2 contains the number of the structure field containing the incorrect value.
MSG_ERR_CS_DBCONNECT	0x00001011	No DBConnect possible
MSG_ERR_CS_DBCDELETE	0x00001012	Error while deleting (no data?)
MSG_ERR_CS_DBOPEN	0x00001013	Table with single alarms could not be opened
MSG_ERR_CS_SOURCES	0x00001014	Out of memory?
MSG_ERR_CS_CREATOR_ID	0x00001015	CreatorID is WinCC-CreatorID!
MSG_ERR_CS_NOTSUPPORTED	0x00001016	This function is not supported
MSG_ERR_UNKNOWN_ERROR	0x00001020	An unspecified error has occurred (WinCC V6.0 and higher)
MSG_ERR_API_TERMINATED	0x00001021	Processing was canceled prematurely (WinCC V6.0 and higher)
MSG_ERR_NO_MEMORY	0x00001022	Insufficient main memory (WinCC V6.0 and higher)
MSG_ERR_DB_FAIL	0x00001023	Inconsistent database (WinCC V6.0 and higher)
MSG_ERR_I_WRITE	0x00001024	Data could not be entered into the database. dwError2 and dwError3 contain further information. (WinCC V6.0 and higher)
MSG_ERR_CS_EXIST	0x00001025	Element already exists (WinCC V6.0 and higher)
MSG_ERR_CS_NO_EXIST	0x00001026	Element does not exist (WinCC V6.0 and higher)
MSG_ERR_HANDLE	0x00000090	Handle invalid

Alarm Logging RT

MSG_ERR_RT_NOTCONNECTED	0x00000001	Connection to the Alarm Logging RT not established
MSG_ERR_RT_NOINIT	0x00000002	Alarm Logging RT not initialized

3.10 Alarm functions (RT Professional)

MSG_ERR_RT_SERVICEMAX	0x00000003	Maximum number of services reached
MSG_ERR_MSG_NOEXIST	0x00000010	Alarm does not exist
MSG_ERR_MSG_NOTFOUND	0x00000011	Alarm not found
MSG_ERR_MSG_NOQUIT	0x00000012	Alarm does not require acknowledgment
MSG_ERR_MSG_ALREADYQUIT	0x00000013	Alarm already acknowledged
MSG_ERR_MSG_NOCLASS	0x00000014	Class information not available
MSG_ERR_MSG_STATE	0x00000015	Message status unknown
MSG_ERR_MSG_LOCKED	0x00000016	Alarm locked
MSG_ERR_MSG_DATETIME	0x00000017	Alarm has invalid date or invalid time
MSG_ERR_WIN_NAMENOTFOUND	0x00000020	Template name not found
MSG_ERR_WIN_CREATE	0x00000021	Error while creating the alarm window
MSG_ERR_WIN_DATA	0x00000022	Alarm window data invalid
MSG_ERR_VAR_BITEXCEEDS_	0x00000030	Bit for tag invalid
MSG_ERR_VAR_NOCHANGE_	0x00000031	No tag change
MSG_ERR_VAR_PROCESS_USED_	0x00000032	Event tag is already assigned
MSG_ERR_VAR_QUIT_USED_	0x00000033	Acknowledgment tag is already assigned
MSG_ERR_VAR_STATE_USED_	0x00000034	Status tag is already assigned
MSG_ERR_VAR_VARIANTCONV_	0x00000035	Error during variant conversion
MSG_ERR_VAR_USED_TYPDIF_	0x00000036	Event tag assigned, different types (Ack, Create, ...)
MSG_ERR_HANDLE	0x00000090	Handle invalid
MSG_ERR_FILTER	0x00000091	Filter condition invalid
MSG_ERR_API_PARAM	0x00000095	Parameter invalid
MSG_ERR_API_SERVICE	0x00000099	dwServiceID invalid
MSG_ERR_API_EXCEPTION	0x0000009a	An exception situation occurred in the code
MSG_ERR_API_DATETIME	0x0000009b	Date/time stamp invalid
MSG_ERR_API_SERVICEMAX	0x0000009c	Maximum number of services reached
MSG_ERR_API_NODATA	0x0000009d	No data
MSG_ERR_PROT_ID	0x00000111	Invalid protocol identifier
MSG_ERR_PROT_BLOCKS	0x00000110	No blocks in protocol
MSG_ERR_PROT_ACTIVE	0x00000112	Protocol service already active
MSG_ERR_NODEFAULTSERVER	0x00000401	No default server configured
MSG_ERR_NOLOCALSERVER	0x00000402	No local server available
MSG_ERR_NOSERVER	0x00000403	No default server configured and no local server available
MSG_ERR_NOMC	0x00000404	No multiclient project (not used here)
MSG_ERR_NOMCDEFAULTSERVER	0x00000405	No multiclient project and "@default" specified (not used here)

3.10.1.4 Constants (RT Professional)

General specifications

MSG_MAX_CLASS	(value: 18)	Number of alarm classes
MSG_MAX_CLASSTYPE	(value: 16)	Number of alarm types in alarm class
MSG_MAX_PVALUE	(value: 10)	Number of process values
MSG_MAX_TB	(value: 10)	Number of user texts
MSG_MAX_HITLIST	(value: 7)	Number of hit list element
MSG_MAX_TB_CONTENT	(value: 255)	Number of characters per user text
MSG_MAX_TEXTLEN	(value: 64)	Number of characters for all texts
MSG_MAX_TEXTVALUE	(value: 32)	Number of characters in associated text values
MSG_MAX_TEXTVALUE256	(value: 256)	Number of characters in associated text values
MSG_MAX_STATE	(value: 4)	Number of status texts
MSG_MAX_LOCKITEMS	(value: 16)	Number of alarms for a lock
MSG_MAX_LEVEL	(value: 6)	Number of nodes for group alarms
MSG_MAX_MSG_COUNT	(value: 150000) V6.2 and higher: (value: 50000)	Maximum number of single alarms in the project
MSG_MAX_GROUPITEMS	(value: 1000)	Number of alarms for a group with Enum
MSG_MAX_USERNAME	(value: 16)	Maximum length of a user group name
MSG_MAX_APPLNAME	(value: 32)	Maximum length of an application name
MSG_MAX_INSTANCE	(value: 512)	Instance name
MSG_MAX_ARCHIV_QUEUE	(value: 1500)	Max. number of bufferable log entries

Alarm states

MSG_STATE_COME	(value: 0x00000001)	Incoming alarm
MSG_STATE_GO	(value: 0x00000002)	Outgoing alarm
MSG_STATE_QUIT	(value: 0x00000003)	Alarm acknowledged
MSG_STATE_LOCK	(value: 0x00000004)	Alarm locked
MSG_STATE_UNLOCK	(value: 0x00000005)	Alarm released
MSG_STATE_HIDE	(value: 0x0000000A)	Alarm hidden
MSG_STATE_UNHIDE	(value: 0x0000000B)	Alarm displayed
MSG_STATE_HIDE_MANUAL	(value: 0x0000000C)	Alarm hidden manually
MSG_STATE_UNHIDE_MANUAL	(value: 0x0000000D)	Alarm displayed manually

MSG_STATE_QUIT_SYSTEM	(value: 0x00000010)	Alarm acknowledged by system
MSG_STATE_QUIT_EMERGENCY	(value: 0x00000011)	Emergency acknowledgment
MSG_STATE_QUIT_HORN	(value: 0x00000012)	Horn acknowledgment
MSG_STATE_COMEGO	(value: 0x00000013)	Incoming and outgoing alarm, display in alarm list only
MSG_STATE_RESET	(value: 0x00020000)	Bit for alarm reset
MSG_STATE_SUMTIME	(value: 0x00040000)	Bit for daylight saving time active
MSG_STATE_INSTANCE	(value: 0x00080000)	Bit for instance alarm (n alarms of a no.)
MSG_STATE_TIMEINVALID	(value: 0x00100000)	Bit for invalid date/time stamp OR operation with (MSG_STATE_COME, MSG_STATE_GO, ...)
MSG_STATE_HIDDEN	(value: 0x02000000)	Bit for currently hidden alarm, available in V6.2 and higher

Flags of an alarm in the database table

MSG_FLAG_SUMTIME	(value: 0x00000001)	Daylight saving time active
MSG_FLAG_COMMENT	(value: 0x00000002)	Alarm has comments
MSG_FLAG_ARCHIV	(value: 0x00000004)	Log
MSG_FLAG_PROTOCOL	(value: 0x00000008)	Report
MSG_FLAG_TEXTVALUES	(value: 0x00000010)	Alarm has associated text values
MSG_FLAG_TIMEINVALID	(value: 0x00000020)	Bit for invalid date/time stamp
MSG_FLAG_INSTANCE	(value: 0x00000040)	Marking of instance alarms
MSG_FLAG_HIDDEN	(value: 0x00000080)	Hidden alarm, available in V6.2 and higher
MSG_FLAG_TEXT10_EXT	(value: 0x00000100)	Extended channel diagnostics, available in V7.0 and higher
MSG_FLAG_INSTANCE10	(value: 0x08000000)	Instance alarm with process value 10, available in V5.0 SP2 and higher
MSG_FLAG_TEXTREF1	(value: 0x10000000)	Text reference 1, available in V5.0 SP2 and higher
MSG_FLAG_TEXTREF2	(value: 0x20000000)	Text reference 2, available in V5.0 SP2 and higher

Flags of an alarm in the online table

Only defined in m_global.h starting in WinCC V6.0 SP4.

MSG_RT_FLAG_COME	(value: 0x00000001)	Incoming alarm
MSG_RT_FLAG_GO	(value: 0x00000002)	Outgoing alarm
MSG_RT_FLAG_QUIT_COME	(value: 0x00000004)	Incoming acknowledgment
MSG_RT_FLAG_QUIT_GO	(value: 0x00000008)	Outgoing acknowledgment
MSG_RT_FLAG_NOQUIT	(value: 0x00000010)	No acknowledgment

3.10 Alarm functions (RT Professional)

MSG_RT_FLAG_QUIT_NOGO	(value: 0x00000020)	TM II acknowledgment
MSG_RT_FLAG_FLASH	(value: 0x00000040)	Flashing
MSG_RT_FLAG_FIRSTFLASH	(value: 0x00000080)	Only first alarm is permitted to flash
MSG_RT_FLAG_NOLIST	(value: 0x00000100)	Alarms are not received in the message list
MSG_RT_FLAG_UNIQUE_USER	(value: 0x00000200)	Alarms with unique user name
MSG_RT_FLAG_ONL_COM_COMMENT	(value: 0x00000400)	Display only comments of incoming alarm in alarm list
MSG_RT_FLAG_HIDDEN	(value: 0x00000800)	Alarm hidden
MSG_RT_FLAG_TEXT10_EXT	(value: 0x00001000)	Extended channel diagnostics, available in V7.0 and higher
MSG_RT_FLAG_DELETE	(value: 0x00010000)	Alarm can be deleted
MSG_RT_FLAG_SUMTIME	(value: 0x00020000)	Daylight saving time flag
MSG_RT_FLAG_SQUIT	(value: 0x00040000)	Alarm has group acknowledgment
MSG_RT_FLAG_QUITNR	(value: 0x00080000)	Alarm number - Ack
MSG_RT_FLAG_MSGLIST	(value: 0x00100000)	Alarm from alarm list
MSG_RT_FLAG_COMMENT	(value: 0x00200000)	Alarm has comments
MSG_RT_FLAG_ARCHIV	(value: 0x00400000)	Alarm is logged
MSG_RT_FLAG_PROTOCOL	(value: 0x00800000)	Alarm is logged
MSG_RT_FLAG_INSTANCE	(value: 0x01000000)	Flag for instance alarm
MSG_RT_FLAG_TIMEINVALID	(value: 0x02000000)	Flag for invalid date/time stamp
MSG_RT_FLAG_LOCKGROUP	(value: 0x04000000)	Flag for group lock
MSG_RT_FLAG_INSTANCE10	(value: 0x08000000)	Bit for new instance alarm with process value 10.
MSG_RT_FLAG_TEXTREF1	(value: 0x10000000)	Bit for text reference 1
MSG_RT_FLAG_TEXTREF2	(value: 0x20000000)	Bit for text reference 2
MSG_RT_FLAG_QUITCOUNTER	(value: 0x40000000)	Bit for acknowledgment via counter
MSG_RT_FLAG_QUITGROUP	(value: 0x80000000)	Bit for acknowledgment via counter

Flags for single alarms

MSG_PARAMS_SQUIT	(value: 0x00000001)	Alarm is subject to single acknowledgment.
MSG_PARAMS_HORN	(value: 0x00000002)	Alarm activates horn.
MSG_PARAMS_ARCHIV	(value: 0x00000004)	Alarm is to be logged.
MSG_PARAMS_PROTOCOL	(value: 0x00000008)	Alarm is to be reported.
MSG_PARAMS_VARFLANK	(value: 0x00000010)	Alarm is generated on falling edge of tag.
MSG_PARAMS_LOCKED	(value: 0x00000020)	Alarm is locked during startup.
MSG_PARAMS_INFOTEXT	(value: 0x00000040)	Alarm has an info text.

MSG_PARAMS_LOOPIN- LARM	(value: 0x00000080)	Alarm has loop-in-alarm.
MSG_PARAMS_APFUNC	(value: 0x00000100)	Alarm activates global AP function.
MSG_PARAMS_NORMDLL	(value: 0x00000200)	Alarm has a normalization DLL.
MSG_PARAMS_TEXT10_EXT	(value: 0x00000400)	Extended channel diagnostics, available in V7.0 and higher.

State machines

MSG_QUIT_QUIT_COME	(value: 0x00000004)	Incoming acknowledgment
MSG_QUIT_QUIT_GO	(value: 0x00000008)	Outgoing acknowledgment
MSG_QUIT_NOQUIT	(value: 0x00000010)	No acknowledgment
MSG_QUIT_QUIT_NOGO	(value: 0x00000020)	TM II acknowledgment (without outgoing alarm)
MSG_QUIT_FLASH	(value: 0x00000040)	Flashing
MSG_QUIT_FIRSTFLASH	(value: 0x00000080)	Only first alarm is permitted to flash
MSG_QUIT_NOLIST	(value: 0x00000100)	Alarms are not received in the message list
MSG_QUIT_UNIQUE_USER	(value: 0x00000200)	Alarms with unique user name
MSG_QUIT_ONL_COM_COME	(value: 0x00000400)	Display only comments of incoming alarm in alarm list

Group alarms

MSG_COLLECT_ALL	(value: 0x00000001)	All alarms
MSG_COLLECT_QUIT	(value: 0x00000002)	All unacknowledged alarms
MSG_COLLECT_CLASS	(value: 0x00000003)	All alarms of an alarm class
MSG_COLLECT_TYPE	(value: 0x00000004)	All alarms of an alarm type
MSG_COLLECT_USER	(value: 0x00000005)	User-defined group alarms

Single alarms

MSG_SINGLE_OPERATION	(value: 0x00BEDBED)	Operator input alarm

Alarm classes

MSG_CLASS_SYSTEM_QUIT	(value: 0x00000011)	System alarm class with acknowledgment
MSG_CLASS_SYSTEM	(value: 0x00000012)	System alarm class without acknowledgment

Alarm blocks

The main type of the alarm blocks is available in the HIWORD. The LOWORD includes the subtypes of the alarm blocks.

MSG_BLOCK_SYSTEM	(value: 0x00010000)	System blocks
MSG_BLOCK_DATE	(value: 0x00010001)	Date
MSG_BLOCK_TIME	(value: 0x00010002)	Time of day
MSG_BLOCK_TIMEDIFF	(value: 0x00010003)	Alarm duration
MSG_BLOCK_SUMWIN	(value: 0x00010004)	Daylight-saving time/standard time
MSG_BLOCK_STATE	(value: 0x00010005)	Alarm status
MSG_BLOCK_QUIT	(value: 0x00010006)	Acknowledgment status
MSG_BLOCK_NUMBER	(value: 0x00010007)	Alarm number
MSG_BLOCK_CLASS	(value: 0x00010008)	Alarm class
MSG_BLOCK_TYP	(value: 0x00010009)	Alarm type
MSG_BLOCK_AGCPU	(value: 0x0001000A)	Controller/CPU number
MSG_BLOCK_DMVAR	(value: 0x0001000B)	Alarm tag
MSG_BLOCK_LIMIT	(value: 0x0001000C)	Limit exceeded
MSG_BLOCK_ARCHIV	(value: 0x0001000D)	Log identifier
MSG_BLOCK_PROTOCOL	(value: 0x0001000E)	Report identifier
MSG_BLOCK_COMMENT	(value: 0x0001000F)	Comment
MSG_BLOCK_INFO	(value: 0x00010010)	Info text
MSG_BLOCK_LOOPINALARM	(value: 0x00010011)	Loop-in-alarm/AP function
MSG_BLOCK_COMPUTER-NAME	(value: 0x00010012)	Computer name
MSG_BLOCK_USERNAME	(value: 0x00010013)	User name
MSG_BLOCK_PRIORITY	(value: 0x00010014)	Priority
MSG_BLOCK_TEXT	(value: 0x00020000)	The maximum amount of 16 user text blocks can be activated using the MSG_BLOCK_TEXT1 to MSG_BLOCK_TEXT16 constants with values 0x00020001 to 0x00020010.
MSG_BLOCK_VALUE	(value: 0x00030000)	The maximum amount of 16 process value text blocks can be activated using the MSG_BLOCK_VALUE1 to MSG_BLOCK_VALUE16 constants with values 0x00030001 to 0x00030010.
MSG_MAX_SYSBL		Number of system alarm blocks = MSG_BLOCK_LOOPINALARM - MSG_BLOCK_SYSTEM
MSG_MAX_SYSBL_V6		Number of system blocks in V6.0 and higher = MSG_BLOCK_PRIORITY - MSG_BLOCK_SYSTEM

MSG_MAX_BLOCKS		Total number of alarm blocks = (MSG_MAX_TB + MSG_MAX_PVALUE + MSG_MAX_SYSBL)
MSG_MAX_BLOCKS_V6		Total number of alarm blocks in V6.0 and higher = (MSG_MAX_TB + MSG_MAX_PVALUE + MSG_MAX_SYSBL_V6)

Date format

DATE_FORMAT_YEAR_2	(value: 0x00000001)	Two-digit year specification
DATE_FORMAT_YEAR_4	(value: 0x00000002)	Four-digit year specification
DATE_FORMAT_DMY	(value: 0x00000004)	Order: Day, month, year
DATE_FORMAT_MDY	(value: 0x00000008)	Order: Month, day, year
DATE_FORMAT_YMD	(value: 0x00000010)	Order: Year, month, day

Time of day format

TIME_FORMAT_HOUR	(value: 0x00000001)	Hour
TIME_FORMAT_MIN	(value: 0x00000002)	Minute
TIME_FORMAT_SEC	(value: 0x00000004)	Second
TIME_FORMAT_MSEC	(value: 0x00000008)	Millisecond
TIME_FORMAT_24HOUR	(value: 0x00000010)	24-hour format
TIME_FORMAT_DAY	(value: 0x00000020)	Hour display

Alarm number - format

The formatting rule (with leading zeros) is available in the HiWord; the LoWord contains the number of digits.

MSGNR_FORMAT_NOLEADING	(value: 0x00000000)	
MSGNR_FORMAT_LEADING-ZERO	(value: 0x00010000)	
MSG_BLOCK_FORMAT_RIGHT	(value: 0x00020000)	Right-justified output
MSG_BLOCK_FORMAT_CENTER	(value: 0x00040000)	Centered output
MSG_BLOCK_FORMAT_FLASH	(value: 0x80000000)	Alarm block flashing

Elements of the status bar

MSG_SB_HELP	(value: 0x00000001)	Help line
MSG_SB_DATE	(value: 0x00000002)	Date
MSG_SB_TIME	(value: 0x00000004)	Time of day
MSG_SB_COUNTALL	(value: 0x00000008)	All alarms
MSG_SB_COUNTSEL	(value: 0x00000010)	Current alarms
MSG_SB_COUNTQUIT	(value: 0x00000020)	Acknowledged alarms
MSG_SB_SELECT	(value: 0x00000040)	Selection active
MSG_SB_LOCK	(value: 0x00000080)	Lock active
MSG_SB_HIDE	(value: 0x00000100)	Hiding of alarms active
MSG_SB_WARNING	(value: 0x00000200)	Hiding information
MSG_MAX_SBITEMS	(value: 10)	Maximum number of elements in the status bar

Elements of the symbol bar

MSG_TB_MSGWIN	(value: 0x00000001)	Display "Alarm window" button.
MSG_TB_ARC_S	(value: 0x00000002)	Display "Circular log" button.
MSG_TB_ARC_L	(value: 0x00000004)	Display "Sequential log" button.
MSG_TB_QH	(value: 0x00000008)	Display "Horn acknowledgment" button.
MSG_TB_QM	(value: 0x00000010)	Display "Single acknowledgment" button.
MSG_TB_QS	(value: 0x00000020)	Display "Group acknowledgment" button.
MSG_TB_SCROLL	(value: 0x00000040)	Display "Scroll function On/Off" button.
MSG_TB_SELECT	(value: 0x00000080)	Display "Selection dialog" button.
MSG_TB_LOCK	(value: 0x00000100)	Display "Lock dialog" button.
MSG_TB_PROTOCOL	(value: 0x00000200)	Display "Report dialog" button.
MSG_TB_CITYRUF	(value: 0x00000400)	Display "City call" button.
MSG_TB_RESET	(value: 0x00000800)	Display "Reset button" button.
MSG_TB_MSGFIRST	(value: 0x00001000)	Display "Start of list" button.
MSG_TB_MSGLAST	(value: 0x00002000)	Display "End of list" button.
MSG_TB_MSGNEXT	(value: 0x00004000)	Display "Next alarm" button.
MSG_TB_MSGPREV	(value: 0x00008000)	Display "Previous alarm" button.
MSG_TB_INFO	(value: 0x00010000)	Display "Info texts" button.
MSG_TB_COMMENT	(value: 0x00020000)	Display "Enter comment" button.
MSG_TB_LOOPINALARM	(value: 0x00040000)	Display "Loop-in-alarm" button.
MSG_TB_LANGUAGESET	(value: 0x00080000)	Display "Language switch" button.
MSG_TB_PRINT	(value: 0x00100000)	Display "Print" button.
MSG_TB_LOCKWIN	(value: 0x00200000)	Display "Lock window" button.

MSG_TB_UNLOCKMSG	(value: 0x00400000)	Display "Release alarm" button.
MSG_TB_LOCK_UNLOCK_MSG	(value: 0x00400000)	Display "Lock or release alarm" button.
MSG_TB_SORTDLG	(value: 0x00800000)	Display "Sorting dialog" button.
MSG_TB_TIMEBASE	(value: 0x01000000)	Display "Dialog for timebase" button.
MSG_TB_HITLIST	(value: 0x02000000)	Display "Hit list" button.
MSG_TB_HIDELIST	(value: 0x04000000)	Display "List of hidden messages" button
MSG_TB_UNHIDEMSG	(value: 0x08000000)	Display "Display alarm" button.
MSG_TB_HIDE_UNHIDE_MSG	(value: 0x08000000)	Display "Hide and display alarm" button.
MSG_TB_HIDEDLG	(value: 0x10000000)	Display "Hide and display alarm dialog" button.
MSG_MAX_TBITEMS	(value: 24)	Maximum number of operator controls in the status bar.
MSG_MAX_TBITEMS_V6	(value: 25)	Maximum number, available in V6 and higher
MSG_MAX_TBITEMS_V6_1	(value: 26)	Maximum number, available in V6.1 and higher
MSG_MAX_TBITEMS_V6_2	(value: 30)	Maximum number, available in V6.2 and higher
MSG_ALIGN_TOP	(value: 1)	Top alignment
MSG_ALIGN_BOTTOM	(value: 2)	Bottom alignment
MSG_ALIGN_LEFT	(value: 3)	Left alignment
MSG_ALIGN_RIGHT	(value: 4)	Right alignment

Window options

MSG_MWWIN_SWITCHENABLE	(value: 0x00000001)	Process value switching
MSG_MWWIN_PROCESS	(value: 0x00000002)	Process alarm window
MSG_MWWIN_ARCHIV_CIRCLE	(value: 0x00000004)	Circular log
MSG_MWWIN_ARCHIV_LONG	(value: 0x00000008)	Sequential log
MSG_MWWIN_GRIDLINE_HORIZ	(value: 0x00000010)	Grid line horizontal
MSG_MWWIN_GRIDLINEVERT	(value: 0x00000020)	Grid line vertical
MSG_MWWIN_LINETITLE	(value: 0x00000040)	Line label
MSG_MWWIN_LINEHEIGHT	(value: 0x00000080)	Line height adjustable
MSG_MWWIN_LINEHEIGHT_FONT	(value: 0x00000100)	Adapt font to line height
MSG_MWWIN_COLUMNNTITLE	(value: 0x00000200)	Column label
MSG_MWWIN_COLUMNNTITLE2	(value: 0x00000400)	2 title lines
MSG_MWWIN_COLUMNNTITLE3	(value: 0x00000800)	3 title lines

3.10 Alarm functions (RT Professional)

MSG_MWWIN_COLUMN-NWIDTH	(value: 0x00001000)	Column width adjustable
MSG_MWWIN_COLUMNMOVE	(value: 0x00002000)	Columns may be moved
MSG_MWWIN_CELLSELECT	(value: 0x00004000)	Selection cell only (default setting: line)
MSG_MWWIN_LINESELECT	(value: 0x00008000)	Line selection
MSG_MWWIN_AUTOSCROLL-OFF	(value: 0x00010000)	Auto scroll Off (default setting: On)
MSG_MWWIN_TITLECUT	(value: 0x00020000)	Title will be truncated with periods if necessary
MSG_MWWIN_CELL CUT	(value: 0x00040000)	Cell content truncated with periods if necessary
MSG_MWWIN_TITLEBAR	(value: 0x00080000)	Display title line of window (+ frame)
MSG_MWWIN_SIZEABLE	(value: 0x00100000)	Size of window adjustable
MSG_MWWIN_TOOLTIP	(value: 0x00200000)	Tool tip display specially for alarm texts
MSG_MWWIN_LOOP_IN_ALARM_DC	(value: 0x00400000)	One click for "Loop-in-alarm"
MSG_MWWIN_SELECTFOCUSRECT	(value: 0x00800000)	Selection made with frame instead of inversion
MSG_MWWIN_OPMSG_HIDE	(value: 0x01000000)	Operator input alarm when hiding
MSG_MWWIN_OPMSGUNHIDE	(value: 0x02000000)	Operating input alarm when displaying
MSG_MWWIN_INSTANCE_COLWIDTH	(value: 0x08000000)	Instance-specific column width
MSG_MWWIN_INSTANCE_BLOCKINFO	(value: 0x10000000)	Instance-specific alarm block settings
MSG_MWWIN_OPMSG_LOCK	(value: 0x20000000)	Operator input alarm when locking
MSG_MWWIN_OPMSG_UNLOCK	(value: 0x40000000)	Operator input alarm when releasing
MSG_MWWIN_OPMSG_ACK	(value: 0x80000000)	Operator input alarm when acknowledging

Filter conditions

MSG_FILTER_DATE_FROM	(value: 0x00000001)	Date from
MSG_FILTER_DATE_TO	(value: 0x00000002)	Date to
MSG_FILTER_TIME_FROM	(value: 0x00000004)	Time from
MSG_FILTER_TIME_TO	(value: 0x00000008)	Time to
MSG_FILTER_NR_FROM	(value: 0x00000010)	Alarm number from
MSG_FILTER_NR_TO	(value: 0x00000020)	Alarm number to
MSG_FILTER_CLASS	(value: 0x00000040)	Alarm classes
MSG_FILTER_STATE	(value: 0x00000080)	Alarm status
MSG_FILTER_AG_FROM	(value: 0x00000100)	AG number from

MSG_FILTER_AG_TO	(value: 0x00000200)	AG number to
MSG_FILTER_AGSUB_FROM	(value: 0x00000400)	AG sub-number from
MSG_FILTER_AGSUB_TO	(value: 0x00000800)	AG sub-number to
MSG_FILTER_ARC	(value: 0x00001000)	Log/report identifier
MSG_FILTER_TEXT	(value: 0x00002000)	Alarm texts
MSG_FILTER_PVALUE	(value: 0x00004000)	Process values
MSG_FILTER_COUNTER_FROM	(value: 0x00008000)	Internal alarm counter from
MSG_FILTER_COUNTER_TO	(value: 0x00010000)	Internal alarm counter to
MSG_FILTER_PROCESSMSG	(value: 0x00020000)	Process alarms
MSG_FILTER_SYSMSG	(value: 0x00040000)	System events
MSG_FILTER_BEDMSG	(value: 0x00080000)	Operator input alarms
MSG_FILTER_QUICKSELECT	(value: 0x00100000)	Quick selection (current day, etc.)
MSG_FILTER_TEXT_EQUAL	(value: 0x00200000)	Exact alarm texts
MSG_FILTER_PRIORITY_FROM	(value: 0x00400000)	Priority from
MSG_FILTER_PRIORITY_TO	(value: 0x00800000)	Priority to
MSG_FILTER_VISIBLEONLY	(value: 0x01000000)	Show displayed alarms
MSG_FILTER_HIDDENONLY	(value: 0x02000000)	Show hidden alarms
MSG_FILTER_NODYNAMIC	(value: 0x04000000)	No selection of dynamic texts
MSG_FILTER_DATE	Value: (MSG_FILTER_DATE_FROM MSG_FILTER_DATE_TO)	
MSG_FILTER_TIME	Value: (MSG_FILTER_TIME_FROM MSG_FILTER_TIME_TO)	
MSG_FILTER_NR	Value: (MSG_FILTER_NR_FROM MSG_FILTER_NR_TO)	

Quick selection

The end time of the quick selection refers to the current system time of the local computer. The start time is back calculated $n * (\text{months, days, hours})$.

MSG_FILTER_QUICK_MONTH	(value: 0x00000001)	Last n months
MSG_FILTER_QUICK_DAYS	(value: 0x00000002)	Last n days
MSG_FILTER_QUICK_HOUR	(value: 0x00000003)	Last n hours

Log types

MSG_ARPRO_LONGARCH	(value: 0x00000001)	Sequential log
MSG_ARPRO_SHORTARCH	(value: 0x00000002)	Circular log
MSG_ARPRO_ARCHPROT	(value: 0x00000003)	Log report
MSG_ARPRO_MFPROT	(value: 0x00000004)	Alarm sequence report

Flags for logs

MSG_ARPRO_PARAM_HSPONLY	(value: 0x00000001)	Short-term log in main memory
MSG_ARPRO_PARAM_CIRCLE	(value: 0x00000002)	Circular log
MSG_ARPRO_PARAM_ENDLES	(value: 0x00000004)	Endless log
MSG_ARPRO_PARAM_RELOAD	(value: 0x00000008)	Reload after power failure
MSG_ARPRO_PARAM_LINEPRINT	(value: 0x00000010)	Print linewise (alarm sequence report only)

IDs for locking alarms

MSG_LOCK_UNKNOWN	(value: 0)	Lock is unknown
MSG_LOCK_CLASS	(value: 1)	Locks of alarm classes
MSG_LOCK_GROUP1	(value: 2)	Locks of alarm group level 1
MSG_LOCK_GROUP2	(value: 3)	Locks of alarm group level 2

IDs for logs

MSG_PROT_MSG	(value: 1)	Alarm sequence report
MSG_PROT_ARCHIV_CIRCLE	(value: 2)	Log report of circular log
MSG_PROT_ARCHIV_LONG	(value: 3)	Log report of sequential log

IDs for export of alarm logs

MSG_DB_FMT_CSV	(value: 1)	CSV file format
MSG_DB_FMT_DBASE3	(value: 2)	DBASE3 file format
MSG_DB_FMT_WATCOM	(value: 3)	Watcom format
MSG_DB_FLAGS_OVERWRITE	(value: 1)	Overwrite existing data records
MSG_DB_FLAGS_ADD	(value: 2)	Add only
MSG_DB_FLAGS_DELETE	(value: 4)	Delete source data records
MSG_DB_FLAGS_NOEXPORT	(value: 8)	Do not export data records --> Delete only (no longer available)

Flags for generic configuration

The flags can be logically combined.

MSG_GENERIC_APPEND	(value: 0x00000001)	Create new
MSG_GENERIC_OVERWRITE	(value: 0x00000002)	Overwrite
MSG_GENERIC_DELETE	(value: 0x00000004)	Delete

Mode for tags

MSG_CS_VAR_NOTSET	(value: 0x0000)	No tag set
MSG_CS_VARNAME_SET	(value: 0x0001)	Tag name is set
MSG_CS_VARID_SET	(value: 0x0002)	Tag ID is set
MSG_CS_VARALL_SET	(value: 0x0003)	Tag ID and name are set

Enumeration of the logs

MSG_ARCHIV_ENUM_DESC	(value: 0x00000001)	Sorted in descending order (default setting: ascending)

3.10.2 Structures (RT Professional)**3.10.2.1 MSG_RTCREATE_STRUCT_PLUS (RT Professional)****Declaration**

```
typedef struct {
    DWORD                dwMsgState;
    DWORD                dwMsgNrLow;
    DWORD                dwMsgNrHigh;
    DWORD                dwFlags;
    SYSTEMTIME           stMsgTime;
    double               dPValue[MSG_MAX_PVALUE];
    WORD                 wPValueUsed;
    MSG_TEXTVAL256_STRUCT_PLUS mtTextValueW[MSG_MAX_PVALUE];
    WORD                 wTextValueUsed;
}
MSG_RTCREATE_STRUCT_PLUS;
```

Members**dwMsgState**

Alarm status in compliance with the constants in the "m_global.h" file:

MSG_STATE_COME	Incoming alarm
MSG_STATE_GO	Outgoing alarm
MSG_STATE_QUIT	Alarm acknowledged
MSG_STATE_LOCK	Alarm locked
MSG_STATE_UNLOCK	Alarm released
MSG_STATE_HIDE	Alarm hidden

MSG_STATE_UNHIDE	Alarm displayed
MSG_STATE_HIDE_MANUAL	Alarm hidden manually
MSG_STATE_UNHIDE_MANUAL	Alarm displayed manually

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

dwFlags

The parameter is reserved for future development and must be preset to 0.

stMsgTime

Date and time for coming-in or going-out of an alarm. The system time of the PLC or the OS will be used when acknowledging.

If the current timer for transferring a SYSTEMTIME parameter is required, the GetLocalTime must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

dPValue

Numeric process values.

wPValueUsed

Process values used in bit coded layout.

Each bit should only be set in one of the two members wPValueUsed or wTextValueUsed, because only number or text is technically possible as an associated value. In the case of erroneous double entry the dPValue has priority.

mtTextValue

Structure of the MSG_TEXTVAL256_STRUCT_PLUS (Page 2388) type with the associated text values.

wTextValueUsed

Text process values used in bit coded layout.

Each bit should only be set in one of the two members wPValueUsed or wTextValueUsed, because only number or text is technically possible as an associated value. In the case of erroneous double entry the dPValue has priority.

Comment

Before you fill out the structure, you should initialize it with "0".

Required files

CCMSRTCLIPlus.h

API functions

AUTOHOTSPOT	Create alarm

See also

MSG_TEXTVAL256_STRUCT_PLUS (Page 2388)

Overview of structures (Page 2347)

3.10.2.2 MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {
    DWORD                dwMsgState;
    DWORD                dwMsgNrLow;
    DWORD                dwMsgNrHigh;
    SYSTEMTIME           stMsgTime;
    DWORD                dwTimeDiff;
    DWORD                dwCounter;
    DWORD                dwFlags;
    WORD                 wPValueUsed;
    WORD                 wTextValueUsed;
    double               dPValue[MSG_MAX_PVALUE];
    MSG_TEXTVAL256_STRUCT_PLUS mtTextValue[MSG_MAX_PVALUE];
    WCHAR                szInstance[MSG_MAX_INSTANCE+1];
    WCHAR                szComment[MSG_MAX_TB_CONTENT+1];
    WCHAR                szUser[MSG_MAX_USERNAME+1];
    WCHAR                szComputerName[MAX_COMPUTERNAME_LENGTH+1];
    WCHAR                szApplicationName[MSG_MAX_APPLNAME+1];
    WCHAR                szServerPrefix[_MAX_PATH+1];
}
MSG_RTDATA_INSTANCECOMMENT_STRUCTPlus;
```

Members

dwMsgState

Alarm status in compliance with the constants in the "m_global.h" file:

The statuses listed will only be used internally by the system and can only be set by the API using MSRTCreateMsgInstanceWithCommentPlus.

MSG_STATE_COME	Incoming alarm
MSG_STATE_GO	Outgoing alarm
MSG_STATE_QUIT	Alarm acknowledged
MSG_STATE_LOCK	Alarm locked
MSG_STATE_UNLOCK	Alarm released

3.10 Alarm functions (RT Professional)

MSG_STATE_HIDE	Alarm hidden
MSG_STATE_UNHIDE	Alarm displayed
MSG_STATE_HIDE_MANUAL	Alarm hidden manually
MSG_STATE_UNHIDE_MANUAL	Alarm displayed manually

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

stMsgTime

Date and time for coming-in or going-out of an alarm.

dwTimeDiff

Duration coming in or going out, telegram time in seconds.

dwCounter

Internal alarm counter.

dwFlags

Flags in compliance with the constants in the "m_global.h" file:

Flags for alarms in the alarm view		
MSG_RT_FLAG_COME	0x00000001	Incoming alarm
MSG_RT_FLAG_GO	0x00000002	Outgoing alarm
MSG_RT_FLAG_QUIT_COME	0x00000004	Incoming acknowledgment
MSG_RT_FLAG_QUIT_GO	0x00000008	Outgoing acknowledgment
MSG_RT_FLAG_NOQUIT	0x00000010	No acknowledgment
MSG_RT_FLAG_QUIT_NOGO	0x00000020	TM II acknowledgment
MSG_RT_FLAG_FLASH	0x00000040	Flashing
MSG_RT_FLAG_FIRSTFLASH	0x00000080	Only first alarm is permitted to flash
MSG_RT_FLAG_DELETE	0x00010000	Alarm can be deleted
MSG_RT_FLAG_SUMTIME	0x00020000	Daylight saving time flag
MSG_RT_FLAG_SQUIT	0x00040000	Alarm has group acknowledgment
MSG_RT_FLAG_QUITNR	0x00080000	Alarm number acknowledgment
MSG_RT_FLAG_MSGLIST	0x00100000	Alarm from the alarm list
MSG_RT_FLAG_COMMENT	0x00200000	Alarm has comments
MSG_RT_FLAG_ARCHIV	0x00400000	Alarm is logged
MSG_RT_FLAG_PROTOCOL	0x00800000	Alarm is reported
MSG_RT_FLAG_INSTANCE	0x01000000	Flag for instance alarm
MSG_RT_FLAG_TIMEINVALID	0x02000000	Flag for invalid date/time stamp
MSG_RT_FLAG_LOCKGROUP	0x04000000	Flag for group lock
MSG_RT_FLAG_INSTANCE10	0x08000000	Bit for new instance alarm with process value 10.

Flags for alarms in the alarm view		
MSG_RT_FLAG_TEXTREF1	0x10000000	Bit for text reference 1.
MSG_RT_FLAG_TEXTREF2	0x20000000	Bit for text reference 2.
MSG_RT_FLAG_QUITCOUNTER	0x40000000	Bit for acknowledgment via counter
MSG_RT_FLAG_QUITGROUP	0x80000000	Bit for acknowledgment via counter

With the MSRTCreateMsgInstanceWithCommentPlus function, flags cannot be sent to the alarm server. Flags are ignored.

wPValueUsed

Process values used in bit coded layout.

Each bit should only be set in one of the two members wPValueUsed or wTextValueUsed, because only number or text is technically possible as an associated value. In the case of erroneous double entry the dPValue has priority.

wTextValueUsed

Text values used, bit-coded.

Each bit should only be set in one of the two members wPValueUsed or wTextValueUsed, because only number or text is technically possible as an associated value.

In the case of erroneous double entry the dPValue has priority.

dPValue

Process values.

mtTextValue

Structure of the MSG_TEXTVAL256_STRUCT_PLUS (Page 2388) type with the associated text values.

szInstance

Instance name. If szInstance is not provided (empty string), a normal alarm is created

szComment

Comment name

szUser

User name.

szComputerName

Computer name.

szApplicationName

Application name.

szServerPrefix

Server prefix.

Comment

Before you fill out the structure, you should initialize it with "0".

Required files

CCMSRTCLIPlus.h

API functions

MSRTCreateMsgInstanceWithCommentPlus (Page 2406)	Create alarm
MSRTGetLastMsgWithCommentPlus (Page 2417)	Query last archived alarm
MSG_SERVICE_NOTIFY_PROCPPlus (Page 2397)	Callback function

See also

MSRTCreateMsgInstanceWithCommentPlus (Page 2406)

MSRTGetLastMsgWithCommentPlus (Page 2417)

MSG_SERVICE_NOTIFY_PROCPPlus (Page 2397)

MSRTGetSelectedMsgPlus (Page 2426)

MSRTCheckWinFilterPlus (Page 2437)

Overview of structures (Page 2347)

3.10.2.3 MSG_FILTER_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {
    WCHAR          szFilterName[MSG_MAX_TEXTLEN+1];
    DWORD          dwFilter;
    SYSTEMTIME     st[2];
    DWORD          dwMsgNrLow[2];
    DWORD          dwMsgNrHigh[2];
    VARIANT        vMsgClasses;
    DWORD          dwMsgState;
    WORD           wAGNr[2];
    WORD           wAGSubNr[2];
    DWORD          dwArchivMode;
    WCHAR          szTB[MSG_MAX_TB][MSG_MAX_TB_CONTENT+1];
    DWORD          dwTB;
    double         dPValue[MSG_MAX_PVALUE][2];
    DWORD          dwPValue[2];
    ULONGLONG      ullMsgCounter[2]; /* in Script it has the union type
MSGULONGLONG, see CCMsrtCliPlus.h */
    DWORD          dwQuickSelect;
    DWORD          dwPriority[2];
    WCHAR          szInstance[MSG_MAX_INSTANCE+1];
}
MSG_FILTER_STRUCT_PLUS;
```

Members

szFilterName

Filter name

dwFilter

The filter conditions are defined by means of the following constants from the file "m_global.h":

MSG_FILTER_DATE_FROM	Date from
MSG_FILTER_DATE_TO	Date to
MSG_FILTER_TIME_FROM	Time from
MSG_FILTER_TIME_TO	Time to
MSG_FILTER_NR_FROM	Alarm number from
MSG_FILTER_NR_TO	Alarm number to
MSG_FILTER_CLASS	Alarm classes
MSG_FILTER_STATE	Alarm status
MSG_FILTER_AG_FROM	AG number from
MSG_FILTER_AG_TO	AG number to
MSG_FILTER_AGSUB_FROM	AG sub-number from
MSG_FILTER_AGSUB_TO	AG sub-number to

MSG_FILTER_TEXT	Alarm texts
MSG_FILTER_PVALUE	Process values
MSG_FILTER_COUNTER_FROM	Internal alarm counter from
MSG_FILTER_COUNTER_TO	Internal alarm counter to
MSG_FILTER_PROCESSMSG	Process alarms (reserved)
MSG_FILTER_SYSMMSG	System alarms (reserved)
MSG_FILTER_BEDMSG	Operator input alarms (reserved)
MSG_FILTER_VISIBLEONLY	Display visible alarms (is not included in the filter condition)
MSG_FILTER_HIDDENONLY	Display hidden alarms (is not included in the filter condition)
MSG_FILTER_NODYNAMIC	No dynamic text selection (is not included in filter condition)
MSG_FILTER_DATE	Date from to
MSG_FILTER_TIME	Time from to
MSG_FILTER_NR	Alarm number from to

The individual filter conditions can be ORed.

st

st[0] Date and time of the start point, st[1] Date and time of the end point

You must assign st, if you use the filter criteria MSG_FILTER_DATE, MSG_FILTER_DATE_FROM, MSG_FILTER_DATE_TO, MSG_FILTER_TIME, MSG_FILTER_TIME_FROM, or MSG_FILTER_TIME_To.

If the current timer for transferring a SYSTEMTIME parameter is required, the GetLocalTime must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgNrHigh Number of the last alarm

You must assign dwMsgNrLow and dwMsgNrHigh, if you use the MSG_FILTER_NR, MSG_FILTER_NR_FROM, or MSG_FILTER_NR_TO filter criteria.

vMsgClasses

Alarm classes bit-coded.

0x0001 = class 1, 0x0002 = class 2, 0x0004 = class 3, ... 0x0040= class 7

0x0204 = classes 3 and 10

You must assign dwMsgClass and dwMsgType, if you use the filter criterion MSG_FILTER_CLASS.

dwMsgState

Alarm status bit-coded.

Assign this field, if you use the filter criterion MSG_FILTER_STATE.

The following status values are checked using MSRTCheckWinFilter:

MSG_STATE_COME	Incoming alarm
MSG_STATE_GO	Outgoing alarm
MSG_STATE_QUIT	Alarm acknowledged
MSG_STATE_QUIT_SYSTEM	Alarm acknowledged by system
MSG_STATE_QUIT_EMERGENCY	Emergency acknowledgment

You must filter other status values yourself, e.g., MSG_STATE_HIDE or MSG_STATE_UNHIDE, since these are not taken into consideration in the filter condition. To filter these status values, use e.g. MSG_FILTER_HIDDENONLY.

wAGNr

wAGNr [0] First PLC number, wAGNr [1] last PLC number.

You must assign wAGNr, if you use the filter criteria MSG_FILTER_AG_FROM or MSG_FILTER_AG_TO.

wAGSubNr

wAGSubNr [0] First PLC number, wAGSubNr [1] last PLC number.

You must assign wAGSubNr, if you use the filter criteria MSG_FILTER_AGSUB_FROM or MSG_FILTER_AGSUB_TO.

dwArchivMode

ID for log or report. The parameter must be assigned 0.

szTB

Texts of the text blocks

You must assign szTB, if you use the filter criterion MSG_FILTER_TEXT.

dwTB

Active text blocks, bit coded

You must assign dwTB, if you use the filter criterion MSG_FILTER_TEXT

dPValue

Process values from - to

You must assign dPValue, if you use the filter criterion MSG_FILTER_PVALUE.

dwPValue

Process values, bit-coded

You must assign dwPValue, if you use the filter criterion MSG_FILTER_PVALUE.

ullMgCounter

Internal alarm counter

3.10 Alarm functions (RT Professional)

You must assign `dwMsgCounter`, if you use the filter criterion `MSG_FILTER_COUNTER_....`. When used in the script, `ullMsgCounter` is not type `ULONGLONG`, but instead a union `MSGULONGLONG` (see `CCMsRtCliPlus.h`).

dwQuickSelect

The parameter is reserved for future development and must be preset to 0.

dwPriority

Priority of the alarm

szInstance

Instance name. If `szInstance` is not provided (empty string), a normal alarm is created.

Required files

`CCMsRtCliPlus.h`

API functions

<code>MSRTCheckWinFilterPlus</code> (Page 2437)	Check alarm filter conditions
<code>MSRTEnumProtDataPlus</code> (Page 2401)	List report data
<code>MSRTGetFilterDataPlus</code> (Page 2435)	Query alarm filter conditions
<code>MSRTSetMsgFilterPlus</code> (Page 2439)	Set alarm filter
<code>MSRTSetMsgWinFilterPlus</code> (Page 2440)	Set alarm filter for an alarm window
<code>MSRTGetLastMsgWithCommentPlus</code> (Page 2417)	Query last archived alarm
<code>MSRTStartMsgServicePlus</code> (Page 2391)	Start service

See also

`MSRTGetLastMsgWithCommentPlus` (Page 2417)

`MSRTSetMsgWinFilterPlus` (Page 2440)

`MSRTStartMsgServicePlus` (Page 2391)

`MSRTSetMsgFilterPlus` (Page 2439)

`MSRTCheckWinFilterPlus` (Page 2437)

`MSRTGetFilterDataPlus` (Page 2435)

`MSRTEnumProtDataPlus` (Page 2401)

Overview of structures (Page 2347)

3.10.2.4 MSG_RTDATA_INSTANCE_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {
    DWORD          dwMsgState;
    DWORD          dwMsgNrLow;
    DWORD          dwMsgNrHigh;
    SYSTEMTIME     stMsgTime;
    DWORD          dwTimeDiff;
    ULONGLONG     ullCounter;
    DWORD          dwFlags;
    WORD           wPValueUsed;
    WORD           wTextValueUsed;
    double         dPValue[MSG_MAX_PVALUE];
    MSG_TEXTVAL256_STRUCT_PLUS mtTextValue[MSG_MAX_PVALUE];
    TCHAR          szInstance[MSG_MAX_INSTANCE+1];
}
MSG_RTDATA_INSTANCE_STRUCT_PLUS;
```

Members

dwMsgState

Alarm status in compliance with the constants in the "m_global.h" file:

The statuses listed will only be used internally by the system and can only be set by the API using MSRTCreateMsgInstanceWithCommentPlus.

MSG_STATE_COME	Incoming alarm
MSG_STATE_GO	Outgoing alarm
MSG_STATE_QUIT	Alarm acknowledged
MSG_STATE_LOCK	Alarm locked
MSG_STATE_UNLOCK	Alarm released
MSG_STATE_HIDE	Alarm hidden
MSG_STATE_UNHIDE	Alarm displayed
MSG_STATE_HIDE_MANUAL	Alarm hidden manually

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

stMsgTime

Date and time for coming-in or going-out of an alarm.

3.10 Alarm functions (RT Professional)

dwTimeDiff

Duration coming in or going out, telegram time in seconds.

ullCounter

Internal alarm counter.

dwFlags

Flags in compliance with the constants in the "m_global.h" file:

Flags for alarms in the alarm view		
MSG_RT_FLAG_COME	0x00000001	Incoming alarm
MSG_RT_FLAG_GO	0x00000002	Outgoing alarm
MSG_RT_FLAG_QUIT_COME	0x00000004	Incoming acknowledgment
MSG_RT_FLAG_QUIT_GO	0x00000008	Outgoing acknowledgment
MSG_RT_FLAG_NOQUIT	0x00000010	No acknowledgment
MSG_RT_FLAG_QUIT_NOGO	0x00000020	TM II acknowledgment
MSG_RT_FLAG_FLASH	0x00000040	Flashing
MSG_RT_FLAG_FIRSTFLASH	0x00000080	Only first alarm is permitted to flash
MSG_RT_FLAG_DELETE	0x00010000	Alarm can be deleted
MSG_RT_FLAG_SUMTIME	0x00020000	Daylight saving time flag
MSG_RT_FLAG_SQUIT	0x00040000	Alarm has group acknowledgment
MSG_RT_FLAG_QUITNR	0x00080000	Alarm number acknowledgment
MSG_RT_FLAG_MSGLIST	0x00100000	Alarm from the alarm list
MSG_RT_FLAG_COMMENT	0x00200000	Alarm has comments
MSG_RT_FLAG_ARCHIV	0x00400000	Alarm is logged
MSG_RT_FLAG_PROTOCOL	0x00800000	Alarm is reported
MSG_RT_FLAG_INSTANCE	0x01000000	Flag for instance alarm
MSG_RT_FLAG_TIMEINVALID	0x02000000	Flag for invalid date/time stamp
MSG_RT_FLAG_LOCKGROUP	0x04000000	Flag for group lock
MSG_RT_FLAG_INSTANCE10	0x08000000	Bit for new instance alarm with process value 10.
MSG_RT_FLAG_TEXTREF1	0x10000000	Bit for text reference 1.
MSG_RT_FLAG_TEXTREF2	0x20000000	Bit for text reference 2.
MSG_RT_FLAG_QUITCOUNTER	0x40000000	Bit for acknowledgment via counter
MSG_RT_FLAG_QUITGROUP	0x80000000	Bit for acknowledgment via counter

Flags for alarms in the log		
MSG_FLAG_SUMTIME	0x00000001	Daylight saving time active
MSG_FLAG_COMMENT	0x00000002	Alarm has comments
MSG_FLAG_ARCHIV	0x00000004	Log
MSG_FLAG_PROTOCOL	0x00000008	Report
MSG_FLAG_TEXTVALUES	0x00000010	Alarm has associated text values
MSG_FLAG_TIMEINVALID	0x00000020	Bit for invalid date/time stamp
MSG_FLAG_INSTANCE	0x00000040	Identifier for instance alarms

Flags for alarms in the log		
MSG_FLAG_INSTANCE10	0x08000000	Instance alarm with process value 10, available from V5.0 SP2
MSG_FLAG_TEXTREF1	0x10000000	Text reference 1
MSG_FLAG_TEXTREF2	0x20000000	Text reference 2

With the MSRTCreateMsgInstanceWithCommentPlus function, flags cannot be sent to the alarm server. Flags are ignored.

wPValueUsed

Process values used in bit coded layout.

Each bit should only be set in one of the two members wPValueUsed or wTextValueUsed, because only number or text is technically possible as an associated value. In the case of erroneous double entry the dPValue has priority.

wTextValueUsed

Text values used, bit-coded.

Each bit should only be set in one of the two members wPValueUsed or wTextValueUsed, because only number or text is technically possible as an associated value.

In the case of erroneous double entry the dPValue has priority.

dPValue

Process values.

mtTextValue

Structure of the MSG_TEXTVAL256_STRUCT_PLUS (Page 2388) type with the associated text values.

szInstance

Instance name. If szInstance is not provided (empty string), a normal alarm is created

Required files

CCMSRTCLIPlus.h

API functions

MSRTCreateMsgInstancePlus (Page 2407)	Create alarm
MSG_SERVICE_NOTIFY_PROC	Send and receive service (callback)

See also

MSRTCreateMsgInstancePlus (Page 2407)

MSG_SERVICE_NOTIFY_PROCLPlus (Page 2397)

MSG_TEXTVAL256_STRUCT_PLUS (Page 2388)

Overview of structures (Page 2347)

3.10.2.5 MSG_CLASS_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {
    WCHAR          szName[MSG_MAX_TEXTLEN+1];
    DWORD          dwName;
    DWORD          dwClassID;
    COLORREF       crStateCome[2];
    COLORREF       crStateGo[2];
    COLORREF       crStateQuit[2];
    WORD           wQuitType;
    WORD           wHornQuit;
    DWORD          dwHornVar;
    WCHAR          szState[MSG_MAX_STATE][MSG_MAX_TEXTLEN+1];
    DWORD          dwState[MSG_MAX_STATE];
    WCHAR          szStateQuit[MSG_MAX_TEXTLEN+1];
    DWORD          dwStateQuit;
    DWORD          dwClassTyp;
    DWORD          dwCreatorID;
}
MSG_CLASS_STRUCT_PLUS;
```

Members

szName

The class name to be specified is dependent on the selected data language!

dwName

If the class name is included in the project texts, the ID can be specified here.

dwClassID

ID of the alarm class (1..18(MSG_MAX_CLASS))

crStateCome

Color for the "incoming" status

crStateGo

Color for the "outgoing" status

crStateQuit

Color for the "acknowledged" status

wQuitType

The acknowledgment philosophy is defined by means of the following constants from the "m_global.h" file:

MSG_QUIT_QUIT_COME	Incoming acknowledgment
MSG_QUIT_QUIT_GO	Outgoing acknowledgment
MSG_QUIT_NOQUIT	No acknowledgment
MSG_QUIT_QUIT_NOGO	Acknowledgment (without outgoing alarm)
MSG_QUIT_FLASH	Flashing
MSG_QUIT_FIRSTFLASH	Only first alarm is permitted to flash
MSG_QUIT_NOLIST	Do not receive alarms in alarm list

You can logically combine the constants of the state machines bit-by-bit with "OR", provided they are not mutually contradictory.

wHornQuit

Acknowledgment of central horn

dwHornVar

Tag of central horn

szState

Text for the "incoming", "outgoing", and "incoming and outgoing" statuses.

dwState

TextID

szStateQuit

Text for the "acknowledged" status.

dwStateQuit

TextID

dwClassTyp

The parameter is reserved for future development and must be preset to 0.

dwCreatorID

CreatorID

Comment

In contrast to the MSG_CLASS_STRUCT_EX structure, MSG_CLASS_STRUCTPlus does not have a creator identification.

Required files

CCMSRTCLIPus.h

API functions

MSRTGetClassInfoPlus (Page 2415)	Query alarm class information

See also

MSRTGetClassInfoPlus (Page 2415)

Overview of structures (Page 2347)

3.10.2.6 MSG_CSDATA_STRUCT_PLUS (RT Professional)**Declaration**

```
typedef struct {
    DWORD    dwMsgNrLow;
    DWORD    dwMsgNrHigh;
    DWORD    dwStatus;
    WORD     wClass;
    WORD     wTyp;
    DWORD    dwTextID[MSG_MAX_TB];
    DWORD    dwQuitVar;
    WORD     wQuitBit;
    DWORD    dwStateVar;
    WORD     wStateBit;
    DWORD    dwMsgVar;
    WORD     wMsgBit;
    WORD     wAGNr;
    WORD     wAGSubNr;
    DWORD    dwGroupID;
    DWORD    dwPriority;
    DWORD    dwHidingMask;
}
MSG_CSDATA_STRUCT_PLUS;
```

Members**dwMsgNrLow**

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

dwStatus

The individual alarm parameters are defined by means of the following constants from the file "m_global.h":

MSG_PARAMS_SQUIT	Single acknowledgment
MSG_PARAMS_HORN	Activates horn
MSG_PARAMS_ARCHIV	Will be archived
MSG_PARAMS_PROTOCOL	Will be logged
MSG_PARAMS_VARFLANK	Will be created on negative edge
MSG_PARAMS_INFOTEXT	Has Info text.
MSG_PARAMS_LOOPINALARM	Has loop-in-alarm
MSG_PARAMS_NORMDLL	Has normalization DLL

You can OR these constants, provided they do not contradict each other.

wClass

Alarm class

wTyp

Alarm type

dwTextID

Reference to the text blocks

dwQuitVar

Acknowledgment tag. Use dwID from DM_VARKEY (Page 1590).

wQuitBit

Bit of acknowledgment tag

dwStateVar

Alarm status as tag. Use dwID from DM_VARKEY (Page 1590)

wStateBit

wStateBit is used to specify which bits of dwStateVar define the alarm status.

If wStateBit = 1 and dwStateVar of data type "8 Bit unsigned", then Bit 1 of dwStateVar specifies the incoming/outgoing status, and bit 5 specifies the need for acknowledgment.

If wStateBit = 2 and dwStateVar is a word tag, then Bit 2 of dwStateVar specifies the incoming/outgoing status, and bit 10 specifies the need for acknowledgment.

dwMsgVar

Alarm tag. Use dwID from DM_VARKEY (Page 1590).

wMsgBit

Alarm tag bit

- wAGNr**
PLC number
- wAGSubNr**
PLC sub-number
- dwGroupID**
ID for the alarm group (internal)
- dwPriority**
Priority of the alarm
- dwHidingMask**
Display suppression mask

Comment

By means of this structure, your application can use API functions to determine the configuration data of a single message.

Required files

CCMSRTCLIPlus.h

API functions

MSRTGetMsgCSDDataPlus (Page 2420)	Query alarm configuration data

See also

- MSRTGetMsgCSDDataPlus (Page 2420)
- DM_VARKEY (Page 1590)
- Overview of structures (Page 2347)

3.10.2.7 MSG_RTGROUPENUM_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {  
    BOOL    fIDUsed;  
    DWORD   dwID;  
    TCHAR   szName[MSG_MAX_TEXTLEN+1];  
    DWORD   dwMsgCount;  
    DWORD   dwMsg[MSG_MAX_GROUPITEMS];  
}  
MSG_RTGROUPENUM_STRUCT_PLUS;
```

Members

fIDUsed

Specifies whether the Group ID (fIDUsed = TRUE) or name (fIDUsed = FALSE) is used to specify the alarm group.

dwID

Group ID

szName

Group name

dwMsgCount

Number of single alarms in dwMsg

dwMsg

Locked single alarms

Required files

CCMSRTCLIPlus.h

API functions

MSRTEnumGroupMsgPlus (Page 2431)	List alarm group single alarms
MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)	Callback function

See also

MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)

MSRTEnumGroupMsgPlus (Page 2431)

Overview of structures (Page 2347)

3.10.2.8 MSG_RTGROUPSET_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {  
    BOOL        fIDUsed;  
    DWORD       dwID;  
    WCHAR       szName[MSG_MAX_TEXTLEN+1];  
    SYSTEMTIME  stTime;  
    DWORD       dwData;  
}  
MSG_RTGROUPSET_STRUCT_PLUS;
```

Members

fIDUsed

Specifies whether the Group ID (fIDUsed = TRUE) or name (fIDUsed = FALSE) is used to specify the alarm group.

dwID

Group ID (for alarm classes, alarm types)

szName

Group name

stTime

Time of occurrence

dwData

If the structure is used within the scope of MSRTLockGroupPlus, dwData = 1 signifies locking and dwData = 0 releasing

Required files

CCMSRTCLIPlus.h

API functions

MSRTLGroupPlus (Page 2433)	Lock alarm group
MSRTEnumGroupMsgPlus (Page 2431)	List single alarms of an alarm group
MSRTQuitGroupPlus (Page 2434)	Acknowledge alarm group

See also

MSRTEnumGroupMsgPlus (Page 2431)

MSRTLGroupPlus (Page 2433)

MSRTQuitGroupPlus (Page 2434)

Overview of structures (Page 2347)

3.10.2.9 MSG_COMMENT_INSTANCE_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {
    SYSTEMTIME    stTime;
    DWORD         dwMsgNrLow;
    DWORD         dwMsgNrHigh;
    WCHAR         szText[MSG_MAX_TB_CONTENT+1];
    WCHAR         szUser[MSG_MAX_USERNAME+1];
    WCHAR         szComputerName[MAX_COMPUTERNAME_LENGTH+1];
    WCHAR         szApplicationName[MSG_MAX_APPLNAME+1];
    WCHAR         szInstance[MSG_MAX_INSTANCE+1];
}
MSG_COMMENT_INSTANCE_STRUCT_PLUS;
```

Members

stTime

Date/time

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

szText

Comment

szUser

User name

szComputerName

Computer name

szApplicationName

Application name

szInstance

Instance name

Comment

If a current time is needed for the passing of a SYSTEMTIME parameter, the GetLocalTime function must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

Required files

CCMSRTCLIPlus.h

API functions

MSRTGetCommentInstancePlus (Page 2443)	Query comment text
MSRTSetCommentInstancePlus (Page 2445)	Specify comment text

See also

MSRTSetCommentInstancePlus (Page 2445)

MSRTGetCommentInstancePlus (Page 2443)

Overview of structures (Page 2347)

3.10.2.10 MSG_INFOTEXT_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {
    DWORD    dwMsgNrLow;
    DWORD    dwMsgNrHigh;
    WCHAR    szText[MSG_MAX_TB_CONTENT+1];
}
MSG_INFOTEXT_STRUCT_PLUS;
```

Members

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgNrHigh Number of the last alarm

szText

Info text

Required files

CCMSRTCLIPlus.h

API functions

MSRTGetInfotextPlus (Page 2446)	Query info text
MSRTSetInfotextPlus (Page 2448)	Specify info text

See also

MSRTGetInfotextPlus (Page 2446)

MSRTSetInfotextPlus (Page 2448)

Overview of structures (Page 2347)

3.10.2.11 MSG_BACKUP_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {
    SYSTEMTIME    stFrom;
    SYSTEMTIME    stTo;
    WCHAR         szArchivFile[MAX_PATH+1];
    WCHAR         szArchivTextFile[MAX_PATH+1];
    WCHAR         szCommentFile[MAX_PATH+1];
    WCHAR         szInstanceFile[MAX_PATH+1];
    WCHAR         szComment[MAX_PATH+1];
    DWORD         dwFlags;
    DWORD         dwFormat;
}
MSG_BACKUP_STRUCTPlus;
```

Members

stFrom

Date from

stTo

Date to

szArchivFile

File name and path of the export file for log

szArchivTextFile

File name and path of the export file for log texts

szCommentFile

File name and path of the export file for comments

szInstanceFile

File name and path of the export file for instances

szComment

User-defined comments

dwFlags

MSG_DB_FLAGS_OVERWRITE	0x0001	Overwrite existing data records
MSG_DB_FLAGS_ADD	0x0002	Add only
MSG_DB_FLAGS_DELETE	0x0004	Delete source data records

dwFormat

You can use the following constants from the "m_global.h" file for formatting the export file:

MSG_DB_FMT_CSV	CSV file format
MSG_DB_FMT_DBASE3	DBASE3 file format
MSG_DB_FMT_WATCOM	Sybase format

Comment

If the current timer for transferring a SYSTEMTIME parameter is required, the GetLocalTime must be used and not GetSystemTime. There is generally a considerable time difference between these two functions.

Required files

CCMSRTCLIPlus.h

API functions

MSRTEnumBackupListPlus (Page 2449)	List entries of an export file
MSRTGetBackupSize	Determine size of export file
MSRTRestore	Import alarms

See also

MSRTEnumBackupListPlus (Page 2449)

MSRTExportPlus (Page 2451)

Overview of structures (Page 2347)

3.10.2.12 MSG_RTLOCK_STRUCT_PLUS (RT Professional)**Declaration**

```
typedef struct {
    WCHAR    szName[MSG_MAX_TEXTLEN+1];
    WCHAR    szParent[MSG_MAX_TEXTLEN+1];
    DWORD    dwLockID;
    BOOL     fLock;
    DWORD    dwMsgNum;
    DWORD    dwMsgLock[MSG_MAX_LOCKITEMS];
}
MSG_RTLOCK_STRUCT_PLUS;
```

Members**szName**

Lock name

szParent

Parent group

dwLockID

Lock ID

fLock

Lock and release

dwMsgNum

Number of locked alarms

dwMsgLock

Locked single alarms

Required files

CCMSRTCLIPlus.h

API functions

MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)	Callback function

See also

MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)

Overview of structures (Page 2347)

3.10.2.13 MSG_TEXTVAL256_STRUCT_PLUS (RT Professional)

Declaration

```
typedef struct {  
    WCHAR szText[MSG_MAX_TEXTVAL256+1];  
}  
MSG_TEXTVAL256_STRUCT_PLUS;
```

Comment

The structure is used within the following structures:

Members

szText
Text

Required files

CCMSRTCLIPlus.h

API functions

MSRTCreateMsgInstanceWithCommentPlus (Page 2406)	Create alarm
MSRTGetLastMsgWithCommentPlus (Page 2417)	Query last pending alarm
MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)	Send and receive service (callback)

See also

MSG_RTDATA_INSTANCE_STRUCT_PLUS (Page 2371)

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363)

MSG_RTCREATE_STRUCT_PLUS (Page 2361)

Overview of structures (Page 2347)

3.10.3 General functions (RT Professional)**3.10.3.1 MSRTEnumArchivDataPlus (RT Professional)****Description**

Enumerates all alarms in the specified log. Its callback function MSG_SERVICE_NOTIFY_PROCPlus is called using the MSG_NOTIFY_ARCHIVENUM notification type. lpbyData refers to data of the MSG_RTARCHIV_STRUCTPlus structure.

Enumeration functions work asynchronously. As a result, you must synchronize the calls using semaphores.

Declaration

```

BOOL MSRTEnumArchivDataPlus(
    DWORD          dwServiceID,
    BOOL           fArchiv,
    DWORD          dwMaxRecords
    DWORD          dwParams
    LPDWORD        lpdwTAID
    LPCMN_ERROR    lpError );

```

Parameters**dwServiceID**

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

3.10 Alarm functions (RT Professional)

fArchiv

Designates the log type.

TRUE	Sequential log

FALSE	Circular log

dwMaxRecords

Maximum number of the data records to be listed. If the dwMaxRecords is exceeded numbering will stop. If dwMaxRecords = 0xFFFFFFFF, all alarms will be listed.

dwParams

Parameters for the sort order:

MSG_ARCHIV_ENUM_DESC	Sort in descending order

lpdwTAID

Pointer to TransactionID for asynchronous processing.

lpError

Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Alarms listed.

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Parameter invalid

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)	Callback function
MSRTStartMsgServicePlus (Page 2391)	Start service

See also

MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)

MSRTStartMsgServicePlus (Page 2391)

Overview of the functions (Page 2346)

3.10.3.2 MSRTStartMsgServicePlus (RT Professional)

Description

Starts a service for sending and receiving of notifications in the alarm system.

Declaration

```

BOOL MSRTStartMsgServicePlus (
    LPDWORD                lpdwServiceID,
    LPCWSTR                lpszServer,
    DWORD                  dwFlags,
    DWORD                  dwFilterCount
    LPMSG_FILTER_STRUCT_PLUS*
    DWORD                  dwNotifyMask,
    MSG_SERVICE_NOTIFY_PROC_PLUS
    LPVOID                 lpfnNotifyProcPlus
    MSG_TAID_COMPLETION_PROC_PLUS
    LPVOID                 lpvUserNotify,
    LPCMN_ERRORW           lpfnTAIDProcPlus
    LPVOID                 lpvUserTAID
    LPCMN_ERRORW           lpError );

```

Parameters

dwServiceID

Contains the identification number of the send and receive service for a function, once it has been successfully called; this service is required by many functions.

lpszServer

Server prefix

dwFlags

The parameter is reserved for future development and must be preset to 0.

dwFilterCount

Number of message filters in the pointer list lppMsgFilterPlus. If 0 is specified here, no message filter from lppMsgFilterPlus are applied.

lppMsgFilter

Pointer to your list of points to message filter structures MSG_FILTER_STRUCT_PLUS (Page 2367).

The list must contain at least as many entries as specified in dwFilterCount. If dwFilterCount is equal to 0, lppMsgFilterPlus should be set to ZERO.

dwNotifyMask

Designates the kind of notification, in accordance with the definition of constants in the "msrtapi.h" file:

MSG_NOTIFY_MASK_MSGLIST	Notification regarding alarm
MSG_NOTIFY_MASK_LOCK	Notification regarding locks
MSG_NOTIFY_MASK_ARCHIV	Notification regarding logged alarms
MSG_NOTIFY_MASK_ALL	All kinds of notifications

lpfnNotifyProcPlus

Pointer to MSG_SERVICE_NOTIFY_PROCPPlus type callback function for passing the notification by the service.

With lpfnNotifyProcPlus = NULL the service will not return any notifications.

Note

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

lpvUserNotify

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpfnTAIDProcPlus

Callback function for asynchronous processing of feedback.

lpvUserTAID

Pointer to application-specific data. This pointer is not evaluated by the function but provided again in the callback function.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Service started.

FALSE

Error

Comment

Your application can install a maximum of 16 services. In total up to 128 services can be installed.

Error messages

MSG_ERR_RT_NOTCONNECTED	Connection to the alarm system not made
MSG_ERR_RT_NOINIT	Alarm system not initialized
MSG_ERR_RT_SERVICEMAX	Maximum number of services reached
MSG_ERR_NOSERVER	No default server specified and no local server available

Required files

CMSRTCLIPlus.h
CMSRTCLIPlus.lib
CMSRTCLIPlus.dll

Related functions

MSRTStopMsgServicePlus (Page 2394)	Stop services
MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)	Callback function

See also

MSRTCreateMsgInstanceWithCommentPlus (Page 2406)
MSRTCreateMsgInstancePlus (Page 2407)
MSRTExportPlus (Page 2451)
MSRTEnumBackupListPlus (Page 2449)
MSRTGetInfotextPlus (Page 2446)
MSRTSetInfotextPlus (Page 2448)
MSRTGetClassInfoPlus (Page 2415)
MSRTSetMsgFilterPlus (Page 2439)

MSG_FILTER_STRUCT_PLUS (Page 2367)
MSRTGetMsgCSDDataPlus (Page 2420)
MSRTCheckWinFilterPlus (Page 2437)
MSRTGetMsgPriorityPlus (Page 2421)
MSRTGetMsgQuitPlus (Page 2423)
MSRTGetMsgTextPlus (Page 2424)
MSRTGetSelectedMsgPlus (Page 2426)
MSRTResetMsgPlus (Page 2428)
MSRTLoopInAlarmPlus (Page 2430)
MSRTGetFilterDataPlus (Page 2435)
MSRTEnumArchivDataPlus (Page 2387)
MSRTStopMsgServicePlus (Page 2394)
MSRTSetCommentInstancePlus (Page 2445)
MSG_SERVICE_NOTIFY_PROCPlus (Page 2397)
MSRTQuitHornPlus (Page 2396)
MSRTGetCommentInstancePlus (Page 2443)
MSRTEnumGroupMsgPlus (Page 2431)
MSRTLckGroupPlus (Page 2433)
MSRTGetMsgActualPlus (Page 2418)
Overview of the functions (Page 2346)

3.10.3.3 MSRTStopMsgServicePlus (RT Professional)

Description

Stops the service for sending and receiving messages

Declaration

```
BOOL MSRTStopMsgServicePlus (  
    DWORD          dwServiceID,  
    LPCMN_ERRORW   lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

LPCMN_ERRORW

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Service ended.

FALSE

Error

Comment**Note**

The call should not be used in an application destructor (EXE, DLL, OCX, etc.), because a Microsoft-specific mechanism can sometimes lead to a call hang and therefore a program hang too.

Error messages

MSG_ERR_RT_NOTCONNECTED	Connection to the alarm system not made
MSG_ERR_RT_NOINIT	Alarm system not initialized

Required files

CMSRTCLIPlus.h

CMSRTCLIPlus.lib

CMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTEnumBackupListPlus (Page 2449)

MSRTStartMsgServicePlus (Page 2389)

Overview of the functions (Page 2346)

3.10.3.4 MSRTQuitHornPlus (RT Professional)

Description

Class-specific central horn acknowledgement.

Declaration

```
BOOL MSRTQuitHornPlus(  
    DWORD          dwServiceID,  
    LPDWORD        lpdwTAID,  
    LPCMN_ERRORW   lpError );
```

Parameters

- dwServiceID**
Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.
- lpdwTAID**
Pointer to the TransactionID for asynchronous processing.
- LPCMN_ERRORW**
Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information into this structure.

Return value

- TRUE**
Alarm acknowledged.
- FALSE**
Error

Error messages

MSG_ERR_API_PARAM	Parameter invalid

Required files

- CCMSRTCLIPlus.h
- CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

Overview of the functions (Page 2346)

3.10.3.5 MSG_SERVICE_NOTIFY_PROCPlus (RT Professional)

Description

Your callback function for enumeration functions. The enumeration functions work asynchronously. As a result, you must synchronize the calls using semaphores.

Declaration

```

BOOL ( * MSG_SERVICE_NOTIFY_PROC) (
    DWORD      dwNotify,
    LPBYTE     lpbyData,
    DWORD      dwItems,
    LPVOID     lpvUserNotify);

```

Parameters

dwNotify

The notification type is specified by the constants in the "MSRTAPI.h" file:

General notifications		lpbyData points to structure
MSG_NOTIFY_CONNECT	Connection established	-
MSG_NOTIFY_DISCONNECT	Connection terminated	-
MSG_NOTIFY_RESTART	New alarm system runtime data	-
MSG_NOTIFY_SHUTDOWN	The alarm system will be stopped	-
MSG_NOTIFY_MSGENUM	Enumeration of the alarm list	MSG_RTDATA_INSTANCE_STRUCTPlus
MSG_NOTIFY_ARCHIVENUM	Enumeration of log alarm(s)	MSG_RTDATA_STRUCTPlus (in Blocks)
MSG_NOTIFY_ARCHIVENUMINST	Enumeration of log alarm(s) with instances	MSG_RTDATA_INSTANCE_STRUCTPlus (in Blocks)
MSG_NOTIFY_LOCKENUM	Enumeration of locks	MSG_LOCK_STRUCTPlus (in Blocks)
MSG_NOTIFY_WINENUM	Alarm windows enumeration	MSG_WIN_STRUCTPlus (in Blocks)
MSG_NOTIFY_BACKUPENUM	Enumeration of export list	MSG_BACKUP_STRUCTPlus (in Blocks)

3.10 Alarm functions (RT Professional)

General notifications		IpbyData points to structure
MSG_NOTIFY_ARCDAYENUM	Enumeration of day list	MSG_ARCHIVDAY_STRUCTPlus
MSG_NOTIFY_ENUMMAX	Enumeration was stopped at maximum amount	-
MSG_NOTIFY_LANGUAGESET	Language switched	-

Single alarms		IpbyData points to structure
MSG_NOTIFY_MSGADD	New alarm added	MSG_RTDATA_INSTANCE_STRUCTPlus (individual)
MSG_NOTIFY_MSGDELETE	Alarm deleted	MSG_RTDATA_INSTANCE_STRUCTPlus (individual)
MSG_NOTIFY_MSGMODIFY	Alarm changed	MSG_RTDATA_INSTANCE_STRUCTPlus (individual)
MSG_NOTIFY_MSGCOUNT	Current number of alarms	-

Alarm groups		IpbyData points to structure
MSG_NOTIFY_GROUPENUM	Enumeration of single alarms in an alarm group	MSG_RTGROUPENUM_STRUCTPlus (in Blocks)

Locks of alarms		IpbyData points to structure
MSG_NOTIFY_LOCKMODIFY	Locks set, changed, deleted	MSG_RTGROUPENUM_STRUCTPlus

Log alarms		IpbyData points to structure
MSG_NOTIFY_ARCHIVADD	Log alarm(s) added	MSG_RTDATA_INSTANCE_STRUCTPlus (individual)
MSG_NOTIFY_ARCHIVDELETE	Log alarm(s) deleted	MSG_RTDATA_INSTANCE_STRUCTPlus (individual)

IpbyData

IpbyData contains a pointer to data depending on the notification type.

General notifications	No data
Single alarms	Runtime data, stored in structures of type MSG_RTDATA_INSTANCE_STRUCT_PLUS (Page 2371)
Alarm groups	Single alarm data in structures of type MSG_RTGROUPENUM_STRUCT_PLUS (Page 2379)
Locks	Lock data in structures of type MSG_RTLOCK_STRUCT_PLUS (Page 2385)
Log alarms	Log data in structures of type MSG_RTDATA_INSTANCE_STRUCT_PLUS (Page 2371)

dwItems

Contains the number of objects returned in lpbyData. If dwItems = 0, then lpbyData = NULL. Notify will be ended and the semaphore will be released.

lpvUserNotify

Pointer to application-specific data. This pointer is made available again in the callback function.

Return value**TRUE**

Notification was recognized and processed.

FALSE

Error

Comment**Note**

Only data should be copied here if possible. The following types of function calls within the callback can lead to deadlocks or a stack overflow:

- Functions in which a message loop is accessed, for example: GetMessage
- API functions from the same DLL
- Enumerations that call other enumerations

If a program declares a Notify routine, it must empty its message queue regularly. Unread messages could block WinCC notifications or finally block WinCC altogether.

In rare cases, the Notify may already be delivered before the function call has returned.

If a callback function is programmed as a project function in script, then the asynchronous call also has the result that the function is not fully available in the script context, and in particular the GetTag...- and SetTag-calls will not work. Where appropriate, processing must be performed in other actions, that are only triggered by the callback.

Required files

CCMSRTCLIPlus.h

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_RTDATA_INSTANCE_STRUCT_PLUS (Page 2371)
MSG_RTGROUPENUM_STRUCT_PLUS (Page 2379)
MSG_RTLOCK_STRUCT_PLUS (Page 2385)
MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363)
MSRTStartMsgServicePlus (Page 2389)
MSG_TEXTVAL256_STRUCT_PLUS (Page 2386)
MSRTEnumGroupMsgPlus (Page 2431)
Overview of the functions (Page 2346)

3.10.3.6 MSRTWebClientPlus (RT Professional)

Description

Initializes the DLL for a WebClient

Declaration

```
VOID WINAPI MSRTWebClientPlus(  
    LPCWSTR      pszProjectPath,  
    DWORD        dwDataLocale);
```

Parameters

pszProjectPath
Path to the Web Project

dwDataLocale

Return value

3.10.4 Functions for protocol processing (RT Professional)

3.10.4.1 MSRTActivateMProtPlus (RT Professional)

Description

Pause or continue report service for output of an alarm sequence report.

Declaration

```
BOOL MSRTActivateMProtPlus(  
    BOOL      fActive,  
    LPCMN_ERROR lpError );
```

Parameters

fActive

Establishes how the report service should be handled:

TRUE	Continue report service
FALSE	Pause report service

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Function successfully completed.

FALSE

Error

Error messages

MSG_ERR_PROT_ACTIVE	Report service already active

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

3.10.4.2 MSRTEnumProtDataPlus (RT Professional)

Description

Enumeration of report data. The individual report data is separated by commas. The function is no longer supported.

Enumeration functions work asynchronously. As a result, you must synchronize the calls using semaphores.

Declaration

```
BOOL MSRTEnumProtDataPlus (
    DWORD                dwProtID,
    MSG_PROT_NOTIFY_PROC lpfnEnum,
    LPMSG_FILTER_STRUCT  lpMsgFilter,
    LPVOID               lpvUser,
    LPCMN_ERROR          lpError );
```

Parameters

dwProtService

Service ID for report data exchange.

lpfnEnum

Your callback function that accepts report data.

lpMsgFilter

Pointer to the data in the alarm filter in the MSG_FILTER_STRUCT_PLUS (Page 2367) structure.

lpvUser

Pointer to application-specific data. This pointer is not evaluated by the function but is made available again in the callback function.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information to this structure.

Return value

TRUE

Report data listed.

FALSE

Error

Comment

WARNING

Your application can use this function to acknowledge alarms, without ensuring that the equipment operator has been made aware of the alarms. This circumstance can result in danger to life and limb or property!

Error messages

MSG_ERR_PROT_ID	Invalid report identifier

Required files

CMSRTCLIPlus.h
CMSRTCLIPlus.lib
CMSRTCLIPlus.dll

Related functions

MSG_PROT_NOTIFY_PROC	Callback function

See also

MSG_FILTER_STRUCT_PLUS (Page 2367)

3.10.4.3 MSRTPrintMProtPlus (RT Professional)

Description

Prints currently pending alarms of the alarm sequence report, even when a page is not yet full.

Declaration

```
BOOL MSRTPrintMProtPlus(  
    DWORD*          pdwLines,  
    LPCMN_ERRORW    lpError );
```

Parameters

pdwLines

Pointer to the row number Here the number of rows printed is returned. Thus the parameter should be preassigned with NULL before the call.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Current pending alarms printed.

FALSE

Error

Error messages

MSG_ERR_NOPROT	Alarm sequence report inactive

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

3.10.5 Functions for alarm processing (RT Professional)

3.10.5.1 MSRTEnumArchivInstancePlus (RT Professional)

Description

Enumerates all alarms in the specified log with instance name. The MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS structure is returned.

Declaration

```
BOOL WINAPI MSRTEnumArchivInstancePlus(  
    DWORD          dwServiceID,  
    BOOL           fArchiv,  
    DWORD          dwMaxRecords,  
    DWORD          dwParams,  
    LPDWORD        lpdwTAID,  
    LPCMN_ERRORW   lpError );
```

Parameters

dwServiceID

ID of the installed service

fArchiv

Circular or sequential log

dwMaxRecords

Maximum number of records for enumeration\n. Enumeration is aborted when the maximum number is exceeded.

dwParams

Supplementary parameter, MSG_ENUM_ARCHIV_DESC = Sort in descending order

lpdwTAID

Pointer to the TransactionID for asynchronous processing feedback

lpError

Pointer to the extended error structure

Return value

TRUE

Function successful

FALSE

Error

3.10.5.2 MSRTCreateMsgInstanceWithCommentPlus (RT Professional)

Description

Generates an alarm with the specified alarm number with instance name and comment. In contrast to most other Create calls, the text associated values may contain up to 256 characters (255 + EndOfString). The alarm is included in the current alarm list with the specified data.

Declaration

```
BOOL WINAPI MSRTCreateMsgInstanceWithCommentPlus(  
    DWORD dwServiceID,  
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS lpMsgCreatePlus,  
    LPDWORD lpdwTAID,  
    LPCMN_ERRORW lpError );
```

Parameters

dwServiceID

ID of the Service - service

lpMsgCreatePlus

Pointer to Createstruct to create the alarm

Use different time stamps for different messages.

lpdwTAID

Pointer to the TransactionID of asynchronous processing feedback

lpError

Pointer that includes the extended error message

Return value

TRUE

Function successful

FALSE

Error

Comment

The Create functions work asynchronously. As a result, you may have to synchronize the calls using semaphores.

If the call is used to acknowledge an alarm, the specified time must be the incoming alarm time stated to the closest millisecond.

**WARNING**

Your application can use this function to acknowledge alarms, without ensuring that the equipment operator has been made aware of the alarms. This circumstance can result in danger to life and limb or property!

Error messages

MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h
 CCMSRTCLIPlus.lib
 CCMSRTCLIPlus.dll

Related functions

MSRTCreateMsgInstancePlus (Page 2407)	Create instance alarm
MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363)
 MSRTStartMsgServicePlus (Page 2389)
 MSRTCreateMsgInstancePlus (Page 2407)
 MSRTSetCommentInstancePlus (Page 2445)
 MSG_TEXTVAL256_STRUCT_PLUS (Page 2386)

3.10.5.3 MSRTCreateMsgInstancePlus (RT Professional)**Description**

Creates an alarm with the stated alarm number. The alarm is included in the current alarm list with the specified data.

Declaration

```
BOOL WINAPI MSRTCreateMsgInstancePlus (
    DWORD                dwServiceID,
    LPMSG_RTDATA_INSTANCE_STRUCT_PLUS lpMsgCreatePlus,
    LPDWORD              lpdwTAID,
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

ID of the installed service

lpMsgCreatePlus

Pointer to Createstruct to create the alarm

Use different time stamps for different messages.

lpdwTAID

Pointer to the TransactionID for asynchronous processing feedback

lpError

Pointer to the extended error structure

Return value

TRUE

Function successful

FALSE

Error

Comment

The Create functions work asynchronously. As a result, you may have to synchronize the calls using semaphores.

If the call is used to acknowledge an alarm, the specified time must be the incoming alarm time stated to the closest millisecond.

**WARNING**

Your application can use this function to acknowledge alarms, without ensuring that the equipment operator has been made aware of the alarms. This circumstance can result in danger to life and limb or property!

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRCreateMsgInstanceWithCommentPlus (Page 2404)	Create instance alarm with comments
MSRTStartMsgServicePlus (Page 2389)	Start service

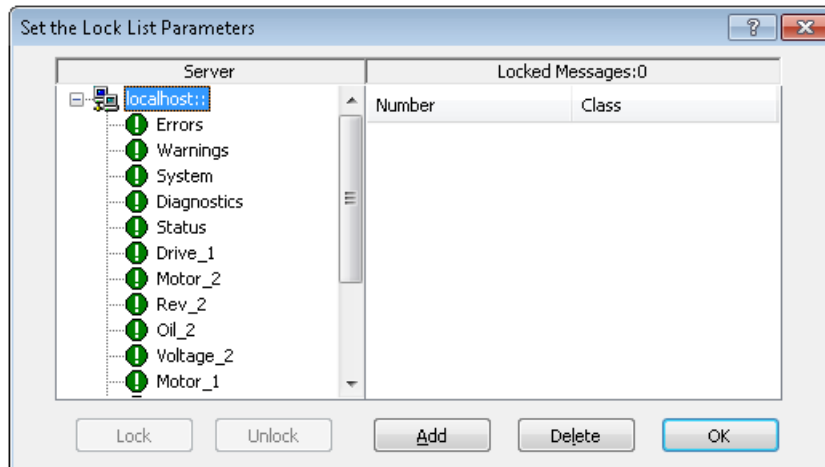
See also

MSRCreateMsgInstanceWithCommentPlus (Page 2404)
MSRTStartMsgServicePlus (Page 2389)
MSG_RTDATA_INSTANCE_STRUCT_PLUS (Page 2371)
MSRTSetCommentInstancePlus (Page 2445)

3.10.5.4 MSRTDialogMsgLockPlus (RT Professional)

Description

Dialog for locking and releasing alarms.



The following options are available:

- Locks on individual alarms
- Locks on alarm classes or alarm groups
- Locks on alarms using the alarm number

Note

Locked alarm:

Locked alarms are neither entered in the alarm lists, logged, nor reported!

Locked alarms send back their acknowledgement signal upon occurrence!

Locked alarms are automatically released when WinCC Runtime restarts. Only alarms that are locked directly in the AS by means of data blocks will continue to be locked (locking at source).

Locked alarm classes/alarm groups:

The locks on alarm classes and alarm groups is not affected by a restart of WinCC Runtime.

Declaration

```
BOOL MSRTDialogMsgLockPlus (
    HWND          hwndParent,
    LPCMN_ERRORW  lpError );
```


Parameters

hwndParent

Parent window, that contains the dialog.

lpError

Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Dialog successfully closed.

FALSE

Error or the dialog was canceled.

Error messages

MSG_ERR_API_PARAM	Parameter invalid

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

3.10.5.5 MSRTCreateMsgPlus (RT Professional)

Description

Creates an alarm with the stated alarm number. The alarm is included in the current alarm list with the specified data.

Declaration

```
BOOL WINAPI MSRTCreateMsgPlus(  
    DWORD                dwServiceID,  
    LPMSG_RTCREATE_STRUCT_PLUS lpMsgCreatePlus,  
    LPDWORD              lpdwTAID,  
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

ID of the installed service

lpMsgCreatePlus

Pointer to Createstruct to create the alarm

Use different time stamps for different messages.

lpdwTAID

Pointer to the TransactionID for asynchronous processing

lpError

Pointer to the extended error structure

Return value

TRUE

Function successful

FALSE

Error

Comment

The Create functions work asynchronously. As a result, you may have to synchronize the calls using semaphores.

**WARNING**

Your application can use this function to acknowledge alarms, without ensuring that the equipment operator has been made aware of the alarms. This circumstance can result in danger to life and limb or property!

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CMSRTCLIPlus.dll

Related functions

MSRTCreateMsgInstanceWithCommentPlus (Page 2404)	Create instance alarm
MSRTCreateMsgInstancePlus (Page 2405)	Create instance alarm with comments
MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTCreateMsgInstanceWithCommentPlus (Page 2404)

MSRTCreateMsgInstancePlus (Page 2405)

MSRTStartMsgServicePlus (Page 2389)

3.10.5.6 MSRTEnumLockedMsgPlus (RT Professional)

Description

Enumerates all the locked alarms.

Enumeration functions work asynchronously. As a result, you must synchronize the calls using semaphores.

Declaration

```
BOOL WINAPI MSRTEnumLockedMsgPlus(  
    DWORD          dwServiceID,  
    LPDWORD        lpdwTAID,  
    LPCMN_ERRORW   lpError);
```

Parameters

dwServiceID

ID of the installed service

lpdwTAID

Pointer to the TransactionID for asynchronous processing

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure.

Return value

TRUE

Alarms enumerated

FALSE

Error.

3.10.5.7 MSRTEnumMsgRTDataPlus (RT Professional)

Description

Enumerates all alarms that are present in the current alarm list. Its callback function MSG_SERVICE_NOTIFY_PROCPPlus callback function is called with notification type MSG_NOTIFY_MSGENUM, enumeration of alarms. lpyData refers to data of the MSG_RTDATA_STRUCTPlus structure.

Enumeration functions work asynchronously. As a result, you must synchronize the calls using semaphores.

Declaration

```
BOOL MSRTEnumMsgRTData (
    DWORD          dwServiceID,
    LPDWORD        pdwTAID
    LPCMN_ERRORW   lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpdwTAID

Pointer to the TransactionID for asynchronous processing.

LPCMN_ERRORW

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Alarms enumerated

FALSE

Error.

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

3.10.5.8 MSRTGetClassInfoPlus (RT Professional)

Description

Get information about an alarm class.

Declaration

```
BOOL WINAPI MSRTGetClassInfoPlus(  
    DWORD                dwServiceID,  
    LPCWSTR              lpszServer,  
    DWORD                dwMsgNrLow,  
    DWORD                dwMsgNrHigh,  
    LPMSG_CLASS_STRUCT_PLUS lpClassPlus,  
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

ID of the Service - service whose server is to be connected

lpszServer

Server to which interconnection is to be carried out

dwMsgNrLow

Alarm number of the lower DWORD

dwMsgNrHigh

Alarm number of the upper DWORD WinCC

lpClassPlus

Pointer that includes the information

lpError

Pointer that includes the extended error message

Return value**TRUE**

Function successful

FALSE

Error

Comment

The MSRTGetClassInfoPlus function can also be called up without MSRTStartMsgServicePlus. These calls are however significantly slower. You can improve performance by performing a MSRTStartMsgServicePlus beforehand and retaining it for all calls. In this case only the first call of MSRTGetClassInfoPlus is slowed.

Error messages

MSG_ERR_RT_NOINIT	Alarm Logging RT is not initialized
MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_MSG_NOEXIST	Alarm does not exist

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_CLASS_STRUCT_PLUS (Page 2374)

MSRTStartMsgServicePlus (Page 2389)

3.10.5.9 MSRTGetLastMsgWithCommentPlus (RT Professional)

Description

Retrieves the last alarm from a log based on filter conditions and an instance name.

Declaration

```
BOOL WINAPI MSRTGetLastMsgWithCommentPlus(  
    LPCWSTR  
    LPCWSTR  
    LPMSG_FILTER_STRUCT_PLUS*  
    DWORD  
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS  
    LPCMN_ERRORW  
    lpszServer,  
    lpszInstance,  
    lppFilter,  
    dwFilterCount,  
    lpMsg,  
    lpError );
```

Parameters

lpszServer

Server prefix without '::.'. If an empty string or NULL is input the default server set will be used.

lpszInstance

Instance name. An empty string or also NULL will be added for the filter condition.

lppFilter

Pointer to a pointer list of structures MSG_FILTER_STRUCT_PLUS for transferring filter conditions.

dwFilterCount

Number of specified filter structures via lppFilter. At least one filter structure must be specified.

lpMsg

Returns the message data.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure.

Return value**TRUE**

Alarm queried.

FALSE

Error.

Error messages

MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_RT_NOTCONNECTED	Connection not made

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

AUTOHOTSPOT	Querying the attributes of a message

See also

MSG_FILTER_STRUCT_PLUS (Page 2367)

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363)

MSG_TEXTVAL256_STRUCT_PLUS (Page 2386)

3.10.5.10 MSRTGetMsgActualPlus (RT Professional)**Description**

The function states the number of currently pending alarms in the alarm list.

Declaration

```
BOOL MSRTGetMsgActualPlus (  
    LPDWORD      lpdwCount,  
    LPCMN_ERRORW lpError );
```


Parameters

lpdwCount

Pointer to a buffer in which the number of currently pending alarms is stored.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Number stated.

FALSE

Error

Comment

The MSRTGetMsgActualPlus function can also be called up without MSRTStartMsgServicePlus. These calls are however significantly slower. You can improve performance by performing a MSRTStartMsgServicePlus beforehand and retaining it for all calls. In this case only the first call of MSRTGetMsgActualPlus is slowed.

Error messages

MSG_ERR_RT_NOINIT	Alarm system not initialized

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

3.10.5.11 MSRTGetMsgCSDDataPlus (RT Professional)**Description**

Query the configured data for an alarm.

Declaration

```

BOOL MSRTGetMsgCSDDataPlus (
    DWORD                dwServiceNr,
    LPCWSTR              lpszServer,
    DWORD                dwMsgNrLow,
    DWORD                dwMsgNrHigh,
    LPMSG_CSDATA_STRUCT_PLUS lpmCSDDataPlus,
    LPCMN_ERRORW         lpError );

```

Parameters**dwServiceID**

Identification number of the send and receive service, as returned when MSRTStartMsgService is called.

lpszServer

Server name.

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

lpmCSDDataPlus

Pointer to the configuration data (alarm class, type of alarm, text block index, etc.) for this alarm in the MSG_CSDATA_STRUCT_PLUS (Page 2376) structure.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Configuration data queried.

FALSE

Error

Comment

The MSRTGetMsgCSDDataPlus function can also be called up without MSRTStartMsgServicePlus. These calls are however significantly slower. You can improve performance by performing a MSRTStartMsgServicePlus beforehand and retaining it for all calls. In this case only the first call of MSRTGetMsgCSDDataPlus is slowed.

Error messages

MSG_ERR_RT_NOINIT	Alarm system not initialized
MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_MSG_NOEXIST	Alarm does not exist

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_CSDATA_STRUCT_PLUS (Page 2376)

MSRTStartMsgServicePlus (Page 2389)

3.10.5.12 MSRTGetMsgPriorityPlus (RT Professional)**Description**

Query the priority of the given alarm.

Declaration

```
BOOL MSRTGetMsgPriorityPlus (
    DWORD          dwServiceID,
    LPCWSTR        lpszServer,
    DWORD          dwMsgNrLow,
    DWORD          dwMsgNrHigh,
    long*          plPriority,
    LPCMN_ERRORW   lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgService is called.

lpszServer

Server name.

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

plPriority

Pointer to a tag of the long data type, in which the priority is stored.

This parameter must not be NULL!

lpError

Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Priority queried.

FALSE

Error

Comment

The MSRTGetMsgPriorityPlus function can also be called up without MSRTStartMsgServicePlus. These calls are however significantly slower. You can improve performance by performing a MSRTStartMsgServicePlus beforehand and retaining it for all calls. In this case only the first call of MSRTGetMsgPriorityPlus is slowed.

Error messages

MSG_ERR_RT_NOINIT	Alarm system not initialized
MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_MSG_NOEXIST	Alarm does not exist

Required files

CCMSRTCLIPlus.h
 CCMSRTCLIPlus.lib
 CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

3.10.5.13 MSRTGetMsgQuitPlus (RT Professional)**Description**

Query the number of alarms that require acknowledgement.

Declaration

```

BOOL MSRTGetMsgQuitPlus(
    LPDWORD      lpdwCount,
    LPCMN_ERRORW lpError );

```

Parameters**lpdwCount**

Pointer for the number of alarms that require acknowledgement

lpError

Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Number stated.

FALSE

Error

Comment

The MSRTGetMsgPriorityPlus function can also be called up without MSRTStartMsgServicePlus. These calls are however significantly slower. You can improve performance by performing a MSRTStartMsgServicePlus beforehand and retaining it for all calls. In this case only the first call of MSRTGetMsgPriorityPlus is slowed.

Error messages

MSG_ERR_RT_NOINIT	Alarm system not initialized

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

3.10.5.14 MSRTGetMsgTextPlus (RT Professional)**Description**

Get the text for a text block

Declaration

```
BOOL WINAPI MSRTGetMsgTextPlus(  
    WORD                wTextBlock,  
    DWORD               dwTextNr,  
    LPWSTR              lpszMsgText,  
    DWORD*              lpdwCount,  
    LPCMN_ERRORW        lpError );
```

Parameters

wTextBlock

Text block number

dwTextNr

Text block ID

lpszMsgText

Pointer to the text block string

lpdwCount

Pointer to the length of the text block string. The length is specified as the number of characters.

lpdwCount=0	Text size is queried.
lpdwCount>0 AND buffer size of lpszMsgText > lpdwCount	Text is limited, and current number of characters is returned.
lpdwCount>0 AND buffer size of lpszMsgText > lpdwCount AND buffer initialized	Text is limited and returned.

lpError

Pointer that includes the extended error message

Return value

TRUE

Function successful

FALSE

Error

Comment

The MSRTGetMsgPriorityPlus function can also be called without MSRTStartMsgServicePlus. These calls are however significantly slower. You can improve performance by performing a MSRTStartMsgServicePlus beforehand and retaining it for all calls. In this case only the first call of MSRTGetMsgPriorityPlus is slowed.

Error messages

MSG_ERR_RT_NOINIT	Alarm system not initialized

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

3.10.5.15 MSRTGetSelectedMsgPlus (RT Professional)**Description**

The function determines the alarm number for an alarm selected in an alarm window or an alarm view. The view is specified using lpszTemplate.

In individual or multiple user projects queries work on the template without restriction. Queries in the display do not work for engineering data and only on the display of existing process values.

Declaration

```
BOOL MSRTGetSelectedMsgPlus (
    LPCWSTR                                lpszTemplate,
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS lpMsgRTPlus,
    LPCMN_ERROR                             lpError );
```


Parameters

lpSzTemplate

Pointer to the name of the view in which an alarm is selected.

lpMsgRTPlus

Pointer to a structure of the MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363) type, in which the alarm number is returned.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Data queried.

FALSE

Error

Comment

The MSRTGetMsgPriorityPlus function can also be called up without MSRTStartMsgServicePlus. These calls are however significantly slower. You can improve performance by performing a MSRTStartMsgServicePlus beforehand and retaining it for all calls. In this case only the first call of MSRTGetMsgPriorityPlus is slowed.

Note

According to which view is set in the display, the flags in dwFlags of the MSG_RTDATA_INSTANCECOMMENT_STRUCTPlus structure will be assigned differently:

- for the online view: MSG_RT_FLAG_xxx flags,
 - for the log view: MSG_FLAG_ flags, and
 - for the hit list: no assignment is possible and flags will be ignored.
-

Error messages

MSG_ERR_API_PARAM	Invalid parameter

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

Examples

Get selected Msg "MS02.cpp"

See also

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363)

MSRTStartMsgServicePlus (Page 2389)

3.10.5.16 MSRTResetMsgPlus (RT Professional)**Description**

The stated alarm will be acknowledged.

Declaration

```

BOOL MSRTResetMsgPlus (
    DWORD          dwServiceID,
    DWORD          dwMsgNrLow,
    DWORD          dwMsgNrHigh,
    LPDWORD        lpdwTAID
    LPCMN_ERRORW   lpError );

```

Parameters**dwServiceID**

Identification number of the send and receive service, as returned when MSRTStartMsgService is called.

dwMsgNrLow

dwMsgNrLow Number of the first alarm

dwMsgNrHigh

dwMsgHigh Number of the last alarm

You must assign dwMsgNrLow and dwMsgNrHigh, if you use the MSG_FILTER_NR, MSG_FILTER_NR_FROM, or MSG_FILTER_NR_TO filter criteria.

lpdwTAID

Pointer to the TransactionID for asynchronous processing.

lpError

Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information to this structure.

Return value

TRUE

Alarm acknowledged.

FALSE

Error

Comment

WARNING

Your application can use this function to acknowledge alarms, without ensuring that the equipment operator has been made aware of the alarms. This circumstance can result in danger to life and limb or property!

Error messages

MSG_ERR_RT_NOINIT	Alarm system not initialized
MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_MSG_NOEXIST	Alarm does not exist
MSG_ERR_MSG_NOTFOUND	Alarm not found
MSG_ERR_MSG_NOQUIT	Alarm does not require acknowledgment
MSG_ERR_MSG_ALREADYQUIT	Alarm already acknowledged

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

3.10.5.17 MSRTLoopInAlarmPlus (RT Professional)

Description

API function for loop-in-alarm The alarm number is decisive.

Declaration

```
BOOL MSRTLoopInAlarmPlus(  
    DWORD          dwServiceID,  
    LPCWSTR        lpszServer,  
    DWORD          dwMsgNrLow,  
    DWORD          dwMsgNrHigh,  
    LPCMN_ERRORW   lpError );
```

Parameters

dwServiceID

ID of the Service - service whose server is to be connected

lpszServer

Server to which interconnection is to be carried out

dwMsgNrLow

Alarm number of the lower DWORD

dwMsgNrHigh

Alarm number of the upper DWORD WinCC

lpError

Pointer that includes the extended error message

Return value

TRUE

Function successful

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_MSG_NOEXIST	Alarm does not exist

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

3.10.6 Functions for alarm group processing (RT Professional)

3.10.6.1 MSRTEnumGroupMsgPlus (RT Professional)

Description

Enumerate all single alarms of the alarm group. MSG_MAX_GROUPITEMS gives the maximum number of single alarms that can be enumerated. Its callback function MSG_SERVICE_NOTIFY_PROCPlus is called using the MSG_NOTIFY_GROUPENUM notification type. lpbyData refers to data of the MSG_RTGROUPENUM_STRUCTPlus structure.

Enumeration functions of the alarm system run asynchronously. As a result, you must synchronize the calls using semaphores.

Declaration

```
BOOL WINAPI MSRTEnumGroupMsgPlus (
    DWORD                dwServiceID,
    LPMSG_RTGROUPSET_STRUCT_PLUS  lpmGroupPlus,
    LPDWORD              lpdwTAID,
    LPCMN_ERROR          lpError );
```

Parameters

dwServiceID

ID of the Service - service

lpmGroupPlus

Pointer to the alarm group info

lpdwTAID

Pointer to the TransactionID for asynchronous processing feedback

lpError

Pointer that includes the extended error message

Return value**TRUE**

Function successful

FALSE

Error

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSG_SERVICE_NOTIFY_PROCPlus (Page 2395)	Callback function
MSRTStartMsgServicePlus (Page 2389)	Start service
MSG_RTGROUPENUM_STRUCT_PLUS (Page 2379)	Single alarms of an alarm group

See also

MSG_RTGROUPSET_STRUCT_PLUS (Page 2380)

MSRTStartMsgServicePlus (Page 2389)

MSG_SERVICE_NOTIFY_PROCPlus (Page 2395)

MSG_RTGROUPENUM_STRUCT_PLUS (Page 2379)

3.10.6.2 MSRTLockGroupPlus (RT Professional)

Description

Lock a configured alarm group. All individual alarms in the alarm group and also individual alarms in underlying groups will be locked.

Declaration

```

BOOL MSRTLockGroupPlus (
    DWORD                dwServiceID,
    LPMSG_RTGROUPSET_STRUCTPlus  lpmGroup,
    LPDWORD              pdwTAID
    LPCMN_ERRORW         lpError );

```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpmGroup

Pointer to the MSG_RTGROUPSET_STRUCT_PLUS (Page 2380) structure with information on the alarm groups.

Lock: dwData != 0; enable: dwData = 0

pdwTAID

lpError

Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Alarms in the alarm group locked.

FALSE

Error

Comment

Only configured alarm groups will be locked or released. Groups defined in runtime will not be included and the function will have no effect.

Error messages

MSG_ERR_RT_NOINIT	Alarm system not initialized
MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h
 CCMSRTCLIPlus.lib
 CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_RTGROUPSET_STRUCT_PLUS (Page 2380)
 MSRTStartMsgServicePlus (Page 2389)

3.10.6.3 MSRTQuitGroupPlus (RT Professional)**Description**

Acknowledgment of an alarm group. All single alarms of the alarm group as well as the alarms of subordinate groups are acknowledged.

Declaration

```

BOOL MSRTQuitGroupPlus (
    DWORD                dwServiceID,
    LPMSG_RTGROUPSET_STRUCTPlus  lpmGroupPlus,
    LPDWORD              pdwTAID,
    LPCMN_ERRORW         lpError );

```

Parameters**dwServiceID**

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpmGroup

Pointer to a structure of the MSG_RTGROUPSET_STRUCT_PLUS (Page 2380) type with information of the alarm groups.

pdwTAID**lpError**

Pointer to the data of the extended error message in the CMN_ERRORW structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Alarms in the alarm group are acknowledged.

FALSE

Error

Required files

CMSRTCLIPlus.h

CMSRTCLIPlus.lib

CMSRTCLIPlus.dll

See also

MSG_RTGROUPSET_STRUCT_PLUS (Page 2380)

3.10.7 Functions for alarm filter processing (RT Professional)**3.10.7.1 MSRTGetFilterDataPlus (RT Professional)****Description**

Reads the selection criteria of the alarm view and writes the result to the MSG_FILTER_STRUCT_PLUS structure.

Declaration

```
BOOL WINAPI MSRTGetFilterDataPlus(  
    DWORD          dwServiceID,  
    LPCWSTR        lpszName,  
    LPMMSG_FILTER_STRUCT_PLUS** lpppMsgFilter,  
    LPDWORD        lpdwFilterCount,  
    LPCMN_ERRORW   lpError );
```

Parameters

dwServiceID

ID of the Service - service

lpszName

Alarm view name. You assign the name in the alarm view properties.

lpMsgFilterPlus

Pointer to array filter pointer to filter structures

lpdwFilterCount**lpError**

Pointer that includes the extended error message

Return value

TRUE

Function successful

FALSE

Error

Comment

Only configured alarm filters are determined. Alarm filters that are set using MSRTStartMsgServicePlus or MSRTSetMsgFilterPlus are not included.

Error messages

MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_API_SERVICE	dwServiceID invalid
MSG_ERR_API_NODATA	No data

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTSetMsgFilterPlus (Page 2439)	Set alarm filter
MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_FILTER_STRUCT_PLUS (Page 2367)
MSRTStartMsgServicePlus (Page 2389)
MSRTSetMsgFilterPlus (Page 2439)

3.10.7.2 MSRTCheckWinFilterPlus (RT Professional)

Description

Tests if the stated alarm corresponds to a filter parameter.

Declaration

```
BOOL MSRTCheckWinFilterPlus (
    DWORD                dwServiceID,
    LPCWSTR              lpszServer,
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS lpMsgRTPlus,
    LPMSG_FILTER_STRUCT_PLUS* lppMsgFilter,
    DWORD                dwFilterCount,
    LPWORD               lpwReturn,
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpszServer

Pointer to the server name. The name will be stated without any :: server separator characters. If an empty string or NULL is input the default server set will be used

IpMsgRTPlus

Pointer to a structure of the MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363) type with the runtime data of an alarm.

IpMsgFilter

Pointer to a structure of the MSG_FILTER_STRUCT_PLUS (Page 2367) type with the alarm filter data.

IpwReturn

Pointer to the function results

TRUE	Alarm corresponds to the filter conditions.
FALSE	Alarm does not correspond to the filter conditions.

IpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Testing will be performed.

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Parameter invalid

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_FILTER_STRUCT_PLUS (Page 2367)

MSRTStartMsgServicePlus (Page 2389)

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (Page 2363)

3.10.7.3 MSRTSetMsgFilterPlus (RT Professional)

Description

Resets the filter for the specified service.

Declaration

```
BOOL WINAPI MSRTSetMsgFilterPlus(  
    DWORD                dwServiceID,  
    LPMMSG_FILTER_STRUCT_PLUS* lppMsgFilterPlus,  
    DWORD                dwFilterCount,  
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lppMsgFilterPlus

Pointer to ARRAY pointer for filter conditions

dwFilterCount

Number of filters in the ARRAY

lpError

Pointer to the data of the extended error message in the CMN_ERRORW structure. The system writes error information into this structure in the case of an error.

Return value

TRUE

Function successful

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPus.h
CCMSRTCLIPus.lib
CCMSRTCLIPus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service
MSRTGetFilterDataPlus (Page 2433)	Read selection criteria of the alarm view

See also

MSRTStartMsgServicePlus (Page 2389)
MSG_FILTER_STRUCT_PLUS (Page 2367)
MSRTGetFilterDataPlus (Page 2433)

3.10.7.4 MSRTSetMsgWinFilterPlus (RT Professional)

Description

Defines new filter conditions for the alarm view. All current alarm views at preset with this name will be updated.

Declaration

```
BOOL MSRTSetMsgWinFilterPlus (
    LPMSG_FILTER_STRUCT_PLUS    lppMsgFilterPlus,
    DWORD                      dwFilterCount,
    LPCMN_ERRORW                lpError );
```

Parameters

lppMsgFilterPlus

Pointer to a structure of the MSG_FILTER_STRUCT_PLUS (Page 2367) type with the alarm filter data. The name of the filter structure is the same as the window name.

dwFilterCount

Number of filters

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Filter was set

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

See also

MSG_FILTER_STRUCT_PLUS (Page 2367)

3.10.8 Functions for processing alarm view (RT Professional)**3.10.8.1 MSRTMsgWinCommandPlus (RT Professional)****Description**

Performs a control function in an alarm view. If you use this function, you only use an alarm window template once within a screen. The commands are also passed on to controls with the caption name identical to the template name. (When screens with alarm templates are converted to screens with controls, the resulting controls have a caption name identical to the template name.)

Declaration

```

BOOL MSRTMsgWinCommand(
    LPTSTR      lp.szTemplate,
    DWORD       dwCommandID,
    LPCMN_ERROR lpError );

```

Parameters

lpzTemplate

Pointer to the window name

dwCommandID

The ID of the control function is specified using the constants in the "m_global.h" file:

MSG_TB_MSGWIN	Open process alarm window
MSG_TB_ARC_S	Open circular log
MSG_TB_ARC_L	Open sequential log
MSG_TB_QH	Initiate horn acknowledgment
MSG_TB_QM	Initiate single acknowledgment
MSG_TB_QS	Initiate group acknowledgment
MSG_TB_SCROLL	Switch scroll function on/off.
MSG_TB_SELECT	Display selection dialog
MSG_TB_LOCK	Display lock dialog
MSG_TB_PROTOCOL	Display report dialog
MSG_TB_RESET	Initiate emergency acknowledgment
MSG_TB_MSGFIRST	Move to start of list (autoscroll must not be activated)
MSG_TB_MSGLAST	Move to end of list (autoscroll must not be activated)
MSG_TB_MSGNEXT	Move to next alarm (autoscroll must not be activated)
MSG_TB_MSGPREV	Move to previous alarm (autoscroll must not be activated)
MSG_TB_INFO	Display info text dialog for currently selected alarm
MSG_TB_COMMENT	Enter comment for currently selected alarm
MSG_TB_LOOPINALARM	Initiate loop-in-alarm for currently selected alarm
MSG_TB_PRINT	Start print

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Control function executed

FALSE

Error

Comment

All opened alarm views with the specified name execute the control function.

Error messages

MSG_ERR_API_PARAM	Parameter invalid

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

3.10.9 Functions for processing comments (RT Professional)

3.10.9.1 MSRTGetCommentInstancePlus (RT Professional)

Description

Gets the comment text for a logged alarm.

Declaration

```
BOOL MSRTGetCommentInstancePlus (
    DWORD                dwServiceID,
    LPMSG_COMMENT_INSTANCE_STRUCT_PLUS lpmCommentPlus,
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpmCommentPlus

Pointer to a MSG_COMMENT_INSTANCE_STRUCT_PLUS (Page 2381) type structure with the data of a comment. The following values in the structure are necessary: Alarm number, date and time, and instance name.

Note

Date and time must exactly match the time stamp of the message for which the comment text is to be determined to the millisecond.

lpError

Pointer to the data of the extended error message in the CMN_ERROR W structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

Comment text determined.

If no comment exists, the call returns TRUE, but the szText element is empty.

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)

MSG_COMMENT_INSTANCE_STRUCT_PLUS (Page 2381)

3.10.9.2 MSRTSetCommentInstancePlus (RT Professional)

Description

Specifies the comment text for a logged alarm.

Declaration

```
Bool MSRTSetCommentInstancePlus (
    DWORD                dwServiceID,
    LPMSG_COMMENT_INSTANCE_STRUCT_PLUS lpmCommentPlus,
    LPDWORD              pdwTAID,
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpmCommentPlus

Pointer to a MSG_COMMENT_INSTANCE_STRUCT_PLUS (Page 2381) type structure with the data of a comment. The following values in the structure are necessary: Alarm number, instance name, date, time of day, text, and user name.

pdwTAID

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Successfully sent to server.

FALSE

Error

Comment

The data of the MSRTSetCommentPlus function are written asynchronously. As a result, not all errors are returned to the user. Use MSRTGetCommentInstancePlus to check how data are written.

If MSRTSetCommentInstancePlus is executed after MSRTCreateMsgPlus or MSRTCreateMsgInstancePlus, the asynchronous writing along with the degree of utilization of the system can cause data to be "overtaken" by other data. As a result, the comment is specified before the alarm is created.

Instead of the call combination, use the MSRTCreateMsgInstanceWithCommentPlus function.

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service
MSRTCreateMsgInstanceWithCommentPlus (Page 2404)	Create alarm
MSRTCreateMsgInstancePlus (Page 2405)	Create alarm

See also

MSG_COMMENT_INSTANCE_STRUCT_PLUS (Page 2381)
MSRTStartMsgServicePlus (Page 2389)
MSRTCreateMsgInstanceWithCommentPlus (Page 2404)
MSRTCreateMsgInstancePlus (Page 2405)

3.10.10 Functions for info text processing (RT Professional)

3.10.10.1 MSRTGetInfotextPlus (RT Professional)

Description

Gets the info text for an alarm. The following values have to be transferred to this purpose:
Alarm number

Declaration

```
BOOL WINAPI MSRTGetInfotextPlus (
    DWORD                dwServiceID,
    LPMSG_INFOTEXT_STRUCT_PLUS  lpmInfotext,
    LPCMN_ERRORW          lpError );
```

Parameters

dwServiceID

ID of the Service - service

lpmlInfotext

Pointer of the info text structure

lpError

Pointer that includes the extended error message

Return value

TRUE

Function successful

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_INFOTEXT_STRUCT_PLUS (Page 2382)

MSRTStartMsgServicePlus (Page 2389)

3.10.10.2 MSRTSetInfotextPlus (RT Professional)

Description

Sets the info text for an alarm.

Declaration

```
BOOL WINAPI MSRTSetInfotextPlus(  
    DWORD dwServiceID,  
    LPMSG_INFOTEXT_STRUCT_PLUS lpmInfotextPlus,  
    DWORD* lpdwTAID,  
    LPCMN_ERRORW lpError );
```

Parameters

dwServiceID

ID of the Service - service

lpmInfotextPlus

Pointer of the info text structure

lpdwTAID

Pointer to the TransactionID with feedback for asynchronous processing.

lpError

Pointer that includes the extended error message.

Return value

TRUE

Function successful

FALSE

Error

Comment

The new info text will be effective from the next minute changeover after one. Until then the previous info text will be used.

Error messages

MSG_ERR_API_PARAM	Invalid parameter
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h
CCMSRTCLIPlus.lib
CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSG_INFOTEXT_STRUCT_PLUS (Page 2382)
MSRTStartMsgServicePlus (Page 2389)

3.10.11 Archiving functions (RT Professional)

3.10.11.1 MSRTEnumBackupListPlus (RT Professional)

Description

Lists the entries of the export file for the sequential archive.

Declaration

```
BOOL MSRTEnumBackupListPlus (  
    DWORD          dwServiceID,  
    LPCMN_ERROR    lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Export list created.

FALSE

Error

Comment

The Enum functions work asynchronously. This has the result that any calls must be synchronized by semaphore

The callback function MSG_SERVICE_NOTIFY_PROCPlus is called using the MSG_NOTIFY_BACKUPENUM parameter. The lpbyData parameter points to the data in the MSG_BACKUP_STRUCT_PLUS (Page 2383) structure.

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service
MSG_SERVICE_NOTIFY_PROCPlus (Page 2395)	Callback function

See also

MSG_BACKUP_STRUCT_PLUS (Page 2383)

MSRTStartMsgServicePlus (Page 2389)

MSG_SERVICE_NOTIFY_PROCPlus (Page 2395)

3.10.11.2 MSRTExportPlus (RT Professional)

Description

Exports alarms from the circular log and the related log backup files.

Declaration

```
BOOL MSRTExportPlus (
    DWORD                dwServiceID,
    LPMSG_BACKUP_STRUCT_PLUS lpMsgBackup,
    LPCMN_ERRORW         lpError );
```

Parameters

dwServiceID

Identification number of the send and receive service, as returned when MSRTStartMsgServicePlus is called.

lpMsgBackup

Pointer to a structure of the MSG_BACKUP_STRUCT_PLUS (Page 2383) type.

lpError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value

TRUE

Alarms exported.

FALSE

Error

Error messages

MSG_ERR_API_PARAM	Parameter invalid
MSG_ERR_API_SERVICE	dwServiceID invalid

Required files

CCMSRTCLIPlus.h
 CCMSRTCLIPlus.lib
 CCMSRTCLIPlus.dll

Related functions

MSRTStartMsgServicePlus (Page 2389)	Start service

See also

MSRTStartMsgServicePlus (Page 2389)
 MSG_BACKUP_STRUCT_PLUS (Page 2383)

3.11 Functions for displaying the PLC code (RT Professional)

3.11.1 Display in STEP 7 (RT Professional)

3.11.1.1 Basics (RT Professional)

General information

You can use this function to switch from a screen in WinCC Runtime directly to the point of use of a process tag in the program code of STEP 7. This provides the possibility of quick and simple diagnostics of faults. To speed up the entry point functions, you can start the TIA Portal and open a project. This is typically done at the startup of WinCC Runtime.

A check is first made to determine whether the TIA Portal is already open at the entry point. If this is not the case, the TIA Portal is automatically started. The corresponding editor opens in the TIA Portal and a search is performed for the point of use, assignment, call or step.

Header files

The declaration of the functions and structures explained here takes place in the header files:

kopapi.h	API interface definition header file

Libraries

The DLL link for the functions explained here is provided in the kopapi.dll "lib" files

kopapi.lib	Library
kopapi.dll	

See also

[OpenTIAPortalProject \(Page 2453\)](#)
[OpenTIAPortalIECPLByCall \(Page 2455\)](#)
[OpenTIAPortalIECPLByAssignment \(Page 2457\)](#)
[OpenTIAPortalS7GraphByBlock \(Page 2459\)](#)
[Example: Integration in a WinCC function \(Page 2462\)](#)

3.11.1.2 OpenTIAPortalProject (RT Professional)

Description

You can use this function to start the TIA Portal and open a project. This will speed up the individual jump functions. However, you should consider that a program open in the background requires memory and computation time.

Declaration

```

BOOL OpenTIAPortalProject (
    DWORD      dwFlags,
    LPCTSTR    lpszTiaPortalProjectPath,
    LPCTSTR    lpszErrorTag,
    LPCMN_ERROR lpdmError);

```

Parameters

dwFlags

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

- KOPAPI_FLAG_TIAPORTAL_CHECK_PROJECT_STATE: If this bit is set, the status of the project is checked. The project is not opened.
- The project is open.
Return value FALSE
error.dwError1 = KOPAPI_E_TIAPORTAL_PROJECT_NOT_OPEN
- The project is open and contains no changes.
Return value TRUE
error.dwError1 = 0
- The project is open and contains changes.
Return value FALSE
error.dwError1 = KOPAPI_E_TIAPORTAL_PROJECT_MODIFIED
- The project is opened read only.
Return value FALSE
error.dwError1 = KOPAPI_E_TIAPORTAL_PROJECT_READ_ONLY

KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT: If this bit is set, the call is canceled when the project is already open and contains changes.

KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY: When this bit is set to TRUE, the project is opened in write-protected mode in the engineering system of the TIA Portal.

lpszTiaPortalProjectPath

Name of the project file including specification of the absolute path, for example, "D:\TIAProjects\Project1\Project1.ap14"

Please note that within a C script the backslash has to be written using an Escape sequence: FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);

lpszErrorTag

Name of an internal WinCC tag of the String data type. The error information is returned in lpszErrorTag (Page 2474) when asynchronous functions that do not provide instant results are called.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. The system writes error information to this structure if an error occurs.

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h
 kopapi.lib
 kopapi.dll

See also

Basics (Page 2450)

3.11.1.3 OpenTIAPortalIECPLByCall (RT Professional)**Description**

The function is used for the LAD and FBD languages and shows the preceding logic of a network input of a standard block.

Declaration

```

BOOL OpenTIAPortalIECPLByCall (
    DWORD          dwFlags,
    LPCTSTR        lpszTiaPortalProjectPath,
    LPCTSTR        lpszCpuName,
    LPCTSTR        lpszContainingBlock,
    LPCTSTR        lpszCalledBlock,
    LPCTSTR        lpszPin,
    LPCTSTR        lpszErrorTag,
    LPCMN_ERROR    lpdmError);
  
```

Parameters**dwFlags**

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

- IECPLVIEWER_PIN_SUBSTRING_SEARCH=0x0001: When searching for the pin name, a substring is searched for, which means the pin name starts with the string passed in lpszPin. If this bit is not set, the complete pin name is compared to lpszPin.
- KOPAPI_FLAG_TIAPORTAL_SUPPRESS_PROGRAM_STATUS=0x0004: The TIA Portal does not go into online mode after the block is opened. If this bit is not set, online mode is started after opening the block.
- KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT 0x0008L: If this bit is set, the call is canceled when the project is already open and contains changes.
- KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY: When this bit is set to TRUE, the project is opened in write-protected mode in the engineering system of the TIA Portal.

lpszTiaPortalProjectPath

Name of the project file including specification of the absolute path, for example, "D:\TIAProjects\Project1\Project1.ap14"

Please note that within a C script the backslash has to be written using an Escape sequence: FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);

lpszCpuName

Name of the S7 CPU. The name is identical to the station name displayed in the project tree in the TIA Portal.

lpszContainingBlock

Name of the block to be opened and displayed or name of the instance of an FB.

The following can be used as the name:

- Name of a single instance DB. Its FB is then displayed. Example "Station1"
- Name of a multi-instance in an instance DB. Its FB is then displayed. When multiinstance name paths are specified, these are data hierarchies such as those displayed in the DB editor, and not those in the call structure. The first part of the name ("Line1") cannot be enclosed in quotes, because this is a global icon, as can be recognized by the context. Quotation marks are required for the individual name components when special characters such as spaces, dots, etc., occur in them. Example: "Line1.Cell1.Station1"
- Name of an FC or OB

The use of the name of a FB is not allowed.

lpszCalledBlock

Name of the local or global instance that is called in the code block belonging to lpszContainingBlock.

- For local instances, the hash sign # must be specified here, for example, "#feeder1".
- For global instances, the global name without hash sign # must be specified here, for example, "feeder3".

The use of the name of an FC is not allowed.

If lpszCalledBlock is called several times within lpszContainingBlock or its FB, the entry point is always the first call of lpszCalledBlock.

Only lpszContainingBlock is opened and displayed in the status with lpszCalledBlock=ZERO, but there is no search for a specific block call or a specific network. lpszPin is ignored in this case.

lpszPin

Name of the input pin of lpszCalledBlock. The parameter is used to make the specified pin visible within the network in the editor.

Only lpszCalledBlock is shown as visible with lpszPin=ZERO.

lpszErrorTag

Name of an internal WinCC tag of the String data type. The error information is returned in lpszErrorTag (Page 2474) when asynchronous functions that do not provide instant results are called.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h

kopapi.lib

kopapi.dll

See also

Basics (Page 2450)

3.11.1.4 OpenTIAPortalIECPLByAssignment (RT Professional)**Description**

The function is used for the LAD and FBD languages and shows the assignment to an operand and its preceding logic.

Declaration

```

BOOL OpenTIAPortalIECPLByAssignment (
    DWORD          dwFlags,
    LPCTSTR        lpszTiaPortalProjectPath,
    LPCTSTR        lpszCpuName,
    LPCTSTR        lpszContainingBlock,
    LPCTSTR        lpszOperand,
    LPCTSTR        lpszErrorTag,
    LPCMN_ERROR    lpdmError);

```

Parameters

dwFlags

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

- KOPAPI_FLAG_TIAPORTAL_SUPPRESS_PROGRAM_STATUS=0x0004: The TIA Portal does not go into online mode after the block is opened. If this bit is not set, online mode is started after opening the block.
- KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT 0x0008L: If this bit is set, the call is canceled when the project is already open and contains changes.
- KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY: When this bit is set to TRUE, the project is opened in write-protected mode in the engineering system of the TIA Portal.

lpszTiaPortalProjectPath

Name of the project file including specification of the absolute path, for example, "D:\TIAProjects\Project1\Project1.ap14"

Please note that within a C script the backslash has to be written using an Escape sequence: `FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);`

lpszCpuName

Name of the S7 CPU. The name is identical to the station name displayed in the project tree in the TIA Portal.

lpszContainingBlock

Name of the block to be opened and displayed or name of the instance of an FB.

The following can be used as the name:

- Name of a single instance DB. Its FB is then displayed. Example "Station1"
- Name of a multi-instance in an instance DB. Its FB is then displayed. When multiinstance name paths are specified, these are data hierarchies such as those displayed in the DB editor, and not those in the call structure. The first part of the name ("Line1") cannot be enclosed in quotes, because this is a global icon, as can be recognized by the context. Quotation marks are required for the individual name components when special characters such as spaces, dots, etc., occur in them. Example: "Line1.Cell1.Station1"
- Name of an FC or OB

The use of the name of a FB is not allowed.

lpszOperand

Name of a local or global operand to which an assignment is made.

Name of the local or global instance that is called in the code block belonging to lpszContainingBlock.

- For local operands, the hash sign # must also be specified here.
- For global operands, the global name without hash sign # must be specified.

If lpszOperand is written several times within lpszContainingBlock or its FB, the entry point is always the first write access of lpszOperand.

lpszErrorTag

Name of an internal WinCC tag of the String data type. The error information is returned in lpszErrorTag (Page 2474) when asynchronous functions that do not provide instant results are called.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h

kopapi.lib

kopapi.dll

See also

Basics (Page 2450)

3.11.1.5 OpenTIAPortalS7GraphByBlock (RT Professional)**Description**

The function is used for the S7 Graph languages and shows a step in a sequencer.

Declaration

```
BOOL OpenTIAPortalS7GraphByBlock (  
    DWORD          dwFlags,  
    LPCTSTR        lpszTiaPortalProjectPath,  
    LPCTSTR        lpszCpuName,  
    LPCTSTR        lpszBlock,  
    DWORD          dwStepNumber,  
    LPCTSTR        lpszErrorTag,  
    LPCMN_ERROR    lpdmError);
```

Parameters

dwFlags

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

- KOPAPI_FLAG_TIAPORTAL_SUPPRESS_PROGRAM_STATUS=0x0004: The TIA Portal does not go into online mode after the block is opened. If this bit is not set, online mode is started after opening the block.
- KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT 0x00000008L: If this bit is set, the call is canceled when the project is already open and contains changes.
- KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY: When this bit is set to TRUE, the project is opened in write-protected mode in the engineering system of the TIA Portal.

lpszTiaPortalProjectPath

Name of the project file including specification of the absolute path, for example, "D:\TIAProjects\Project1\Project1.ap14"

Please note that within a C script the backslash has to be written using an Escape sequence: FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);

lpszCpuName

Name of the S7 CPU. The name is identical to the station name displayed in the project tree in the TIA Portal.

lpszBlock

Instance name of the S7 Graph block to be displayed.

dwStepNumber

Number of the step to be displayed

dwStepNumber=0 automatically searches for the active step and enables "Track active step" mode.

lpszErrorTag

Name of an internal WinCC tag of the String data type. The error information is returned in lpszErrorTag (Page 2474) when asynchronous functions that do not provide instant results are called.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. The system writes error information to this structure if an error occurs.

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h
kopapi.lib
kopapi.dll

See also

Basics (Page 2450)

3.11.1.6 Example: Integration in a WinCC function (RT Professional)

Description

The example shows the embedding of the function calls in a user-defined function. For better maintainability, the call should be made externally in a global script function, in which the path to the TIA Portal and the name of the PLC station are then determined.

```
void OnClick(char* screenName, char* objectName, char* propertyName)
{
    // Funktionen bekannt machen'
    #pragma code("KOPAPI.dll")
    #include "KOPAPI.h"
    #pragma code()

    BOOL useTiaPortal = TRUE; // use the TIA Portal or the viewer control?
    char* pTiaPortalProject = "c:\\Projects\\myproject\\myproject.apl4";
    char* pStationName = "OW063IZ1M01_PLC2_STD"; // TODO: read from internal
tag
    char* pContainingBlock = "SG01_FG02_ST001+FX001_IDB"; // TODO: get from
current selection
    char* pOperand = "#HOME_SEQ1.HOME_MISC[8]"; // TODO: get from current
selection
    char* pErrorTag = "IECPLViewerErrorTag"; // TODO: define an own tag
    CMN_ERROR error;
    BOOL result;

    if(useTiaPortal)
    {
        result = OpenTIAPortalIECPLByAssignment(0, pTiaPortalProject,
pStationName, pContainingBlock, pOperand, pErrorTag, &error);
    }
    else
    {
        char *pServerPrefix = NULL, *pTagPrefix = NULL, *pWindowPrefix =
NULL;

        // determine ServerPrefix of the current environment
        GetServerTagPrefix(&pServerPrefix, &pTagPrefix, &pWindowPrefix);
        // make the screen which contains the viewer control visible
        SetVisible("SYSTEM#Basic_Screen", "Screen_window_IECPLViewer",
TRUE);
        result = OpenViewerIECPLByAssignment(0, pServerPrefix,
"SYSTEM#IECPLViewer", "IECPLViewerObject", pStationName, pContainingBlock,
pOperand, pErrorTag, &error);
    }
}
```

See also

Basics (Page 2450)

3.11.2 Display in the PLC code display (RT Professional)

3.11.2.1 Basics (RT Professional)

General information

You can use these functions to display the current program status of PLC programs in a PLC code display.

The screen containing the PLC code display must have been opened via `OpenPicture()` or `SetPictureName()`, for example.

Header files

The declaration of the functions and structures explained here takes place in the header files:

kopapi.h	API interface definition header file

Libraries

The DLL link for the functions explained here is provided in the "lib" files: pdecsccli.dll

kopapi.lib	Library
kopapi.dll	

Note on defining multi-instance names

When multi-instance name paths are specified, these are data hierarchies such as those displayed in the DB editor and not call structures. The first part of the name ("Line1") does not need to be enclosed in quotation marks as it is clear from the context that it is a global symbol. Quotation marks are required for the individual name components where there are special characters such as spaces, dots, etc. Example: "Line1.Cell1.Station1". Names that do not contain special characters must not be put in quotation marks.

See also

[OpenViewerIECPLByCall](#) (Page 2466)

[OpenViewerIECPLByAssignment](#) (Page 2472)

[OpenViewerS7GraphByBlock](#) (Page 2463)

3.11.2.2 OpenViewerS7GraphByBlock (RT Professional)

Description

The function displays in the PLC language, S7 Graph, a step called from a sequencer in the PLC code display.

Declaration

```
BOOL OpenViewerS7GraphByBlock (  
    DWORD        dwFlags,  
    LPCTSTR      lpszServerPrefix,  
    LPCTSTR      lpszPictureName,  
    LPCTSTR      lpszObjectName,  
    LPCTSTR      lpszCpuName,  
    LPCTSTR      lpszBlock,  
    DWORD        dwStepNumber,  
    LPCMN_ERROR  lpdmError);
```

Parameter

dwFlags

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

lpszServerPrefix

The parameter is reserved for later upgrades.

lpszPictureName

Name of the screen with the PLC code viewer.

lpszObjectName

Name of the PLC code viewer.

lpszCpuName

Name of the S7 CPU. The name is identical to the station name displayed in the project tree in the TIA Portal.

lpszBlock

Instance name of the S7 Graph block to be displayed. If special characters such as spaces, periods, etc. occur in the name, quotation marks need to be used.

dwStepNumber

Number of the step to be displayed.

dwStepNumber=0 automatically searches for the active step and enables "Track active step" mode.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. The system writes error information to this structure if an error occurs.

Return value**TRUE**

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h

kopapi.lib

kopapi.dll

Example

The example shows the embedding of the function call in a user-defined C function. For better maintainability, the call should be swapped out to a global script function in which the name of the PLC station is also determined.

```

BOOL OpenViewerS7GraphByBlock(char* screenName, char* objectName, char*
cpuName, char* instanceDBName)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
DWORD stepNumber = 0;
result = OpenViewerS7GraphByBlock(dwFlags, serverPrefix, screenName,
objectName,
cpuName, instanceDBName, stepNumber, &error);
if(!result)
{
// there are only few reasons why the call to OpenViewerS7GraphByBlock
will
// fail, in most cases the viewer control could not be found
// most of the errors have to be handled in the OnError event of the viewer
printf("OpenViewerS7GraphByBlock failed: err1=%ld, err2=%ld, err3=%ld,
err4=%ld, err5=%ld, text=\"%s\\\"\\r\\n\", result, error.dwError1,
error.dwError2, error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}

```

See also

Error-handling (Page 2474)

Basics (Page 2461)

3.11.2.3 OpenViewerIECPLByCall (RT Professional)

Description

The function is used for the LAD and FBD languages and shows the preceding logic of a network input of a standard block in the PLC code view.

.

Note

If the input is occupied by a constant (TRUE, FALSE), this value is not displayed in the PLC code viewer. To display the value in the PLC code viewer correctly, you should allocate a tag to the input that has the appropriate value.

Declaration

```
BOOL OpenViewerIECPLByCall (  
    DWORD          dwFlags,  
    LPCTSTR        lpszServerPrefix,  
    LPCTSTR        lpszPictureName,  
    LPCTSTR        lpszObjectName,  
    LPCTSTR        lpszCpuName,  
    LPCTSTR        lpszContainingBlock,  
    LPCTSTR        lpszCalledBlock,  
    LPCTSTR        lpszPin,  
    LPCMN_ERROR    lpdmError);
```

Parameter

dwFlags

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

- IECPLVIEWER_PIN_SUBSTRING_SEARCH=0x0001: A substring is sought in the search for the pin name, which means the pin name starts with the string transferred in lpszPin. If this bit is not set, the complete pin name is compared to lpszPin.

lpszServerPrefix

The parameter is reserved for future expansions.

IpszPictureName

Name of the screen with the PLC code viewer.

IpszObjectName

Name of the PLC code viewer.

IpszCpuName

Name of the S7 CPU. The name is identical to the station name displayed in the project tree in the TIA Portal.

IpszContainingBlock

Name of the block to be opened and displayed or name of the instance of an FB.

The following can be used as the name:

- Name of a single instance DB. Its FB is then displayed. Example "Station1"
- Name of a multi-instance in an instance DB. Its FB is then displayed. When multiinstance name paths are specified, these are data hierarchies such as those displayed in the DB editor, and not those in the call structure. The first part of the name ("Line1") cannot be enclosed in quotes, because this is a global icon, as can be recognized by the context. Quotation marks are required for the individual name components when special characters such as spaces, dots, etc., occur in them. Example: "Line1.Cell1.Station1"
- Name of an FC or OB

The use of the name of a FB is not allowed.

IpszCalledBlock

Name of the local or global instance that is called in the code block belonging to IpszContainingBlock.

- For local instances, the hash sign # must be specified here, for example, "#feeder1".
- For global instances, the global name without hash sign # must be specified here, for example, "feeder3".

The use of the name of an FC is not allowed.

If IpszCalledBlock is called several times within IpszContainingBlock or its FB, the entry point is always the first call of IpszCalledBlock.

IpszPin

Name of the input pin of IpszCalledBlock. The parameter is used to display the network interconnected to the input pin in the PLC code viewer.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. When an error occurs, the system writes error information into this structure.

Return value**TRUE**

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h

kopapi.lib

kopapi.dll

Example

The example shows the embedding of the function call in a user-defined C function. For better maintainability, the call should be swapped out to a global script function in which the name of the PLC station is also determined.

```
BOOL OpenCodeViewerByCall(BOOL matchSubstringPin, char* screenName, char*
objectName, char* cpuName, char* containingBlock, char* calledBlock, char*
pinName)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
if(matchSubstringPin)
    dwFlags |= KOPAPI_FLAG_TIAPORTAL_PIN_SUBSTRING_SEARCH;
result = OpenViewerIECPLByCall(dwFlags, serverPrefix, screenName,
objectName,
    cpuName, containingBlock, calledBlock, pinName, &error);
if(!result)
{
    // there are only few reasons why the call to OpenViewerIECPLByCall will
fail, in // most cases the viewer control could not be found
    // most of the errors have to be handled in the OnError event of the viewer
    printf("OpenViewerIECPLByCall failed: err1=%ld, err2=%ld, err3=%ld, err4=
%ld, err5=%ld, text=\"%s\\r\\n\", result, error.dwError1,
error.dwError2, error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}
```

See also

Error-handling (Page 2474)

Basics (Page 2461)

3.11.2.4 OpenViewerIECPLByFCCall (RT Professional)**Description**

The function is used for the LAD and FBD languages and shows the preceding logic of a network input of a standard block in the PLC code view while taking the UDT instance into consideration.

Note

If the input is occupied by a constant (TRUE, FALSE), this value is not displayed in the PLC code viewer. To display the value in the PLC code viewer correctly, you should allocate a tag to the input that has the appropriate value.

Declaration

```

BOOL OpenViewerIECPLByFCCall (
    DWORD          dwFlags,
    LPCTSTR        lpszServerPrefix,
    LPCTSTR        lpszPictureName,
    LPCTSTR        lpszObjectName,
    LPCTSTR        lpszCpuName,
    LPCTSTR        lpszContainingBlock,
    LPCTSTR        lpszCalledBlock,
    LPCTSTR        lpszPin,
    LPCTSTR        lpszUdtInstance
    LPCMN_ERROR    lpdmError);

```

Parameters**dwFlags**

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

- IECPLVIEWER_PIN_SUBSTRING_SEARCH=0x0001: A substring is sought in the search for the pin name, which means the pin name starts with the string transferred in lpszPin. If this bit is not set, the complete pin name is compared to lpszPin.

lpszServerPrefix

The parameter is reserved for future development and must be an empty string ("").

lpszPictureName

Name of the screen with the PLC code viewer.

lpszObjectName

Name of the PLC code viewer.

lpszCpuName

Name of the S7 CPU. The name is identical to the station name displayed in the project tree in the TIA Portal.

lpszContainingBlock

Name of the block to be opened and displayed or name of the instance of an FB.

The following can be used as the name:

- Name a single instance DB. Its FB is then displayed. Example "Station1"
- Name of a multiinstance in an instance DB. Its FB is then displayed. When multiinstance name paths are specified, these are data hierarchies such as those displayed in the DB editor, and not those in the call structure. The first part of the name ("Line1") cannot be enclosed in quotes, because this is a global icon, as can be recognized by the context. Quotation marks are required for the individual name components, when special characters such as spaces, dots, etc., occur in it. Example: "Line1.Cell1.Station1"
- Name of an FC or OB

The use of the name of a FB is not allowed.

lpszCalledBlock

Name of the local or global instance that is called in the code block belonging to lpszContainingBlock.

- For local instances, the hash sign # must be specified here, for example, "#feeder1".
- For global instances, the global name without hash sign # must be specified here, for example, "feeder3".

The use of the name of an FC is not allowed.

If lpszCalledBlock is called several times within lpszContainingBlock or its FB, the entry point is always the first call of lpszCalledBlock.

lpszPin

Name of the input pin of lpszCalledBlock. The parameter is used to display the network interconnected to the input pin in the PLC code viewer.

lpszUdtInstance

The parameter is used to limit the display of FBs or FCs called multiple times. The limitation is based on an UDT instance interconnected to a given input pin or inout pin.

lpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. The system writes error information to this structure if an error occurs.

Return value**TRUE**

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h

kopapi.lib

kopapi.dll

Example

The example shows the embedding of the function call in a user-defined C function. For better maintainability, the call should be swapped out to a global script function in which the name of the PLC station is also determined.

```

BOOL OpenCodeViewerByFCCall(BOOL matchSubstringPin, char* screenName,
char* objectName, char* cpuName, char* containingBlock, char* calledBlock,
char* pinName, char* udtInstance)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
if(matchSubstringPin)
    dwFlags |= KOPAPI_FLAG_TIAPORTAL_PIN_SUBSTRING_SEARCH;
result = OpenViewerIECPLByFCCall(dwFlags, serverPrefix, screenName,
objectName, cpuName, containingBlock, calledBlock, pinName, udtInstance,
&error);
if(!result)
{
    // there are only few reasons why the call to OpenViewerIECPLByFCCall will
    fail,
    // in most cases the viewer control could not be found
    // most of the errors have to be handled in the OnError event of the viewer
    control
    printf("OpenViewerIECPLByFCCall failed: err1=%ld, err2=%ld, "" err3=%ld,
err4=%ld, err5=%ld, text=\"%s\"\\r\\n", result, error.dwError1,
error.dwError2, error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}

```

3.11.2.5 OpenViewerIECPLByAssignment (RT Professional)

Description

The function is used for the LAD and FBD languages and shows the assignment of an operand and its preceding logic in the PLC code view.

Declaration

```
BOOL OpenViewerIECPLByAssignment (  
    DWORD      dwFlags,  
    LPCTSTR    lpszServerPrefix,  
    LPCTSTR    lpszPictureName,  
    LPCTSTR    lpszObjectName,  
    LPCTSTR    lpszCpuName,  
    LPCTSTR    lpszContainingBlock,  
    LPCTSTR    lpszOperand,  
    LPCMN_ERROR lpdwError);
```

Parameter

dwFlags

Bit array in which the individual values are ORed bit-by-bit. dwFlags should be 0 by default.

lpszServerPrefix

The parameter is reserved for later upgrades.

lpszPictureName

Name of the screen with the PLC code viewer.

lpszObjectName

Name of the PLC code viewer.

lpszCpuName

Name of the S7 CPU. The name is identical to the station name displayed in the project tree in the TIA Portal.

lpszContainingBlock

Name of the block to be opened and displayed or name of the instance of an FB.

The following can be used as names:

- Name a single instance DB. Its FB is then displayed. Example "Station1"
- Name of a multiinstance in an instance DB. Its FB is then displayed. When multiinstance name paths are specified, these are data hierarchies such as those displayed in the DB editor, and not those in the call structure. The first part of the name ("Line1") cannot be enclosed in quotes, because this is a global icon, as can be recognized by the context. Quotation marks are required for the individual name components, when special characters such as spaces, dots, etc., occur in it. Example: "Line1.Cell1.Station1"
- Name of an FC or OB

The use of the name of a FB is not allowed.

IpszOperand

Name of a local or global operand to which an assignment is made.

Name of the local or global instance that is called in the code block belonging to IpszContainingBlock.

- For local operands, the hash sign # must be specified here.
- For global operands, the global name without hash sign # must be specified.

If IpszOperand is written several times within IpszContainingBlock or its FB, the entry point is always the first write access of IpszOperand.

IpdmError

Pointer to the data of the extended error message in the CMN_ERROR structure. The system writes error information to this structure if an error occurs.

Return value

TRUE

The function has been completed without any errors.

FALSE

An error has occurred.

Required files

kopapi.h

kopapi.lib

kopapi.dll

Example

The example shows the embedding of the function call in a user-defined C function. For better maintainability, the call should be swapped out to a global script function in which the name of the PLC station is also determined.

```

BOOL OpenCodeViewerByAssignment(char* screenName, char* objectName, char*
cpuName, char* containingBlock, char* operand)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
result = OpenViewerIECPLByAssignment(dwFlags, serverPrefix, screenName,
objectName, cpuName, containingBlock, operand, &error);
if(!result)
{
    // there are only few reasons why the call to OpenViewerIECPLByAssignment
    will
    // fail, in most cases the viewer control could not be found
    // most of the errors have to be handled in the OnError event of the viewer
    printf("OpenViewerIECPLByAssignment failed: err1=%ld, err2=%ld, err3=%ld,
        err4=%ld, err5=%ld, text=\"%s\"\\r\\n", result, error.dwError1,
error.dwError2, error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}

```

See also

Error-handling (Page 2474)

Basics (Page 2461)

3.11.3 Error-handling (RT Professional)

General information

Execution of the functions for the network entry is divided in a synchronous part and an asynchronous part. The synchronous part checks the parameters and transfers them to the asynchronous part.

The return value of the functions specifies whether an error has occurred in the asynchronous section of the function. If an error has occurred, the return value is FALSE. Such errors occur when parameters, for example, the `lpszErrorTag` parameter is not supplied or incorrectly supplied.

Errors that occur in the asynchronous functions are reported through the `IpszErrorTag` parameter. Such errors could be, for example: Project not found, block not available, ... `IpszErrorTag` contains the name of a tag of the String data type. In order to respond to the return value, you have to configure a function that is triggered with the cycle "To change" at a value change of the error tags. The error tag should also be used to prevent further function calls or the operation of a button, for example, as long as the error tag has the value "RUNNING".

IpszErrorTag

Name of a tag of the String data type. If the tag is not required, NULL can be transferred as the parameter. If required, the tag name can include a ServerPrefix.

The value of the error tag is changed as follows:

- When the asynchronous part is started, the value of the error tag is set to "RUNNING".
- When the asynchronous part is completed without an error, the value of the error tag is set to "OK".
- When the asynchronous part was terminated with an error, the error tag contains a multi-line string whose lines are separated by a linefeed character ('\n').

In the case of an error, the error tag has the following structure:

- Line 1: "ERROR"
- Line 2 - Line 6: Decimal numbers, data type: 32 bits unsigned; DWORD
- Line 7: Error text

Error text

When the asynchronous part was terminated with an error, the error tag contains a multi-line string whose lines are separated by a linefeed character ('\n'). The seventh line contains one of the following error texts:

IDS_E_IS_TIA_PROJECT	KOPAPI: This function cannot be used in a TIA Portal project!
IDS_E_NO_TIA_PROJECT	KOPAPI: The function call is disabled because no TIA Portal project is open!
IDS_E_TIAPORTAL_UNKNOWN_FLAGS	dwFlags contains an unknown value.
IDS_E_TIAPORTAL_ERRORTAG_NOT_EXIST	The error tag [%s] does not exist.
IDS_E_TIAPORTAL_PREVIOUS_CALL_IS_RUNNING	The previous call is still running (RUNNING). Please terminate it first.
IDS_E_TIAPORTAL_CANNOT_WRITE_ERRORTAG	The error tag [%s] cannot be written.
IDS_E_TIAPORTAL_COMACCESS_REGIS-TERPS_FAILED	No connection to TIA Portal possible.
IDS_E_TIAPORTAL_CANNOT_START_PORTAL	TIA Portal cannot be started.
IDS_E_TIAPORTAL_CANNOT_SEARCH_STAR-TED_PORTAL	A search for a started Portal is not possible.

3.12 Functions for displaying the PLC code in an external application (RT Professional)

IDS_E_TIAPORTAL_NO_PORTAL_STARTED	A Portal could not be started.
IDS_E_TIAPORTAL_EXCEPTION_SYNC_PART	Exception in synchronous processing part.
IDS_E_TIAPORTAL_EXCEPTION_ASYNC_PART	Exception in asynchronous processing part.
IDS_E_TIAPORTAL_NOT_INSTALLED	No TIA Portal installed.
IDS_E_TIAPORTAL_PROJECT_CANNOT_OPEN	Project [%s] cannot be opened.
IDS_E_TIAPORTAL_ALREADY_OPENED_WITH_OTHER_PROJECT	TIA Portal is opened with another project.
IDS_E_TIAPORTAL_CREATECOMMAND	CreateCommand error.
IDS_E_TIAPORTAL_COMMAND_ADDARGUMENT	Command AddArgument error.
IDS_E_TIAPORTAL_EXECUTECOMMAND	Serious error in ExecuteCommand.
IDS_E_TIAPORTAL_COMMAND_ERROR	TIA project command error: [%s,%ld].
IDS_E_TIAPORTAL_COMMAND_UNKNOWN	TIA project unknown command: [%s,%ld].

3.12 Functions for displaying the PLC code in an external application (RT Professional)

3.12.1 Basics (RT Professional)

Introduction

WinCC provides a library that allows you to display PLC code from Runtime in an external application. Information about networks or values pending at block interfaces is returned in "XML" format.

The FBD and LAD programming languages are supported. The table below shows the instructions supported by the respective programming language:

Instruction	FBD	LAD
Contact	No	Yes
NegContact	No	Yes
Coil	Yes	Yes
NegCoil	Yes	Yes
RCoil	Yes	Yes
SCoil	Yes	Yes
PinCoil	Yes	Yes
OR	Yes	Yes
AND	Yes	No
XOR	Yes	No
NOT	No	Yes

If other instructions are contained in networks, an error is returned in XML in the section "NetworkHeaderInfo".

Recommended development environment

Siemens recommends using the following development environment:

- MS Visual Studio 2012
- Technology: COM object

Requirements

The following requirements apply in order to be able to display PLC code using the application programmed in MS Visual Studio:

- The application is on the PC on which Runtime has been started.
- An HMI connection between HMI and PLC is configured.
- The PLC is in RUN - required for querying dynamic data.

Procedure

Querying data is carried out in three steps:

1. Calling static information
In this step, you use the "RequestByAssignment" or "RequestByCall" functions to obtain information about available networks
2. Dynamic information
In this step, you call information about the current flow or pending states for one of the determined networks.
3. Responding to events (optional)
State changes in the PLC
 - Change to the user program
 - Connection to PLC interrupted
 - Connection to PLC restored

These state changes are provided via the interface "IPlcCodeViewerExternalClientCB" using the "OnEvent" method:

```
HRESULT OnEvent( [in] PlcCodeViewerExternalClientEvent event );
```

Implementation in MS Visual Studio 2012

Implement the functions for accessing the interface "IPlcCodeViewerExternalClientAccess" in the following order:

1. Initialize
2. SetLanguage
3. RequestByAssignment | RequestByCall
4. CancelRequest
5. Terminate

See also

Structure of the XML data (Page 2484)

3.12.2 Querying static information of networks (RT Professional)

3.12.2.1 Initialize (RT Professional)

Description

The function initializes the COM object.

Declaration

```
HRESULT Initialize( [in] IPlcCodeViewerExternalClientCB* pCallback,  
                   [in] BSTR serverString);
```

Parameters

IPlcCodeViewerExternalClientCB

Implements the interface which provides the methods for querying network information.

serverString

Name of the PC on which Runtime is running.

Return value

-

See also

Terminate (Page 2483)

3.12.2.2 SetLanguage (RT Professional)

Description

The function defines the language in which comments of tags of a network are output in the XML file.

Declaration

```
HRESULT SetLanguage( [in] BSTR language);
```

Parameters

language

Defines the language. Use the "Culture Name" as notation from the following reference: National Language Support (NLS) API Reference (<https://msdn.microsoft.com/de-de/global/bb896001.aspx>)

If there are no comments in the specified language, comments are output in one of the following languages:

1. German (de-DE)
2. English (en-US or en-GB)
3. The available language

If you do not define a language, the comments are output in all available languages.

Return value

-

3.12.2.3 RequestByAssignment (RT Professional)

Description

The function shows the assignment to an operand and its preceding logic. All networks in which the operand is written are output.

Note

The call of this function cancels the following actions:

- Not yet completed calls of "RequestByAssignment" or "RequestByCall"
 - Running subscriptions.
-

Declaration

```
HRESULT RequestByAssignment( [in] BSTR PLCName,  
                             [in] BSTR ContainingBlock,  
                             [in] BSTR OperandName,  
                             [out] long* RequestId);
```

Parameters

PLCName

Name of the PLC whose user program contains the block. The name is identical to the station name displayed in the project tree in the TIA Portal.

ContainingBlock

Name of the block to be displayed or name of the instance of an FB.

The following can be used as names:

- Name of a single instance DB. Its FB is then displayed. Example "Station1"
- Name of a multi-instance in an instance DB. Its FB is then displayed. When multi-instance name paths are specified, these are data hierarchies such as those displayed in the DB editor, and not the call structure. The first part of the name ("Line1") does not have to be enclosed in quotation marks, because it is clear from the context that this is a global icon. Quotation marks are required for the individual name components, when special characters such as spaces, dots, etc., occur in it. Example: "Line1.Cell1.Station1"
- Name of an FC or OB

OperandName

Name of the block to be searched for in the code block belonging to "ContainingBlock".

- For local operands, the hash sign # must be specified here.

Return value

ID with which the request can be clearly identified. The ID is generated and returned with the call of the function.

See also

[RequestByCall \(Page 2481\)](#)

[StartDynamicDataSubscription \(Page 2487\)](#)

[CancelRequest \(Page 2482\)](#)

3.12.2.4 RequestByCall (RT Professional)

Description

The function displays the preceding logic of a network input of a standard block. All networks in which the input pin is contained are returned.

Note

Calling this function cancels the following actions:

- Not yet completed calls of "RequestByAssignment" or "RequestByCall"
 - Current subscriptions.
-

Declaration

```
HRESULT RequestByCall( [in] BSTR PLCName,  
                      [in] BSTR ContainingBlock,  
                      [in] BSTR CalledBlock,  
                      [in] BSTR PinName,  
                      [in] boolean PartialPinNameMatch,  
                      [out] long* RequestId);
```

Parameters

PLCName

Name of the PLC with the user program in which the block is contained. The name is identical to the station name displayed in the project tree in the TIA Portal.

ContainingBlock

Name of the block to be displayed or name of the instance of an FB.

CalledBlock

Name of the FB or FC which is called in the network of "ContainingBlock". Alternatively, you can also configure a DB or a multi-instance.

The following can be used as names:

- Name of a single instance DB. Its FB is then displayed. Example "Station1"
- Name of a multi-instance in an instance DB. Its FB is then displayed. When multi-instance name paths are specified, these are data hierarchies such as those displayed in the DB editor, and not the call structure. The first part of the name ("Line1") does not have to be enclosed in quotation marks, because it is clear from the context that this is a global icon. Quotation marks are required for the individual name components when special characters such as spaces, dots, etc., occur in it. Example: "Line1.Cell1.Station1"
- Name of an FC or OB

PinName

Name of an input pin of "CalledBlock". The parameter is used to ensure that the network interconnected to the input pin is contained in the XML data.

PartialPinNameMatch

TRUE if input pins which begin with the name specified under "PinName" are also to be taken into consideration.

Return value

ID with which the request is uniquely identified. The ID is generated and returned when the function is called.

See also

[RequestByAssignment \(Page 2477\)](#)

[StartDynamicDataSubscription \(Page 2487\)](#)

[CancelRequest \(Page 2482\)](#)

3.12.2.5 CancelRequest (RT Professional)

Description

The function cancels a request.

Declaration

```
HRESULT CancelRequest( [in] long RequestId);
```

Parameters

RequestId

ID of the request which is canceled.

Return value

-

See also

[RequestByAssignment \(Page 2477\)](#)

[RequestByCall \(Page 2479\)](#)

3.12.2.6 Terminate (RT Professional)

Description

The function ends the execution of a COM object implemented with "Initialize":

- All active requests and subscriptions are ended.
- The callback functions "OnModeXML", "OnDynamicDataXML" and "OnEvent" are ended.

Declaration

```
HRESULT Terminate();
```

Parameters

-

Return value

-

See also

Initialize (Page 2476)

3.12.2.7 Structure of the XML data (RT Professional)

Structure

```
<PlcCodeViewerExternalClientModel Version="1.0.0.0">
  <CommonHeaderInfo>
  </CommonHeaderInfo>
  <Networks>
    <Networks>
      <NetworkHeaderInfo>
      <NetworkHeaderInfo>
      <FlgNet>
        <Parts>
          <Part>
            <Parameters>
            <Parameters>
          </Part>
        </Parts>
        <Wires>
          <Wire>
          </Wire>
        </Wires>
      </FlgNet>
    </Networks>
  </Networks>
</PlcCodeViewerExternalClientModel>
```

The XSD file can be found by default in the following directory: C:\Users\Public\Documents\Siemens\WinCC.

"<PlcCodeViewerExternalClientModel>" section

Contained: [1]

Contains the sections "CommonHeaderInfo" and "Networks".

```
<PlcCodeViewerExternalClientModel Version="1.0.0.0">
  <CommonHeaderInfo>
    <HeaderInfo Description="PlcName" Text="Plc_1"/>
  </CommonHeaderInfo>
  <Networks>
    ...
  </Networks>
</PlcCodeViewerExternalClientModel>
```

"<Networks>" section

Contained: [0..n]

Represents a network which is uniquely identified via the attribute "NetId". Contains the sections "NetworkHeaderInfo" and "FlgNet".

```

<Networks>
  <Network NetId="1">
    <NetworkHeaderInfo>
      <HeaderInfo Description="FBNum" Text="3"/>
      <HeaderInfo Description="FBName" Text="Block1"/>
      <HeaderInfo Description="Title">
        <DictEntry Language="DE-EN">First part of
Headers</DictEntry>
      </HeaderInfo>
      <HeaderInfo>
        <DictEntry Language="DE-EN">Second part of
Headers</DictEntry>
      </HeaderInfo>
    </NetworkHeaderInfo>

    <FlgNet Version="1.0.0.0" Lang="LAD">
      <Parts>
        ...
      </Parts>
      <Wires>
        ...
      </Wires>
    </FlgNet>
  </Network>
</Networks>

```

"<NetworkHeaderInfo>" section

Contained: [1]

Contains information about the network, e.g. the name of the instruction or the network title.

"<FlgNet>" section

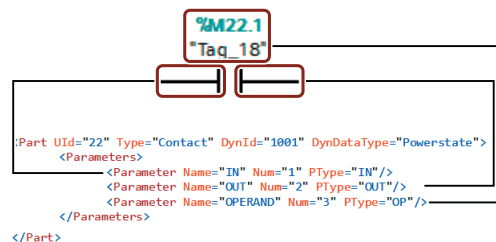
Contained: [1]

Represents the description of the network. Contains the sections "Parts" and "Wires".

"<Part>" section

Contained: [0..n]

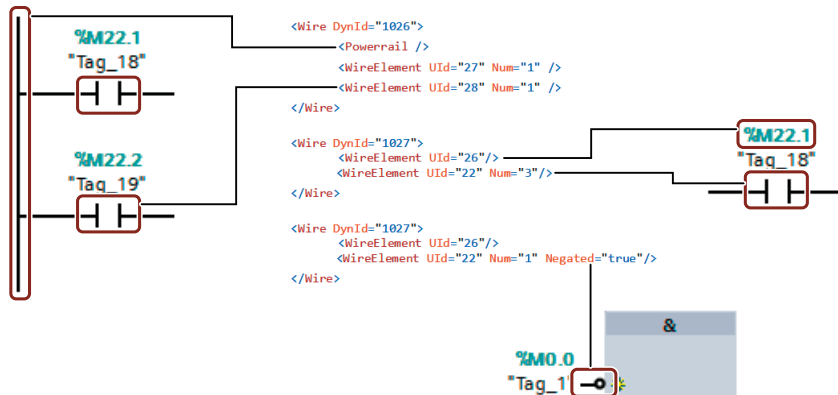
Represents an instruction or an operand.



"<Wire>" section

Contained: [0..n]

Represents networking between instructions and operands.

**Relevant attributes**

The following table shows the attributes of the XML data.

Attribute	Description	Contained in
AbsAdress	Absolute address	Operand
DynData-Type	Data type if dynamic information is retrieved. Depending on object type: - Operand - Bool: "True" or "False" - Part - Powerflow - State of the current flow: "On", "Off" or "Undef" - Bool: "True" or "False" - State of a parameter or an instruction from: - Wire - Powerstate	Operand Part Wire
DynID	References the object for querying its dynamic data with the function "StartDynamicDataSubscription"	Operand Part Wire
Name	Object name. With operands, the symbolic address is displayed.	Operand Parameter
Negated	True if the input is negated.	
Num	References a Parameter.	Parameter
PType	Defines whether a pin is an input or output or represents the switching state.	Parameter

3.12 Functions for displaying the PLC code in an external application (RT Professional)

Attribute	Description	Contained in
Type	Defines the type.	
UId	References the object Id used to describe relationships such as networks between objects.	Operand Part Wire WireElement

3.12.3 Querying dynamic statistical information of networks (RT Professional)

3.12.3.1 StartDynamicDataSubscription (RT Professional)

Description

The function queries cyclic information for the specified network. The information is integrated into the XML data which you queried beforehand with "RequestByAssignment" or "RequestByCall".

If you transfer the value "0" for DynIDCount or "DynIDs", only information about operands and instructions is retrieved.

Declaration

```
HRESULT StartDynamicDataSubscription( [in] int NetworkID,
                                       [in] long DynIDCount,
                                       [in, [size_is(DynIDCount)]]
                                       long* DynIDs,
                                       [out] long* SubscriptionId);
```

Parameters

NetworkID

ID of the network whose static data was queried previously. The ID is available in the XML data in the attribute "NetID".

DynIDCount

Number of areas for which the dynamic information is queried.

DynIDs

IDs of pins or inputs for which current flows or current values are displayed. The IDs are available in the XML data in the attribute "DynID".

Return value

ID with which the subscription is uniquely identified. The ID is generated and returned with the call of the function.

See also

[RequestByAssignment \(Page 2477\)](#)

[RequestByCall \(Page 2479\)](#)

3.12.3.2 StopDynamicSubscription (RT Professional)

Description

The function ends a current subscription.

Declaration

```
HRESULT StopDynamicSubscription( [in] long SubscriptionId);
```

Parameters

SubscriptionId

ID of the subscription which is ended.

Return value

-

3.13 Troubleshooting (RT Professional)

3.13.1 CMN_ERROR (RT Professional)

Description

The extended error structure `CMN_Error` contains the error code and an error text for the error that occurred. Each application can use the error structure to evaluate or to output error messages.

Declaration

```
Typedef struct {  
    DWORD    dwError1;  
    DWORD    dwError2;  
    DWORD    dwError3;  
    DWORD    dwError4;  
    DWORD    dwError5;  
    Char      szErrorText[512];  
}  
CMN_ERROR;
```

Members

dwError1 .. dwError5

Each of the API descriptions contains information on the values of the entries in case of an error. Unless otherwise specified, the error codes are in dwError1.

szErrorText

Buffer for the text description of the error cause. The content is determined from the resources and is generally language-dependent.

Comment

Some modules use the following error code assignments, as required

dwError1	API error code
dwError2	Subordinate error code (e.g., CCStorageError.h)
dwError3	Line number of the source code
dwError4	Array Index 1; Row; Record / Status /
dwError5	Array Index 2; Column; Record Element / Counter /

See also

Error messages (Page 1556)

3.13.2 CCStorageError.h (RT Professional)

Subordinate error messages CCStorageError.h

The following error messages can be returned by the API functions in the CMN_ERROR error structure in addition to the actual error message in dwError1 and in dwError2, if available in the respective module.

Error message	Value	
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN	0x80046101L	The WinCC project cannot be opened.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_1	0x80046102L	The WinCC project cannot be opened/created. Project Manager cannot start the SDIAGRT basic application.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_2	0x80046103L	The WinCC project cannot be opened/created. Project Manager cannot start the PASSDBRTT basic application.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_3	0x80046104L	The WinCC project cannot be opened/created. Project Manager cannot start the SCRIPT basic application.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_31	0x80046105L	The WinCC project cannot be opened/created. Project Manager cannot start the XREF basic application.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_4	0x80046106L	The WinCC project cannot be opened/created. Project Manager cannot initialize the ASO basic application.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_5	0x80046107L	The WinCC project cannot be opened/created. Project Manager can only open one WinCC project at a time.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_6	0x80046108L	The WinCC project cannot be opened/created. Project Manager cannot configure a start line for the data source.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_7	0x80046109L	The WinCC project cannot be opened/created. Project Manager cannot configure a data source.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_8	0x8004610AL	The WinCC project cannot be opened. Project Manager for the configured WinCC server not running yet. Start the Project Manager for the configured server.

Error message	Value	
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_9	0x8004610BL	The WinCC project cannot be opened/created. Project Manager cannot create the Distributed Configuration file (Project.dcf).
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_10	0x8004610CL	The WinCC project cannot be opened. The project path is invalid.
CCF_STOR-AGE_PM_E_NO_PROJECT_CLOSE_1	0x8004610DL	The WinCC project cannot be closed. Some users are already connected to the WinCC project via the network.
CCF_STOR-AGE_PM_E_NO_PROJECT_CLOSE_2	0x8004610EL	The WinCC project cannot be closed. Project Manager is missing specific information about this WinCC project. Please check your project ID.
CCF_STOR-AGE_PM_E_NO_PROJECT_CLOSE_3	0x8004610FL	The WinCC project cannot be closed. An active WinCC project is already running on your PC.
CCF_STOR-AGE_PM_E_NO_PROJECT_CLOSE_4	0x80046110L	The WinCC project cannot be closed. Project Manager does not support the close mode of the project.
CCF_STOR-AGE_PM_E_NO_PROJECT_CLOSE_5	0x80046111L	The WinCC project cannot be closed. Some users are already connected to the WinCC project. Timeout!
CCF_STORAGE_PM_E_NO_PROJECT_LANGUAGE_1	0x80046112L	The project does not support the requested language.
CCF_STORAGE_PM_E_NO_ACTIVATE	0x80046113L	No active WinCC projects running on this PC.
CCF_STORAGE_PM_E_NO_ACTIVATE_1	0x80046114L	The WinCC project cannot be activated. The WinCC server properties have been changed. Close and open the WinCC project again.
CCF_STORAGE_PM_E_NO_PROJECT_CREATE	0x80046115L	The WinCC project cannot be created.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_11	0x80046116L	The WinCC project cannot be opened. Close the running WinCC project.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_12	0x80046117L	The WinCC project cannot be opened. The WinCC project path exceeds the maximum permissible length of 255 characters.
CCF_STOR-AGE_PM_E_NO_PROJECT_OPEN_13	0x80046118L	The WinCC project cannot be opened. The WinCC project name exceeds the maximum permissible length of 64 characters.

3.13 Troubleshooting (RT Professional)

Error message	Value	
CCF_STORAGE_PM_E_NO_ACTIVATE_2	0x80046119L	The WinCC project cannot be deactivated. The timeout interval has expired (SCRIPT.exe).
CCF_STORAGE_PM_E_NO_ACTIVATE_3	0x8004611AL	The WinCC project cannot be deactivated. The timeout interval has expired (PASSDBRT.exe).
CCF_STORAGE_PM_E_NO_ACTIVATE_4	0x8004611BL	The WinCC project cannot be deactivated. The timeout interval has expired (SDIAGRT.exe).
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_14	0x8004611CL	The WinCC project cannot be opened. The project path could not be inserted in the SIMATIC.cfg file.
CCF_STORAGE_PM_E_NO_ACTIVATE_5	0x8004611DL	The WinCC project cannot be activated. No WinCC project exists.
CCF_STORAGE_PM_E_NO_ACTIVATE_6	0x8004611EL	The WinCC project cannot be opened. The WinCC client cannot receive the server activation status over broadcast.
CCF_STORAGE_PM_E_NO_ACTIVATE_7	0x8004611FL	The activation of the WinCC project is running. The activation of the WinCC client is not yet complete.
CCF_STORAGE_PM_E_CREATE_SHARE_FAILED	0x80046120L	The WinCC project share cannot be created. The WinCC project could not be opened.
CCF_STORAGE_PM_E_OPEN_FILE_FAILED	0x80046121L	The WinCC project file cannot be opened. The WinCC project cannot be opened.
CCF_STORAGE_PM_E_CLIENT_CANNOT_ACTIVATE	0x80046122L	The client cannot activate this project because it is not included in the client list.
CCF_STORAGE_PM_E_SYBASE_MIGRATION	0x80046123L	This WinCC version does not support the old Sybase projects. Migrate your projects.
CCF_STORAGE_PM_E_NO_PROJECT_OPENED	0x80046124L	No project is currently open.
CCF_STORAGE_PM_E_COMPUTER_NOT_IN_MACHINE_LIST	0x80046125L	This client is not included in the list of specified servers.
CCF_SQL_SERVER_NOT_INSTALLED_ERROR	0x80046126L	The SQL Server may have been installed incorrectly. Check whether the SQL Server was installed correctly.
CCF_SQL_SERVER_DATABASE_ACCESS_ERROR	0x80046127L	The SQL Server cannot access the database. Check access rights.
CCF_PM_E_PROJECT_IS_LOCKED	0x80046128L	The project is locked.

Error message	Value	
CCF_SQL_SERVER_DATABASE_NTFS_ERROR	0x80046129L	The SQL Server 2005 does not support compressed NTFS volumes. Decompress at least the project directory and its content.
CCF_PM_OPEN_ERROR_RDP	0xC004612AL	This project is generated by RDP. It is not permitted to open it in WinCC.
CCF_PM_OPEN_CODEPAGE_MISMATCH	0x4004612BL	This code page does not match the code page at the time of creation.
CCF_PM_MIGRATION_NEEDED	0x8004612CL	PM migration required.
CCF_STORAGE_DBASO_E_NO_LICENSE	0x80046201L	No license is available for this functionality.
CCF_STORAGE_DBASO_E_LIMIT	0x80046202L	The high limit of the storage capacity has been reached.
CCF_STORAGE_DBASO_E_DATA_VERSION	0x80046203L	ASO cannot work with this file version.
CCF_STORAGE_DBASO_E_NO_CONNECTION	0x80046204L	No connection to the database.
CCF_STORAGE_DBASO_E_NO_TABLE	0x80046205L	The table does not exist or cannot be opened. Check access rights.
CCF_STORAGE_DBASO_E_NO_RECORD	0x80046206L	No data record
CCF_STORAGE_DBASO_E_NO_RESOURCE	0x80046207L	No resource
CCF_STORAGE_DBASO_E_NOT_CREATED	0x80046208L	The data record was not created.
CCF_STORAGE_DBASO_E_NOT_DELETED	0x80046209L	The data record was not deleted.
CCF_STORAGE_DBASO_E_NOT_MODIFIED	0x8004620AL	The data record was not changed.
CCF_STORAGE_DBASO_E_RECORD_EXISTS	0x8004620BL	The data record already exists.
CCF_STORAGE_DBASO_E_RECORD_IN_USE	0x8004620CL	The data record is already in use.
CCF_STORAGE_DBASO_E_RECORD_FIELDSINVALID	0x8004620DL	Invalid count or field type
CCF_STORAGE_DBASO_E_RECORD_PROTECTED	0x8004620EL	The data record cannot be changed or moved.
CCF_STORAGE_DBASO_E_NO_CONTEXT	0x8004620FL	No or invalid project context. ASO does not function correctly.
CCF_STORAGE_DBASO_E_ENUM_NOENTRIES	0x80046210L	No entries found for enumeration.
CCF_STORAGE_DBASO_E_NOT_FOUND	0x80046211L	Could not find the specified object.
CCF_STORAGE_DBASO_E_ID_CHANGES_NOT_POSSIBLE	0x80046212L	You are not allowed to change the IDs.
CCF_STORAGE_DBASO_E_PARAMETER_TOO_LONG	0x80046213L	The parameter is too long.

3.13 Troubleshooting (RT Professional)

Error message	Value	
CCF_STORAGE_DBASO_E_DATA_TOO_OLD	0x80046214L	The data could not be read because it originates from an older WinCC version.
CCF_STORAGE_DBASO_E_DATA_NEWER_THAN_SOFTWARE	0x80046215L	The data comes from a more recent version than the installed software. Please upgrade your software.
CCF_STORAGE_DBASO_E_ENUMSINK_RETURNED_WITH_ERROR	0x80046216L	Enum sink executed with errors.
CCF_STORAGE_DBASO_E_DATA_EXIST_IN_REFERENCED_TABLE	0x80046217L	The primary key is referenced in a different table.
CCF_STORAGE_DBASO_E_DATA_OUT_OF_RANGE	0x80046218L	The value lies outside the value range.
CCF_STORAGE_DM_ASO_BULK_FAILED	0x80046219L	The bulk process failed.
CCF_STORAGE_DBASO_E_RT_LIMIT_WARNING	0x8004621AL	The runtime limit has been reached.
CCF_STORAGE_DBASO_E_BAD_LOCAL_ID	0x8004621BL	Bad local ID
CCF_STORAGE_FILEASO_E_NO_PICTURE	0x80046301L	No picture available
CCF_STORAGE_GDO_BADTYPE_IN_VARIANT	0x80046405L	Incorrect type in the variant
CCF_STORAGE_GDO_DATAINCONSISTENT	0x80046406L	Inconsistent data: Some data is invalid or is outside the value range.
CCF_STORAGE_GDO_UNITID_INVALID	0x80046407L	Inconsistent data: The ID of the channel unit is invalid.
CCF_STORAGE_GDO_USERTYPE_INVALID	0x80046408L	Inconsistent data: The user type is invalid.
CCF_STORAGE_GDO_ADDRESSBUILD_FAILURE	0x80046409L	Inconsistent data Addressing of tags was formed incorrectly.
CCF_STORAGE_GDO_MODIFY_NOCHANGE	0x8004640AL	Inconsistent data: No relevant changes were made.
CCF_STORAGE_GDO_CHANNELWRAPPER_LOAD	0x8004640BL	CCMfcChannelWrapper.DLL could not be found.
CCF_STORAGE_GDO_CHANNEL_LOAD	0x8004640CL	Error while loading the channel: Channel not found.
CCF_STORAGE_GDO_CHANNEL_FUNC	0x8004640DL	Channel access point not found.
CCF_STORAGE_GDO_CHANNEL_UNITNR	0x8004640EL	The number of the channel unit is invalid.
CCF_STORAGE_GDO_CONNECTION	0x8004640FL	The connection is invalid or could not be found.
CCF_STORAGE_GDO_TAGGROUP	0x80046410L	The tag group is invalid or could not be found.
CCF_STORAGE_GDO_NAMENOTFOUND	0x80046411L	The specified name could not be found.

Error message	Value	
CCF_STORAGE_GDO_TAGOFUSERTY-PEEXISTS	0x80046412L	The "UserType" object cannot be moved.
CCF_STORAGE_GDO_PROJECTISACTIVE	0x80046413L	The project was activated. No changes or moves permitted.
CCF_STORAGE_GDO_UNIT	0x80046414L	The channel unit is invalid or could not be found.
CCF_STORAGE_GDO_MINMAX_LIMIT	0x80046415L	The low or high limit was exceeded.
CCF_STORAGE_GDO_PROJECT_IN_RUNTIME	0x80046416L	This process is not permitted while the project is in runtime.
CCF_STORAGE_GDO_BAD_POINTER	0x80046600L	The NULL pointer is not permitted for GetProperty.
CCF_STORAGE_GDO_UNEXPECTED	0x80046601L	An unexpected error has occurred.
CCF_STORAGE_GDO_NO_RECORD	0x80046602L	No data record exists.
CCF_STORAGE_GDO_OPERATION_NOT_COMMITTED	0x80046603L	The current process is not permitted.
CCF_STORAGE_GDO_NO_PROJECT_CONNECTOR	0x80046604L	No Project Connector available.
CCF_STORAGE_GDO_PROJECT_NOT_OPENED	0x80046605L	The project was not opened.
CCF_STORAGE_GDO_GET_ASO_FAILED	0x80046606L	"GetAso" failed.
CCF_STORAGE_GDO_BAD_TYPEID	0x80046607L	The type ID is unsuitable.
CCF_STORAGE_GDO_OTHER_TYPE_NOT_COMMITTED	0x80046608L	Another type is not permitted.
CCF_STORAGE_GDO_CACHE_READ	0x80046609L	Error reading the cache.
CCF_STORAGE_GDO_CACHE_WRITE	0x8004660AL	Error writing to the cache.
CCF_STORAGE_GDO_DM_GDO_ERROR	0x8004660BL	An error occurred in DM Gdo.
CCF_STORAGE_GDO_TEXT_GDO_ERROR	0x8004660CL	An error occurred in Text Gdo.
CCF_STORAGE_GDO_ENUMERATOR_CREATION	0x8004660DL	Creation of ASO counter failed.
CCF_STORAGE_GDO_ENUMDATA_FAILED	0x8004660EL	The call to ICCTxtAsoEnum::EnumData() failed.
CCF_STORAGE_GDO_ENUMSINK_CREATION	0x8004660FL	The creation of an Enumeration Sink object failed.
CCF_STORAGE_GDO_WRONG_INTERFACE	0x80046610L	The pointer was sent to the wrong interface.
CCF_STORAGE_GDO_BAD_LOCALE_ID	0x80046611L	The local ID was not found in the card.
CCF_STORAGE_GDO_CREATE_OBJECT	0x80046612L	CreateObject failed.
CCF_STORAGE_GDO_MODIFY_OBJECT	0x80046613L	ModifyObject failed.
CCF_STORAGE_GDO_READ_OBJECT	0x80046614L	ReadObject failed.
CCF_STORAGE_GDO_NOT_IMPLEMENTED	0x80046615L	Not implemented.
CCF_STORAGE_GDO_REMOVE_OBJECT	0x80046616L	RemoveObject failed.
CCF_STORAGE_GDO_REMOVE_ALL	0x80046617L	RemoveAll failed.
CCF_STORAGE_GDO_ENUM_OBJECTS	0x80046618L	EnumObjects failed.

3.13 Troubleshooting (RT Professional)

Error message	Value	
CCF_STORAGE_GDO_GET_OBJECT_COUNT	0x80046619L	GetObjectCount failed.
CCF_STORAGE_GDO_GET_RW_INTERFACE_FROM_ASO	0x8004661AL	GetRWInterface of ASO failed.
CCF_STORAGE_GDO_ON_ENUM_DATA	0x8004661BL	OnEnumData of Sink object failed.
CCF_STORAGE_GDO_ASO_CREATE	0x8004661CL	The creation of a call to Aso failed.
CCF_STORAGE_GDO_CREATE_CACHED_OBJECT	0x8004661DL	CreateCachedObject failed.
CCF_STORAGE_GDO_CREATE_SINK_OBJECT_FOR_ASO	0x8004661EL	CreateSinkObject for ASO failed.
CCF_STORAGE_GDO_CREATE_DM_GDO	0x8004661FL	Error creating DM Gdo.
CCF_STORAGE_GDO_CONNECT_DM_GDO	0x80046620L	Error connecting to DM Gdo.
CCF_STORAGE_GDO_CREATE_TEXT_GDO	0x80046621L	Error creating Text Gdo.
CCF_STORAGE_GDO_CONNECT_TEXT_GDO	0x80046622L	Error connecting to Text Gdo.
CCF_STORAGE_GDO_INVALID_ENUM_SINK_POINTER	0x80046623L	Error in Enum Sink pointer.
CCF_STORAGE_GDO_VARIANT_CHANGE_TYPE	0x80046624L	Error in Variant ChangeType.
CCF_STORAGE_GDO_GET_COLLECT_ID	0x80046625L	Error in GetCollectID.
CCF_STORAGE_GDO_COMMIT_TO_TYPE_HANDLER	0x80046626L	CommitToTypeHandler failed.
CCF_STORAGE_GDO_SET_TAG_ID	0x80046627L	Error in SetTagId.
CCF_STORAGE_GDO_SET_TEXT_ID	0x80046628L	Error in SetTextId.
CCF_STORAGE_GDO_COMMIT_CACHED_OBJECT	0x80046629L	Error in CommitCachedObject.
CCF_STORAGE_GDO_QUERY_INTERFACE_OF_SINK	0x8004662AL	QueryInterface of sink failed.
CCF_STORAGE_GDO_BAD_PROPERTY_PARAMETER	0x8004662BL	Invalid parameters defined for the property.
CCF_STORAGE_GET_TAGNAME_FROM_ID	0x8004662CL	The tag name cannot be read from the tag ID.
CCF_STORAGE_DELETE_RECORD_OF_CREATORID	0x8004662DL	The data for the Creator ID that is not identical to 0 cannot be deleted.
CCF_STORAGE_CACHE_OVERFLOW	0x8004662EL	Cache overflow
CCF_STORAGE_SERVER_DOWN	0x8004662FL	Server failure
CCF_STORAGE_ASO_PTR_NOT_INITIALIZED	0x80046630L	Aso SmartPointer has not been initialized.
CCF_STORAGE_CREATE_CONNECTION_CONTROLLER	0x80046631L	The connection controller could not be created.
CCF_STORAGE_CONNECTION_CONTROLLER_CONNECT_TO_PROJECT	0x80046632L	The connection controller could not establish a connection to the project.

Error message	Value	
CCF_STORAGE_ASO_WRAPPER_SET_CONNECTION_CONTROLLER	0x80046633L	A connection controller could not be set for Aso wrapper.
CCF_STORAGE_ASO_WRAPPER_SET_PROJECT_STORAGE	0x80046634L	No project memory could be set up for Aso wrapper.
CCF_STORAGE_PROGID_FROM_CLSID	0x80046635L	Error in ProgIDFromCLSID.
CCF_STORAGE_ASO_WRAPPER_GET_ASO	0x80046636L	Error in Aso wrapper, in "GetAso" call.
CCF_STORAGE_ASO_WRAPPER_SET_PROJECT_STORAGE_NULL	0x80046637L	Error in Aso wrapper, SetProjectStorage (NULL).
CCF_STORAGE_ASO_WRAPPER_GET_RWINTERFACE	0x80046638L	Error in Aso wrapper, in "GetRWInterface" call.
CCF_STORAGE_GDO_GET_PROJECT	0x80046639L	Error in GdoCoreBase, in GetProject call.
CCF_STORAGE_TLGGDO_VAR_NO_TRIGGER	0x8004663AL	No trigger available
CCF_STORAGE_PACKAGES_PROJECT_ASO	0x80046501L	Project ASO could not be instantiated.
CCF_STORAGE_PACKAGES_EXPORT_DATA	0x80046502L	Data export failed.
CCF_STORAGE_PACKAGES_LIST	0x80046503L	Enum Packages failed.
CCF_STORAGE_CCPACKAGEMGR_EXPORT_IS_RUNNING	0x80046504L	Data export is running.
CCF_STORAGE_CCPACKAGEMGR_UPDATE_IS_RUNNING	0x80046505L	Data update is running.
CCF_STORAGE_PACKAGES_EXPORT_USER_CANCELED	0x80046506L	Export was canceled by user.
CCF_STORAGE_PACKAGES_IMPORT_ONLY_ONE_CAS_CAN_BE_IMPORTED	0x80046507L	Only one central archive server can be imported.
CCF_STORAGE_PACKAGES_FORBID_STANDARD_SERVER	0x80046508L	The standard server configuration is not permissible for this server type.
CCF_STORAGE_PACKAGES_IMPORT_IMPORT_RED	0x80046509L	Do not import the redundant server.
CCF_STORAGE_PACKAGES_IMPORT_CHANGE_RED	0x8004650AL	Do not update the package on the redundant server.
CCF_STORAGE_PACKAGES_IMPORT_IMPORT_PHYS	0x8004650BL	Do not import the package on the identical server.
CCF_STORAGE_PACKAGES_IMPORT_CHANGE_PHYS	0x8004650CL	Do not update the package on the identical server.
CCF_STORAGE_PACKAGES_IMPORT_CHANGE_OWN	0x8004650DL	Do not import or update the package on your own server.
CCF_STORAGE_PACKAGES_IMPORT_INCLUDE_OWN_RED	0x8004650EL	Do not import or update the package on your own redundant server.
PCC_E_CANNOT_CONNECT_TO_DATABASE	0x80047000L	A connection to the database cannot be established.
PCC_E_RECONNECT_TO_DATABASE_FAILED	0x80047001L	Reconnection to the database failed.

3.13 Troubleshooting (RT Professional)

Error message	Value	
PCC_E_JOB_ID_IS_INVALID	0x80047002L	The job ID is invalid.
PCC_E_INVALID_CONTEXT_INFO_PROVIDED	0x80047003L	Invalid context information
PCC_E_DATABASE_ERROR_SEE_EVENT_LOG	0x80047004L	Database error. See event log for additional information.
PCC_E_ERROR_SEE_EVENT_LOG	0x80047005L	Undefined error, see event log for additional information.
PCC_E_INVALID_PARAMETERS	0x80047006L	Invalid parameters specified.
PCC_E_ROLLBACK_TRANSACTION_FAILED	0x80047007L	Transaction cannot be undone.
PCC_E_COMMIT_TRANSACTION_FAILED	0x80047008L	Transaction cannot be executed.
PCC_E_BEGIN_TRANSACTION_FAILED	0x80047009L	Transaction cannot be started.
PCC_E_COPY_WORLDMAP_FAILED_WORLDMAP_LOCKED	0x8004700AL	Copying of world map failed. The world map is locked.
PCC_E_MODIFY_WORLDMAP_FAILED_WORLDMAP_LOCKED	0x8004700BL	Changing of world map failed. The world map is locked.
PCC_E_REMOVE_WORLDMAP_FAILED_WORLDMAP_LOCKED	0x8004700CL	Moving of world map failed. The world map is locked.
PCC_E_MODIFY_VIEW_FAILED_VIEW_LOCKED	0x8004700DL	Changing of view failed. The view is locked.
PCC_E_REMOVE_VIEW_FAILED_VIEW_LOCKED	0x8004700EL	Moving of view failed. The view is locked.
PCC_E_COPY_WORLDMAP_FAILED_VIEWS_LOCKED	0x8004700FL	Copying of world map failed. One or more views are locked.
PCC_E_MODIFY_WORLDMAP_FAILED_VIEWS_LOCKED	0x80047010L	Changing of world map failed. One or more views are locked.
PCC_E_REMOVE_WORLDMAP_FAILED_VIEWS_LOCKED	0x80047011L	Deleting world map failed. One or more views are locked.
PCC_E_COPY_WORLDMAP_FAILED_OBJECTS_LOCKED	0x80047012L	Copying of world map failed. One or more objects are locked.
PCC_E_MODIFY_WORLDMAP_FAILED_OBJECTS_LOCKED	0x80047013L	Changing of world map failed. One or more objects are locked.
PCC_E_REMOVE_WORLDMAP_FAILED_OBJECTS_LOCKED	0x80047014L	Deleting world map failed. One or more objects are locked.
PCC_E_MODIFY_OBJECTS_FAILED_OBJECTS_LOCKED	0x80047015L	Changing of objects failed. One or more objects are locked.
PCC_E_REMOVE_OBJECTS_FAILED_OBJECTS_LOCKED	0x80047016L	Deleting objects failed. One or more objects are locked.
PCC_E_COPY_WORLDMAP_FAILED_NO_ACTIVATED_WORLDMAP	0x80047017L	Job for copying world map failed. No world map is activated.
PCC_E_REMOVE_WORLDMAP_FAILED_NO_ACTIVATED_WORLDMAP	0x80047018L	Job for moving world map failed. No world map is activated.
PCC_E_REMOVE_OBJECTS_FAILED_NO_ACTIVATED_OBJECTS	0x80047019L	Job for moving objects failed. No object is activated.
PCC_E_MODIFY_OBJECTS_FAILED_NO_ACTIVATED_OBJECTS	0x8004701AL	Job for changing objects failed. No object is activated.
PCC_E_REMOVE_VIEW_FAILED_NO_ACTIVATED_VIEW	0x8004701BL	Job for changing view failed. No view is activated.

Error message	Value	
PCC_E_MODIFY_VIEW_FAILED_NO_ACTIVATED_VIEW	0x8004701CL	Job for changing world map failed. No job is activated.
PCC_E_MODIFY_WORLD-MAP_FAILED_NO_ACTIVATED_JOB	0x8004701DL	Job for changing world map failed. No job is activated.
PCC_E_COPY_WORLD-MAP_FAILED_NOT_FOUND	0x8004701EL	Copying of world map failed. World map was not found.
PCC_E_WORLDMAP_NAME_IN_USE	0x8004701FL	The name of the world map is already in use.
PCC_E_VERSION_NUMBER_INVALID	0x80047020L	The version number is invalid.
PCC_E_USER_DOES_NOT_HAVE_WRITE_PERMISSION	0x80047021L	The user does not have write permission.
PCC_E_CREATE_AREALOCK_FAILED_OBJECTS_LOCKED	0x80047022L	Creation of the area lock failed. Locked objects exist.
PCC_E_MODIFY_AREALOCK_FAILED_OBJECTS_LOCKED	0x80047023L	Change of the area lock failed. Locked objects exist.
PCC_E_JOB_NOT_FOUND	0x80047024L	Job not found.
PCC_E_JOB_MUST_EXIST_MORE_THAN_ONCE	0x80047025L	Job must exist more than once.
PCC_E_CANNOT_MODIFY_ROOT_CTX	0x80047026L	The root context cannot be changed.
PCC_E_CANNOT_MODIFY_JOB	0x80047027L	The job cannot be changed.
PCC_E_CANNOT_MODIFY_JOB_RESERVED_BY_OTHER_USER	0x80047028L	The job cannot be changed. It is reserved by another user.
WINCC_DBPLUGIN_E_ROW_EXISTS	0x80047100L	The row already exists.
WINCC_DBPLUGIN_E_CONSTRAINT_CONFLICT	0x80047101L	A constraint conflict has occurred.
WINCC_DBPLUGIN_E_INVALID_OBJECT_NAME	0x80047102L	The object name is invalid.
WINCC_DBPLUGIN_E_INVALID_COLUMN_NAME	0x80047103L	The column name is invalid.
WINCC_DBPLUGIN_E_DATA_CONVERSION	0x80047104L	Error occurred during data conversion.
WINCC_DBPLUGIN_E_DATA_CONVERSION_DATETIME	0x80047105L	The string conversion to date/time format failed.
WINCC_DBPLUGIN_E_STRING_TOO_LONG	0x80047106L	The string is shortened.
WINCC_DBPLUGIN_E_ARITHMETIC_OVERFLOW	0x80047107L	The data type "Number overflow" cannot save the value.
WINCC_DBPLUGIN_RECORD_LOCKED	0x80047108L	The data record is locked.
WINCC_DBPLUGIN_TABLE_LOCKED	0x80047109L	The table is locked.
WINCC_DBPLUGIN_CANNOTOPENDB	0x8004710AL	The database cannot be opened.
PCC_E_CANNOT_DO_MAKE_UNIQUE_NAME	0x8004710BL	Cannot assign unique name to object.
PCC_E_NO_DB_FOUND_ON_RTS	0x8004710CL	Database on RTS master cannot be accessed or the database is in standby mode.

3.13 Troubleshooting (RT Professional)

Error message	Value	
PCC_E_OPERATION_NOT_ALLOWED_ON_RTS	0x8004710DL	This process is not permitted in the RTS database. This process must be executed in the engineering database on PSOS.
PCC_E_NO_DB_FOUND_ON_PSOS	0x8004710EL	Database on PSOS could not be accessed.

Index

—

_ismbcalnum function, 1527
_ismbcalpha function, 1527
_ismbcdigit function, 1527
_ismbcgraph function, 1527
_ismbcclower function, 1527
_ismbcprint function, 1527
_ismbcpunct function, 1527
_ismbcspc function, 1527
_ismbcupper function, 1527
_mbcat function, 1528
_mbschr function, 1528
_mbscmp function, 1527
_mbscopy function, 1528
_mbscspn function, 1527
_mbsdec function, 1528
_mbsinc function, 1528
_mbslen function, 1528
_mbsncat function, 1528
_mbsncmp function, 1527
_mbsncpy function, 1528
_mbspbrk function, 1528
_mbsrchr function, 1527
_mbsspn function, 1527
_mbsstr function, 1528
_mbstok function, 1528
_mbstrlen function, 1528

A

abort function, 1530
AboveUpperLimitColor property (VBS), 493
abs function, 1529
AcceptOnExit property (VBS), 494
AcceptOnFull property (VBS), 494
AcknowledgeAlarm, 23
acos function, 1526
Activate method (VBS), 1253
ActivateDynamic method (VBS), 1255
ActivateNextScreen, 1373
ActivatePLCCodeView, 24
ActivatePreviousScreen, 25, 1374
ActivateScreen, 26, 121, 1375
ActivateScreenByNumber, 27
ActivateScreenInScreenWindow, 122, 1376
ActivateStartScreen, 1377

ActivateStoredScreen, 1378
ActivateSystemDiagnosticsView, 28
ActiveScreen property (VBS), 496
ActiveScreenItem object, 498
ActualPointIndex property (VBS), 499
ActualPointLeft property (VBS), 499
ActualPointTop property (VBS), 500
AdaptBorder property (VBS), 501
AdaptPicture property (VBS), 502
AdaptScreenToWindow property (VBS), 502
AdaptWindowtoScreen property (VBS), 503
Address property (VBS), 503
AddressEnabled property (VBS), 504
AddressPreview property (VBS), 504
AdvancedButtonPositions property (VBS), 504
AdvancedView property (VBS), 504
Alarm object, 199
Alarm property (VBS), 504
AlarmAreaHeight property (VBS), 505
AlarmAreaWidth property (VBS), 505
AlarmClasses property (VBS), 505
AlarmColor property (VBS), 505
AlarmControl object, 236
AlarmID property, 505
AlarmLog property (VBS), 505
AlarmLogs object, 201
AlarmLowerLimit property (VBS), 506
AlarmLowerLimitColor property (VBS), 507
AlarmLowerLimitEnabled property (VBS), 508
AlarmLowerLimitRelative property (VBS), 508
Alarms object (list), 200
AlarmSource property (VBS), 509
AlarmTextTag property (VBS), 509
AlarmUpperLimit property (VBS), 509
AlarmUpperLimitColor property (VBS), 510
AlarmUpperLimitEnabled property (VBS), 511
AlarmUpperLimitRelative property (VBS), 511
AlarmView object, 251
AllFilters property (VBS), 512
AllFiltersForHitlist property (VBS), 512
AllowEdit property (VBS), 512
AllowMenu property (VBS), 512
AllServer property (VBS), 512
AllTagTypesAllowed property (VBS), 513
Analog property (VBS), 513
AngleMax property (VBS), 514
AngleMin property (VBS), 514
AnimationIgnore property (VBS), 515
Appearance property (VBS), 515

ApplicationWindow object, 257, 408
ApplyProjectSettings property (VBS), 515
ApplyProjectSettingsForDesignMode property (VBS), 516
ArchiveLogFile, 29
ArchiveName property (VBS), 516
ArchiveType property (VBS), 517
asctime function, 1531
asin function, 1526
AskOperationMotive property (VBS), 517
AspectRatio property (VBS), 518
AssignedFilters property (VBS), 519
AssignedHitlistFilters property (VBS), 519
Assignments property (VBS), 519
AssociatedS7GraphDBName property (VBS), 519
AssociatedS7GraphDBTag property (VBS), 520
atan function, 1526
atan2 function, 1526
atof function, 1529
atoi function, 1529
atol function, 1530
AttachDB method (VBS), 1259
Authorization property (VBS), 520
AutoCompleteColumns property (VBS), 522
AutoCompleteRows property (VBS), 522
AutoPosition property (VBS), 523
AutoScroll property (VBS), 524
AutoSelectionColors property (VBS), 524
AutoSelectionRectColor property (VBS), 525
AutoShow property (VBS), 526
AutoSizing property (VBS), 526
AutoStart property (VBS), 526
AvailableStatusBarElements property (VBS), 527
AvailableToolBarButtons property (VBS), 527
AverageLast15Values property (VBS), 527
AxisXBunchCount property (VBS), 527
AxisXMarkCount property (VBS), 528
AxisXNoOfDigits property (VBS), 528
AxisXShowBunchValues property (VBS), 528
AxisXStyle property (VBS), 528
AxisY1BunchCount property (VBS), 528
AxisY1MarkCount property (VBS), 528
AxisY1ShowBunchValues property (VBS), 528
AxisY2BunchCount property (VBS), 529
AxisY2MarkCount property (VBS), 529
AxisY2ShowBunchValues property (VBS), 529

B

Back up RAM file system, 30
BackButtonVisible property (VBS), 529
BackColor property (VBS), 529

BackColorBottom property (VBS), 532
BackColorTop property (VBS), 533
BackFillStyle property (VBS), 534
BackFlashingColorOff property (VBS), 536
BackFlashingColorOn property (VBS), 537
BackFlashingEnabled property (VBS), 538
BackFlashingRate property (VBS), 539
BackgroundColor property (VBS), 541
BackPicture property (VBS), 541
BackStyle property (VBS), 541
BackupRAMFileSystem, 30
Bar object, 258
BarBackColor property (VBS), 542
BarBackFillStyle property (VBS), 543
BarBackFlashingColorOff property (VBS), 544
BarBackFlashingColorOn property (VBS), 544
BarBackFlashingEnabled property (VBS), 545
BarBackFlashingRate property (VBS), 545
BarColor property (VBS), 545
BarEdgeStyle property (VBS), 546
BarOrientation property (VBS), 547
BaseScreenName property (VBS), 547
Batteryview object, 265
BelowLowerLimitColor property (VBS), 548
BitNumber property (VBS), 549
BlinkColor property (VBS), 550
BlinkMode property (VBS), 550
BlinkSpeed property (VBS), 551
BlockAlignment property (VBS), 552
BlockAutoPrecisions property (VBS), 552
BlockCaption property (VBS), 553
BlockCount property (VBS), 553
BlockDateFormat property (VBS), 554
BlockExponentialFormat property (VBS), 554
BlockHideText property (VBS), 555
BlockHideTitleText property (VBS), 555
BlockId property (VBS), 556
BlockIndex property (VBS), 556
BlockLength property (VBS), 557
BlockName property (VBS), 557
BlockPrecisions property (VBS), 558
Blocks property (VBS), 558
BlockShowDate property (VBS), 558
BlockShowIcon property (VBS), 559
BlockShowTitleIcon property (VBS), 560
BlockTimeFormat property (VBS), 560
BlockUseSourceFormat property (VBS), 561
BorderBackColor property (VBS), 561
BorderBrightColor3D property (VBS), 564
BorderColor property (VBS), 565
BorderEnabled property (VBS), 568
BorderEndStyle property (VBS), 568

div function, 1529
exit function, 1530
exp function, 1526
fabs function, 1526
fclose function, 1528
feof function, 1528
ferror function, 1528
fflush function, 1528
fgetc function, 1528
fgetpos function, 1528
fgets function, 1528
floor function, 1526
fmod function, 1526
fopen function, 1528
fputc function, 1529
fputs function, 1529
fread function, 1529
free function, 1530
freopen function, 1528
frexp function, 1526
fseek function, 1529
fsetpos function, 1529
ftell function, 1529
fwrite function, 1529
getc function, 1529
getenv function, 1529
gmtime function, 1531
isalnum function, 1524
isalpha function, 1524
isdigit function, 1524
isgraph function, 1524
islower function, 1524
isprint function, 1525
ispunct function, 1525
isspace function, 1525
isupper function, 1525
isxdigit function, 1525
labs function, 1530
ldexp function, 1526
ldiv function, 1529
localtime function, 1531
log function, 1526
log10 function, 1526
malloc function, 1530
memchr function, 1527
memcmp function, 1527
memcpy function, 1527
memmove function, 1527
memset function, 1527
mktime function, 1531
modf function, 1526
pow function, 1526

putc function, 1529
qsort function, 1530
rand function, 1530
realloc function, 1530
remove function, 1529
rename function, 1529
rewind function, 1529
setbuf function, 1529
setvbuf function, 1529
sin function, 1526
sinh function, 1526
sqrt function, 1526
srand function, 1530
strcat function, 1530
strchr function, 1530
strcmp function, 1531
strcpy function, 1530
strcspn function, 1531
strerror function, 1530
strftime function, 1531
strlen function, 1531
strncat function, 1530
strncmp function, 1531
strncpy function, 1530
strpbrk function, 1530
strrchr function, 1530
strspn function, 1531
strstr function, 1530
strtod function, 1529
strtok function, 1530
strtol function, 1530
strtoul function, 1530
system function, 1530
tan function, 1526
tanh function, 1526
time function, 1531
tmpfile function, 1528
tmpnam function, 1528
tolower function, 1525
toupper function, 1525
ungetc function, 1529
vfprintf function, 1529
vsprintf function, 1529
C text lists, 2050, 2052
CalculateStatistic method (VBS), 1260
CalibrateTouchScreen, 31
calloc function, 1530
CameraControl object, 272
CameraUrl property (VBS), 590
CanBeGrouped property (VBS), 590
Caption property (VBS), 590
CaptionBackColor property (VBS), 591

- CaptionColor property (VBS), 592
- CaptionFont property (VBS), 592
- CaptionText property (VBS), 593
- CaptionTop property (VBS), 594
- ceil function, 1526
- CellCut property (VBS), 594
- CellSpaceBottom property (VBS), 595
- CellSpaceLeft property (VBS), 596
- CellSpaceRight property (VBS), 596
- CellSpaceTop property (VBS), 597
- CenterColor property (VBS), 598
- CenterSize property (VBS), 598
- ChangeConnection, 32
- ChangeConnectionEIP, 33
- ChangeMouseCursor property (VBS), 599
- ChannelDiagnose object, 274
- CheckBox object, 276
- CheckMarkAlignment property (VBS), 600
- CheckMarkCount property (VBS), 600
- Circle object, 279
- CircleSegment object, 282
- CircularArc object, 285
- ClearAlarmBuffer, 35
- ClearAlarmBufferProTool, 36
- ClearAlarmBufferProtoolLegacy, 36
- ClearDataRecord, 37
- ClearDataRecordMemory, 38
- clearerr function, 1529
- ClearLog, 39
- ClearOnError property (VBS), 601
- ClearOnFocus property (VBS), 601
- clock function, 1531
- clock functions
 - asctime function, 1531
- Clock object, 287
- Closeable property (VBS), 602
- CloseAllLogs, 40
- Color property (VBS), 603
- ColorChangeHysteresis property (VBS), 604
- ColorChangeHysteresisEnabled property (VBS), 604
- ColumnAdd property (VBS), 605
- ColumnAlias property (VBS), 605
- ColumnAlignment property (VBS), 606
- ColumnAutoPrecisions property (VBS), 607
- ColumnCaption property (VBS), 607
- ColumnCount property (VBS), 608
- ColumnDateFormat property (VBS), 608
- ColumnDMVarName property (VBS), 609
- ColumnExponentialFormat property (VBS), 609
- ColumnFlagNotNull property (VBS), 610
- ColumnFlagUnique property (VBS), 610
- ColumnHideText property (VBS), 611
- ColumnHideTitleText property (VBS), 612
- ColumnIndex property (VBS), 612
- ColumnLeadingZeros property (VBS), 613
- ColumnLength property (VBS), 613
- ColumnMaxValue property (VBS), 614
- ColumnMinValue property (VBS), 614
- ColumnName property (VBS), 615
- ColumnOrder property (VBS), 616
- ColumnPosition property (VBS), 616
- ColumnPrecisions property (VBS), 616
- ColumnReadAccess property (VBS), 617
- ColumnReadonly property (VBS), 617
- ColumnRemove property (VBS), 618
- ColumnRepos property (VBS), 618
- ColumnResize property (VBS), 619
- Columns property (VBS), 620
- ColumnScrollbar property (VBS), 620
- ColumnSettings property (VBS), 621
- ColumnSettingsBufferView property (VBS), 621
- ColumnShowDate property (VBS), 621
- ColumnShowIcon property (VBS), 622
- ColumnShowTitleIcon property (VBS), 622
- ColumnsMoveable property (VBS), 623
- ColumnSort property (VBS), 623
- ColumnSortIndex property (VBS), 624
- ColumnStartValue property (VBS), 624
- ColumnStringLength property (VBS), 625
- ColumnTextAckGroup property (VBS), 625
- ColumnTextAlarmState property (VBS), 626
- ColumnTextAlarmText property (VBS), 626
- ColumnTextBit property (VBS), 626
- ColumnTextClassName property (VBS), 626
- ColumnTextConnection property (VBS), 626
- ColumnTextDataType property (VBS), 626
- ColumnTextDateTime property (VBS), 627
- ColumnTextDbNumber property (VBS), 627
- ColumnTextDevice property (VBS), 627
- ColumnTextDiagnosable property (VBS), 627
- ColumnTextFormat property (VBS), 627
- ColumnTextGroup property (VBS), 627
- ColumnTextLogTime property (VBS), 627
- ColumnTextNumber property (VBS), 628
- ColumnTextOffset property (VBS), 628
- ColumnTextPassword property (VBS), 628
- ColumnTextTagConnection property (VBS), 628
- ColumnTextTime property (VBS), 628
- ColumnTextTrend property (VBS), 628
- ColumnTextType property (VBS), 629
- ColumnTextUser property (VBS), 629
- ColumnTextValue property (VBS), 629
- ColumnTextWrite property (VBS), 629
- ColumnTextXValue property (VBS), 629

- ColumnTimeFormat property (VBS), 629
- ColumnTitleAlignment property (VBS), 630
- ColumnTitles property (VBS), 631
- ColumnType property (VBS), 631
- ColumnVisible property (VBS), 632
- ColumnWidth property (VBS), 633
- ColumnWriteAccess property (VBS), 633
- ComboBox object, 290
- ComboboxFont property (VBS), 633
- CompatibilityMode property (VBS), 634
- ComplexViewToolbar property (VBS), 634
- ComplexViewToolbarBounds property (VBS), 634
- ComponentInfoText property (VBS), 634
- ConfiguredAlarmClasses property (VBS), 634
- ConnectionType property (VBS), 634
- ConnectOnStart property (VBS), 635
- Connector object, 293
- ConnectTrendWindows property (VBS), 635
- ContinuousChange property (VBS), 636
- Control Center
 - Change tag values, 1733
 - Change tag values with alarm output, 1738
 - Change tag values with notification, 1742
 - Change tag values with notification and alarm output, 1747
 - Connect notify function, 1618
 - Define update tags, 1808
 - Edit tag properties via dialog, 1752
 - Get data type of a tag, 1726
 - Get limits of a tag, 1719
 - Get tag values, 1691
 - Get tag values with update, 1704
 - List connection data, 1789
 - List information about a tag group, 1681
 - List tags in a structured tag, 1778
 - Open tag selection dialog, 1756, 1762
 - Remove notify function, 1642, 1649
 - Retrieve tag ID, 1708
 - Retrieve tag name, 1708
- ControlDesignMode property (VBS), 637
- Controls
 - WinCC MediaPlayer, 343
- ControlSmartServer, 40
- ControlWebServer, 41
- CopyRows method (VBS), 1260
- CornerRadius property (VBS), 638
- CornerStyle property (VBS), 638
- cos function, 1526
- cosh function, 1526
- Count property (VBS), 639
- CountDivisions property (VBS), 640
- CountOfLinesPerAlarms property (VBS), 640

- CountOfVisibleAlarms property (VBS), 640
- CountSubDivisions property (VBS), 640
- CountVisibleItems property (VBS), 641
- Create method (VBS), 1261
- CreateTagSet-Methode, 1261
- ctime function, 1531
- ctype functions
 - isalnum function, 1524
 - isalpha function, 1524
 - isdigit function, 1524
 - isgraph function, 1524
 - islower function, 1524
 - isprint function, 1525
 - ispunct function, 1525
 - isspace function, 1525
 - isupper function, 1525
 - isxdigit function, 1525
 - tolower function, 1525
 - toupper function, 1525
- CursorControl property (VBS), 642
- Curves property (VBS), 643
- CutRows method (VBS), 1262

D

- DangerRangeColor property (VBS), 643
- DangerRangeStart property (VBS), 644
- DangerRangeVisible property (VBS), 645
- DataFormat property (VBS), 645
- DataItem object, 203
- DataLogs object, 204
- DataRecordNameCaption property (VBS), 647
- DataRecordNrCaption property (VBS), 647
- DataSet object (list), 206
- DataSource property (VBS), 648
- DateTimeField object, 296
- DeactivateDynamic method (VBS), 1262
- DecreaseTag, 43, 123, 1379
- DefaultFilterEom property (VBS), 648
- DefaultHitListFilterEom property (VBS), 648
- DefaultMsgFilterSQL property (VBS), 648
- DefaultSort property (VBS), 649
- DefaultSort2 property (VBS), 649
- DefaultSort2Column property (VBS), 650
- DeleteRows method (VBS), 1264
- DetachDB method (VBS), 1265
- DeviceStyle property (VBS), 651
- DiagnosticsContext property (VBS), 651
- DiagramAreaHeight property (VBS), 651
- DiagramAreaLeft property (VBS), 651
- DiagramAreaTop property (VBS), 652
- DiagramAreaWidth property (VBS), 652

DialColor property (VBS), 652
 DialFillStyle property (VBS), 653
 DialPicture property (VBS), 653
 DialSize property (VBS), 654
 difftime function, 1531
 DiskSpaceView object, 299
 Display3D property (VBS), 654
 DisplayButton2Plc property (VBS), 654
 DisplayButtonComparison property (VBS), 655
 DisplayButtonDelete property (VBS), 655
 DisplayButtonFromPlc property (VBS), 655
 DisplayButtonHelp property (VBS), 655
 DisplayButtonNew property (VBS), 655
 DisplayButtonSave property (VBS), 655
 DisplayButtonSaveAs property (VBS), 655
 DisplayCentury property (VBS), 656
 DisplayComboBox property (VBS), 656
 DisplayGridLines property (VBS), 656
 DisplayLabeling property (VBS), 656
 DisplayNumbers property (VBS), 656
 DisplayOptions property (VBS), 656
 DisplaySize property (VBS), 657
 DisplayStatusBar property (VBS), 657
 DisplaySystemTime property (VBS), 657
 DisplayTable property (VBS), 657
 div function, 1529
 DMAddNotify (Control Center), 1618
 DMEnumConnectionDataExStr (Control Center), 1789
 DMEnumVarGrpDataExStr (Control Center), 1681
 DMEnumVariables (Control Center), 1738
 DMFireNotifyData (Control Center), 1642
 DMGetValueExStr (Control Center), 1691
 DMGetValueWaitExStr (Control Center), 1704
 DMGetVarInfoExStr (Control Center), 1708
 DMGetVarLimitsExStr (Control Center), 1719
 DMGetVarTypeExStr (Control Center), 1726
 DMRemoveNotify (Control Center), 1649
 DMSetValueExStr (Control Center), 1733
 DMSetValueWaitExStr (Control Center), 1742
 DMSetValueWaitMessageExStr (Control Center), 1747
 DMShowVarDatabaseExStr (Control Center), 1756
 DMShowVarDatabaseMultiExStr (Control Center), 1762
 DMShowVarPropertiesExStr (Control Center), 1752
 DMStartVarUpdateExStr (Control Center), 1808
 DoubleClickAction property (VBS), 658
 DrawInsideFrame property (VBS), 658
 Drive property (VBS), 659

E

EdgeStyle property (VBS), 660
 Edit method (VBS), 1265
 EditAlarm, 44
 EditOnFocus property (VBS), 662
 Ellipse object, 301
 EllipseSegment object, 304
 EllipticalArc object, 306
 Enabled property (VBS), 663
 EnableDelete property (VBS), 667
 EnableEdit property (VBS), 667
 EnableInsert property (VBS), 668
 EnableNavigateButtons property (VBS), 668
 EnableNavigateKeys property (VBS), 669
 Encode, 44
 EncodeEx, 45
 EncryptCommunication property (VBS), 669
 EndAngle property (VBS), 669
 EndLeft property (VBS), 670
 EndStyle property (VBS), 670
 EndTop property (VBS), 671
 EnterButtonVisible property (VBS), 671
 EntryNameCaption property (VBS), 671
 EntryNameColumnWidth property (VBS), 671
 EntryValueColFirst property (VBS), 671
 EntryValueColumnWidth property (VBS), 671
 EntryValueFieldLength property (VBS), 671
 EntryValuePos property (VBS), 672
 ErrorColor property (VBS), 672
 Errorflag property (VBS), 674
 ES2RT_ButtonPositions property (VBS), 675
 ES2RT_ColumnOrder property (VBS), 675
 ES2RT_ColumnWidth property (VBS), 675
 ES2RT_EntryNameColumnWidth property (VBS), 675
 ES2RT_EntryValueColumnWidth property (VBS), 675
 ES2RT_ListAreaHeight property (VBS), 675
 ES2RT_ListAreaWidth property (VBS), 675
 ES2RT_MessageAreaHeight property (VBS), 676
 ES2RT_MessageAreaWidth property (VBS), 676
 ES2RT_StoreAsCheckBack property (VBS), 676
 Es2rtButtonPositions property (VBS), 676
 Es2rtTableBounds property (VBS), 676
 EscButtonVisible property (VBS), 676
 EvenRowBackColor property (VBS), 677
 exit function, 1530
 exp function, 1526
 Export
 Recipe, 48

Export method (VBS), 1266
ExportDataRecords, 46
ExportDataRecordsWithChecksum, 48
ExportDelimiter property (VBS), 677
ExportDirectoryChangeable property (VBS), 677
ExportDirectoryname property (VBS), 678
ExportFileExtension property (VBS), 678
ExportFilename property (VBS), 679
ExportFilenameChangeable property (VBS), 680
ExportFormat property (VBS), 681
ExportFormatGuid property (VBS), 681
ExportFormatName property (VBS), 682
ExportImportUserAdministration, 51, 124
ExportParameters property (VBS), 683
ExportSelection property (VBS), 683
ExportShowDialog property (VBS), 684
ExtraHeightOffset property (VBS), 686

F

fabs function, 1526
fclose function, 1528
feof function, 1528
ferror function, 1528
fflush function, 1528
fgetc function, 1528
fgetpos function, 1528
fgets function, 1528
FieldLength property (VBS), 686
FileName property (VBS), 686
FillColorMode property (VBS), 687
FillingDirection property (VBS), 690
FillPattern property (VBS), 688
FillPatternColor property (VBS), 688
FillStyle property (VBS), 690
Filter property (VBS), 691
FilterSQL property (VBS), 691
FilterTag property (VBS), 692
FilterText property (VBS), 692
FirstConnectedObject property (VBS), 692
FirstConnectedObjectIndex property (VBS), 692
FirstConnectedObjectName property (VBS), 693
FirstGradientColor property (VBS), 693
FirstGradientOffset property (VBS), 693
FitToLargest property (VBS), 694
FitToSize property (VBS), 694
FitToSizeLowerRows property (VBS), 694
FitToSizeUpperRows property (VBS), 694
FixedAspectRatio property (VBS), 694
Flashing property (VBS), 695
FlashingColorOff property (VBS), 696
FlashingColorOn property (VBS), 698

FlashingEnabled property (VBS), 700
FlashingOnLimitViolation property (VBS), 702
FlashingRate property (VBS), 702
FlashTransparentColor property (VBS), 704
Flip property (VBS), 705
floor function, 1526
fmod function, 1526
FocusColor property (VBS), 706
FocusWidth property (VBS), 707
Font property (VBS), 708
FontBold property (VBS), 711
FontItalic property (VBS), 711
FontName property (VBS), 712
FontSize property (VBS), 714
FontUnderline property (VBS), 714
fopen function, 1528
ForeColor property (VBS), 715
ForeColorTransparency property (VBS), 717
Format property (VBS), 717
FormatPattern property (VBS), 717
fputc function, 1529
fputs function, 1529
FrameColor property (VBS), 718
fread function, 1529
free function, 1530
Free property (VBS), 718
FreePercent property (VBS), 718
freopen function, 1528
frexp function, 1526
fseek function, 1529
fsetpos function, 1529
ftell function, 1529
FunctionTrendControl object, 309
fwrite function, 1529

G

GAPIEnumTypeMembersExStr (Control Center), 1778
Gauge object, 321
GetBrightness, 52
getc function, 1529
GetColumn, 1266, 1285, 1286
GetColumn method (VBS), 1345, 1364
GetColumnCollection, 1267
GetDataRecordFromPLC, 52
GetDataRecordName, 54
GetDataRecordTagsFromPLC, 56
getenv function, 1529
GetGroupNumber, 56
GetHitlistColumn, 1270
GetHitlisteColumnCollection, 1268

GetLanguageByLocaleID, 1384
GetLinkedTag, 1383
GetLocalScreen, 1381
GetMessageBlock, 1271
GetMessageBlockCollection, 1272
GetMessageColumn, 1273
GetMessageColumnCollection, 1279
GetOperatorMessage, 1275
GetOperatorMessageCollection, 1276
GetParentScreen, 1386
GetParentScreenWindow, 1388
GetPassword, 57
GetPropBOOL, 1390
GetPropChar, 1392
GetPropDouble, 1393
GetPropLong, 1394
GetRow, 1277
GetRulerBlock, 1280
GetRulerBlockCollection, 1281
GetRulerColumn, 1282
GetRulerColumnCollection, 1283
GetRulerData, 1284
GetStatisticAreaColumn, 1288
GetStatisticAreaColumnCollection, 1289
GetStatisticResultColumn, 1290
GetStatisticResultColumnCollection, 1291
GetStatusbarElement, 1292
GetStatusbarElementCollection, 1293
GetTagBit, 1398
GetTagBitState, 1405
GetTagBitStateQC, 1407
GetTagBitStateQCWait, 1410
GetTagBitStateWait, 1412
GetTagBitWait function, 1421
GetTagByte, 1398
GetTagByteState, 1405
GetTagByteStateQC, 1407
GetTagByteStateQCWait, 1410
GetTagByteStateWait, 1412
GetTagByteWait function, 1421
GetTagChar, 1398
GetTagCharState, 1405
GetTagCharStateQC, 1407
GetTagCharStateQCWait, 1410
GetTagCharStateWait, 1413
GetTagCharWait function, 1421
GetTagDateTime, 1398, 1399
GetTagDouble, 1398
GetTagDoubleState, 1405
GetTagDoubleStateQC, 1407
GetTagDoubleStateQCWait, 1410
GetTagDoubleStateWait, 1413
GetTagDoubleWait function, 1421
GetTagDWord function, 1398
GetTagDWordState, 1405
GetTagDWordStateQC, 1407
GetTagDWordStateQCWait, 1410
GetTagDWordStateWait, 1413
GetTagDWordWait function, 1421
GetTagFloat, 1398
GetTagFloatState, 1405
GetTagFloatStateQC, 1407
GetTagFloatStateQCWait, 1410
GetTagFloatStateWait, 1413
GetTagFloatWait function, 1421
GetTagMultiStateQCWait function, 1400
GetTagMultiStateWait, 1402
GetTagMultiWait, 1403
GetTagRaw, 1398
GetTagRawState, 1405
GetTagRawStateQC, 1407
GetTagRawStateQCWait, 1410
GetTagRawStateWait, 1413
GetTagRawWait function, 1421
GetTagSByte, 1398
GetTagSByteState, 1405
GetTagSByteStateQC, 1407
GetTagSByteStateQCWait, 1410
GetTagSByteStateWait, 1413
GetTagSByteWait function, 1421
GetTagSDWord function, 1398
GetTagSDWordState, 1405
GetTagSDWordStateQC, 1407
GetTagSDWordStateQCWait, 1410
GetTagSDWordStateWait, 1413
GetTagSDWordWait function, 1421
GetTagSWord, 1398
GetTagSWordState, 1405
GetTagSWordStateQC, 1408
GetTagSWordStateQCWait, 1410
GetTagSWordStateWait, 1413
GetTagSWordWait function, 1421
GetTagValue, 1414
GetTagValueStateQC, 1416
GetTagValueStateQCWait, 1418
GetTagValueWait, 1419
GetTagWord, 1398
GetTagWordState, 1405
GetTagWordStateQC, 1408
GetTagWordStateQCWait, 1410
GetTagWordStateWait, 1413
GetTagWordWait function, 1421
GetTimeAxis, 1295
GetTimeAxisCollection, 1296

GetTimeColumn, 1298
GetTimeColumnCollection, 1299
GetToolBarButton, 1300
GetToolBarButtonCollection, 1301
GetTrend, 1303
GetTrendCollection, 1304
GetTrendWindow, 1305
GetTrendWindowCollection, 1306
GetUserName, 58
GetValueAxis, 1307
GetValueAxisCollection, 1308
GetValueColumn, 1310
GetValueColumnCollection, 1311
GetXAxis, 1312
GetXAxisCollection, 1313
GetYAxis, 1315
GetYAxisCollection, 1316
gmtime function, 1531
GoToEnd, 58
GoToHome, 59
Gradation property (VBS), 719
GraphDirection property (VBS), 720
GraphicIOField object, 324
GraphicView object, 328
GridlineAxis property (VBS), 720
GridLineColor property (VBS), 720
GridlineEnabled property (VBS), 722
GridlineFillColor property (VBS), 722
GridlineStyle property (VBS), 722
GridLineWidth property (VBS), 722
GSCRuntimeAllowed property (VBS), 723

H

HeaderFont property (VBS), 723
Height property (VBS), 723
HelpText property (VBS), 727
HiddenInput property (VBS), 728
HideAlarm method (VBS), 1317
HighlightColor property (VBS), 728
HighLimitColor property (VBS), 729
Hitlist property (VBS), 730
HitlistColumnAdd property (VBS), 730
HitlistColumnCount property (VBS), 730
HitlistColumnName property (VBS), 732
HitlistColumnRemove property (VBS), 732
HitlistColumnRepos property (VBS), 733
HitlistColumnSort property (VBS), 733
HitlistColumnSortIndex property (VBS), 734
HitlistColumnVisible property (VBS), 735
HitlistDefaultSort property (VBS), 735
HitlistFilter property (VBS), 736

HitlistMaxSourceItems property (VBS), 736
HitlistMaxSourceItemsWarn property (VBS), 737
HitlistRelTime property (VBS), 737
HitlistRelTimeFactor property (VBS), 738
HitlistRelTimeFactorType property (VBS), 738
HMIRuntime, 189
HMIRuntime object, 189, 208
HomeButtonVisible property (VBS), 739
HorizontalAlignment property (VBS), 739
HorizontalGridLines property (VBS), 740
HorizontalPictureAlignment property (VBS), 741
HorizontalScrollBarPosition property (VBS), 741
HorizontalScrollingEnabled property (VBS), 742
HotKey property (VBS), 742
HourNeedleHeight property (VBS), 742
HourNeedleWidth property (VBS), 743
HTMLBrowser object, 331

I

IconSpace property (VBS), 743
ImportDataRecords, 59
ImportDataRecordsWithChecksum, 61
IncreaseTag, 63, 128, 1422
Index property (VBS), 744
IndipendentWindow property (VBS), 745
InfoArea_BackgroundColor property (VBS), 745
InfoArea_ColumnsMovable property (VBS), 745
InfoArea_DefaultTextColor property (VBS), 745
InfoArea_ErrorTextBackgroundColor property (VBS), 745
InfoArea_ErrorTextColor property (VBS), 745
InfoArea_FocusFrameColor property (VBS), 746
InfoArea_FocusFrameWidth property (VBS), 746
InfoArea_Font property (VBS), 746
InfoArea_RootNodeText property (VBS), 746
InfoArea_SelectionBackgroundColor property (VBS), 746
InfoArea_SelectionForegroundColor property (VBS), 746
InfoArea_ShowGridLines property (VBS), 746
InfoArea_TableHeaderBackgroundColor property (VBS), 747
InfoArea_TableHeaderTextColor property (VBS), 747
InnerBackColorOff property (VBS), 747
InnerBackColorOn property (VBS), 748
InnerDialColor property (VBS), 748
InnerDialInnerDistance property (VBS), 748
InnerDialOuterDistance property (VBS), 749
InnerHeight property (VBS), 749
InnerWidth property (VBS), 749

- InputAddressText property (VBS), 750
 - InputValue property (VBS), 750
 - InquireLanguage, 1423
 - InspectorViewInternalColumnOrder property (VBS), 751
 - InspectorViewRowOrder property (VBS), 751
 - IntegerDigits property (VBS), 751
 - Interval property (VBS), 752
 - InverseLinearScaling, 63, 129, 1425
 - InvertBit, 65, 130, 1427
 - InvertBitInTag, 66, 132, 1428
 - InvertLinearScaling, 63, 129, 1425
 - IOField object, 333
 - IsActive property (VBS), 752
 - isalnum function, 1524
 - isalpha function, 1524
 - isdigit function, 1524
 - isgraph function, 1524
 - IsImageMiddleAligned property (VBS), 752
 - islower function, 1524
 - IsMinPasswordValueSet property (VBS), 752
 - isprint function, 1525
 - ispunct function, 1525
 - IsRunningUnderCE property (VBS), 753
 - isspace function, 1525
 - isupper function, 1525
 - IsUserAuthorized, 1430
 - IsVerticalScrollBarEnabled property (VBS), 753
 - isxdigit function, 1525
 - Item methods, 1318
 - Item object, 210
 - ItemBorderStyle property (VBS), 753
 - ItemText_AKZ property (VBS), 754
 - ItemText_Descriptor property (VBS), 754
 - ItemText_ErrorText property (VBS), 754
 - ItemText_HardwareRevision property (VBS), 754
 - ItemText_IMDataVersion property (VBS), 754
 - ItemText_InstallationDate property (VBS), 754
 - ItemText_LADDR property (VBS), 754
 - ItemText_ManufacturerID property (VBS), 755
 - ItemText_Name property (VBS), 755
 - ItemText_OKZ property (VBS), 755
 - ItemText_OperationState property (VBS), 755
 - ItemText_OrderID property (VBS), 755
 - ItemText_ProfileID property (VBS), 755
 - ItemText_Rack property (VBS), 755
 - ItemText_RevisionCounter property (VBS), 756
 - ItemText_SerialNumber property (VBS), 756
 - ItemText_Slot property (VBS), 756
 - ItemText_SoftwareRevision property (VBS), 756
 - ItemText_SpecificProfileData property (VBS), 756
 - ItemText_State property (VBS), 756
 - ItemText_Station property (VBS), 756
 - ItemText_SubAddress property (VBS), 757
 - ItemText_SubSlot property (VBS), 757
 - ItemText_SubSystem property (VBS), 757
 - ItemText_Type property (VBS), 757
 - IV_ShowItem_AKZ property (VBS), 757
 - IV_ShowItem_Descriptor property (VBS), 757
 - IV_ShowItem_ErrorText property (VBS), 757
 - IV_ShowItem_HardwareRevision property (VBS), 758
 - IV_ShowItem_IMDataVersion property (VBS), 758
 - IV_ShowItem_InstallationDate property (VBS), 758
 - IV_ShowItem_LADDR property (VBS), 758
 - IV_ShowItem_ManufacturerID property (VBS), 758
 - IV_ShowItem_Name property (VBS), 758
 - IV_ShowItem_OKZ property (VBS), 758
 - IV_ShowItem_OperationState property (VBS), 759
 - IV_ShowItem_OrderID property (VBS), 759
 - IV_ShowItem_ProfileID property (VBS), 759
 - IV_ShowItem_Rack property (VBS), 759
 - IV_ShowItem_RevisionCounter property (VBS), 759
 - IV_ShowItem_SerialNumber property (VBS), 759
 - IV_ShowItem_Slot property (VBS), 759
 - IV_ShowItem_SoftwareRevision property (VBS), 760
 - IV_ShowItem_SpecificProfileData property (VBS), 760
 - IV_ShowItem_State property (VBS), 760
 - IV_ShowItem_Station property (VBS), 760
 - IV_ShowItem_SubAddress property (VBS), 760
 - IV_ShowItem_SubSlot property (VBS), 760
 - IV_ShowItem_SubSystem property (VBS), 760
 - IV_ShowItem_Type property (VBS), 761
- J**
- JumpToLimitsAfterMouseClicked property (VBS), 761
- K**
- KeyboardOnline property (VBS), 761
- L**
- LabelColor property (VBS), 761
 - labs function, 1530
 - Language property (VBS), 762
 - LargeTickLabelingStep property (VBS), 763
 - LargeTicksBold property (VBS), 763
 - LargeTicksSize property (VBS), 764
 - LastConnectedObject property (VBS), 764
 - LastConnectedObjectIndex property (VBS), 765

LastConnectedObjectName property (VBS), 765
Layer object, 210
Layer property (VBS), 767
Layers object (list), 212
Idexp function, 1526
Idiv function, 1529
Left property (VBS), 773
LeftMargin property (VBS), 778
LeftOffset property (VBS), 778
Limit4LowerLimit property (VBS), 778
Limit4LowerLimitColor property (VBS), 779
Limit4LowerLimitEnabled property (VBS), 780
Limit4LowerLimitRelative property (VBS), 780
Limit4UpperLimit property (VBS), 781
Limit4UpperLimitColor property (VBS), 782
Limit4UpperLimitEnabled property (VBS), 782
Limit4UpperLimitRelative property (VBS), 783
Limit5LowerLimit property (VBS), 784
Limit5LowerLimitColor property (VBS), 784
Limit5LowerLimitEnabled property (VBS), 785
Limit5LowerLimitRelative property (VBS), 786
Limit5UpperLimit property (VBS), 786
Limit5UpperLimitColor property (VBS), 787
Limit5UpperLimitEnabled property (VBS), 788
Limit5UpperLimitRelative property (VBS), 788
LimitRangeCollection property (VBS), 789
Line object, 337
LineAlarmView property (VBS), 789
LinearScaling, 68, 133, 1432
LineBackgroundColor property (VBS), 789
LineColor property (VBS), 789
LineEndShapeStyle property (VBS), 790
LinesPerDiagEntry property (VBS), 793
LineStyle property (VBS), 793
LineWidth property (VBS), 793
LineWrap property (VBS), 794
ListAreaHeight property (VBS), 794
ListAreaLeft property (VBS), 795
ListAreaTop property (VBS), 795
ListAreaWidth property (VBS), 795
Listbox, 340
Lists, 232
Lists in VBS
 Alarms object (list), 200
Lists:DataSet, 206
Lists:Layers, 212
Lists:Tags Object (List), 231
LoadDataImmediately property (VBS), 795
LoadDataRecord, 70
LocalCursor property (VBS), 796
localtime function, 1531
Location property (VBS), 796

LockAlarm method (VBS), 1320
LockSquaredExtent property (VBS), 796
log function, 1526
log10 function, 1526
Logging object, 213
Logoff, 71
Logon, 71
LogOperation property (VBS), 797
LongDateTimeFormat property (VBS), 798
LongTermArchiveConsistency property (VBS), 798
Look3D property (VBS), 798
LookupText, 72, 135
LoopInAlarm method (VBS), 1320
LowerLimit property (VBS), 798
LowLimitColor property (VBS), 799

M

Machine property (VBS), 800
MachineName property (VBS), 800
MaintainAspectRatio property (VBS), 800
MaintainOriginalSize property (VBS), 801
Making the contents of screen windows
 inoperable, 1348
malloc function, 1530
MarginToBorder property (VBS), 801
math functions
 acos function, 1526
 asin function, 1526
 atan function, 1526
 ceil function, 1526
 cos function, 1526
 cosh function, 1526
 exp function, 1526
 fabs function, 1526
 floor function, 1526
 fmod function, 1526
 frexp function, 1526
 Idexp function, 1526
 log function, 1526
 log10 function, 1526
 modf function, 1526
 pow function, 1526
 sin function, 1526
 sinh function, 1526
 sqrt function, 1526
 tan function, 1526
 tanh function, 1526
MaximumNumberOfTimeAxes property (VBS), 801
MaximumNumberOfTimeColumns property
 (VBS), 801
MaximumNumberOfValueAxes property (VBS), 802

- MaximumNumberOfValueColumns property (VBS), 802
- MaximumValue property (VBS), 802
- MaxNrOfCurves property (VBS), 803
- MaxNumberOfComboBoxCharacters property (VBS), 803
- MaxToolBarRows property (VBS), 803
- MediaPlayer, 343
- memchr function, 1527
- memcmp function, 1527
- memcpy function, 1527
- memmove function, 1527
- memory functions
 - memchr function, 1527
 - memcmp function, 1527
 - memcpy function, 1527
 - memmove function, 1527
 - memset function, 1527
- memset function, 1527
- MenuButtonVisible property (VBS), 803
- MenuToolBarConfig property (VBS), 803
- MessageAreaHeight property (VBS), 804
- MessageAreaLeft property (VBS), 804
- MessageAreaTop property (VBS), 804
- MessageAreaWidth property (VBS), 804
- MessageBlockAlignment property (VBS), 804
- MessageBlockAutoPrecisions property (VBS), 805
- MessageBlockCaption property (VBS), 806
- MessageBlockCount property (VBS), 806, 811
- MessageBlockDateFormat property (VBS), 807
- MessageBlockExponentialFormat property (VBS), 807
- MessageBlockFlashOn property (VBS), 808
- MessageBlockHideText property (VBS), 809
- MessageBlockHideTitleText property (VBS), 809
- MessageBlockID property (VBS), 810
- MessageBlockLeadingZeros property (VBS), 811
- MessageBlockLength property (VBS), 812
- MessageBlockName property (VBS), 812
- MessageBlockPrecisions property (VBS), 813
- MessageBlockSelected property (VBS), 813
- MessageBlockShowDate property (VBS), 814
- MessageBlockShowIcon property (VBS), 815
- MessageBlockShowTitleIcon property (VBS), 815
- MessageBlockTextId property (VBS), 816
- MessageBlockTimeFormat property (VBS), 816
- MessageBlockType property (VBS), 817
- MessageColumnAdd property (VBS), 818
- MessageColumnCount property (VBS), 818
- MessageColumnIndex property (VBS), 819
- MessageColumnName property (VBS), 820
- MessageColumnRemove property (VBS), 820
- MessageColumnRepos property (VBS), 821
- MessageColumnSort property (VBS), 821
- MessageColumnSortIndex property (VBS), 822
- MessageColumnVisible property (VBS), 823
- MessageListType property (VBS), 823
- Methods, 1266, 1267, 1268, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1288, 1289, 1290, 1291, 1292, 1293, 1295, 1296, 1298, 1299, 1300, 1301, 1303, 1304, 1305, 1306, 1307, 1308, 1310, 1311, 1312, 1313, 1315, 1316
- Methods (VBS), 1261
 - Activate, 1253
 - ActivateDynamic, 1255
 - AttachDB, 1259
 - CalculateStatistic, 1260
 - CopyRows, 1260
 - Create, 1261
 - CreateTagSet, 1261
 - CutRows, 1262
 - DeactivateDynamic, 1262
 - DeleteRows, 1264
 - DetachDB, 1265
 - Edit, 1265
 - Export, 1266
 - GetColumn, 1345, 1364
 - HideAlarm, 1317
 - Item, 1318
 - LockAlarm, 1320
 - LoopInAlarm, 1320
 - MoveAxis, 1321
 - MoveToFirst, 1321
 - MoveToFirstLine, 1322
 - MoveToFirstPage, 1322
 - MoveToLast, 1323
 - MoveToLastLine, 1323
 - MoveToLastPage, 1324
 - MoveToNext, 1324
 - MoveToNextLine, 1325
 - MoveToNextPage, 1325
 - MoveToPrevious, 1326
 - MoveToPreviousLine, 1326
 - MoveToPreviousPage, 1327
 - NextColumn, 1327
 - NextTrend, 1328
 - OneToOneView, 1328
 - PasteRows, 1329
 - PreviousColumn, 1329
 - PreviousTrend, 1330
 - Print, 1330
 - QuitHorn, 1331
 - QuitSelected, 1331

- QuitVisible, 1332
- ReadTags, 1335
- SelectedStatisticArea, 1345
- SelectRow, 1346
- ServerExport, 1346
- ServerImport, 1347
- SetHTML, 1347
- ShowColumnSelection, 1349
- ShowComment, 1349
- ShowDisplayOptionsDialog, 1350
- ShowEmergencyQuitDialog, 1350
- ShowHelp, 1351
- ShowHideList, 1351
- ShowHitList, 1352
- ShowInfoText, 1352
- ShowLockDialog, 1353
- ShowLockList>, 1353
- ShowLongTermArchiveList, 1354
- ShowMessageList, 1354
- ShowPercentageAxis, 1355
- ShowPropertyDialog, 1355
- ShowSelectArchive, 1356
- ShowSelection, 1356
- ShowSelectionDialog, 1357
- ShowSelectTimeBase, 1357
- ShowShortTermArchiveList, 1358
- ShowSort, 1358
- ShowSortDialog, 1359
- ShowTagSelection, 1359
- ShowTimebaseDialog, 1360
- ShowTimeSelection, 1360
- ShowTrendSelection, 1361
- StartStopUpdate, 1361
- Stop, 1362
- Trace, 1362
- UnhideAlarm, 1363
- UnlockAlarm, 1363
- UnselectRow, 1364
- WriteTag, 1368
- ZoomArea, 1369
- ZoomInOut, 1369
- ZoomInOutTime, 1370
- ZoomInOutValues, 1370
- ZoomInOutX, 1371
- ZoomInOutY, 1371
- ZoomMove, 1372
- Methods:Add, 1257
- Methods:Read, 1332
- Methods:Refresh, 1336
- Methods:Remove, 1337
- Methods:RemoveAll, 1340
- Methods:Restore, 1342
- Methods:Write, 1365
- MiddleGradientColor property (VBS), 824
- MinimumNumberOfTimeAxes property (VBS), 824
- MinimumNumberOfTimeColumns property (VBS), 824
- MinimumNumberOfValueAxes property (VBS), 824
- MinimumNumberOfValueColumns property (VBS), 825
- MinimumValue property (VBS), 825
- MinNrOfCurves property (VBS), 825
- MinPasswordValue property (VBS), 826
- MinuteNeedleHeight property (VBS), 826
- MinuteNeedleWidth property (VBS), 826
- mktime function, 1531
- Mode property (VBS), 827
- modf function, 1526
- MonitorNumber property (VBS), 828
- Moveable property (VBS), 828
- MoveAxis method (VBS), 1321
- MoveToFirst method (VBS), 1321
- MoveToFirstLine method (VBS), 1322
- MoveToFirstPage method (VBS), 1322
- MoveToLast method (VBS), 1323
- MoveToLastLine method (VBS), 1323
- MoveToLastPage method (VBS), 1324
- MoveToNext method (VBS), 1324
- MoveToNextLine method (VBS), 1325
- MoveToNextPage method (VBS), 1325
- MoveToPrevious method (VBS), 1326
- MoveToPreviousLine method (VBS), 1326
- MoveToPreviousPage method (VBS), 1327
- MsgFilterSQL property (VBS), 829
- multibyte functions
 - _ismbcalnum function, 1527
 - _ismbcalpha function, 1527
 - _ismbcdigit function, 1527
 - _ismbcgraph function, 1527
 - _ismbcclower function, 1527
 - _ismbcprint function, 1527
 - _ismbcpunct function, 1527
 - _ismbcspc function, 1527
 - _ismbcupper function, 1527
 - _mbscat function, 1528
 - _mbschr function, 1528
 - _mbscmp function, 1527
 - _mbscopy function, 1528
 - _mbscspn function, 1527
 - _mbsdec function, 1528
 - _mbsinc function, 1528
 - _mbslen function, 1528
 - _mbsncat function, 1528
 - _mbsncmp function, 1527

_mbsncpy function, 1528
 _mbspbrk function, 1528
 _mbsrchr function, 1527
 _mbsspn function, 1527
 _mbsstr function, 1528
 _mbstok function, 1528
 _mbstrlen function, 1528
 MultiLineEdit, 345

N

Name property (VBS), 830
 NameColumnWidth property (VBS), 833
 NavigateTo property (VBS), 833
 NavigationButtons property (VBS), 833
 NavigationPath_Font property (VBS), 834
 NavigationPath_RootText property (VBS), 834
 NavigationPath_TextColor property (VBS), 834
 NavigationpathDiagbufferDetailText property (VBS), 834
 NavigationpathDiagbufferText property (VBS), 834
 NColumnTextDate property (VBS), 626
 NeedleBorderColor property (VBS), 834
 NeedleColor property (VBS), 835
 NeedleFillStyle property (VBS), 836
 NeedleHeight property (VBS), 836
 NextColumn method (VBS), 1327
 NextTrend method (VBS), 1328
 NoAccessInRuntime property (VBS), 628, 676
 NoHitTest property (VBS), 837
 NormalColor property (VBS), 837
 NormalRangeColor property (VBS), 837
 NormalRangeVisible property (VBS), 838
 NotifyUserAction, 73
 NumberOfButtons property (VBS), 839
 NumberOfLines property (VBS), 839
 NumberOfVisibleLines property (VBS), 839
 NumberStyle property (VBS), 840

O

Object property (VBS), 840
 Objects (VBS)
 ActiveScreenItem, 498
 AlarmControl object, 236
 AlarmView, 251
 ApplicationWindow, 257, 408
 Bar, 258
 BatteryView, 265
 Button, 266
 CameraControl, 272

ChannelDiagnose, 274
 CheckBox, 276
 Circle, 279
 CircleSegment, 282
 CircularArc, 285
 Clock, 287
 ComboBox, 290
 Connector, 293
 DateTimeField, 296
 DiskSpaceView, 299
 Ellipse object, 301
 EllipseSegment, 304
 EllipticalArc, 306
 FunctionTrendControl, 309
 Gauge, 321
 GraphicIOField, 324
 GraphicView, 328
 HMIRuntime, 189
 HTMLBrowser, 331
 IOField object, 333
 Line, 337
 Listbox, 340
 MultiLineEdit, 345
 OnlineTrendControl, 360
 OptionGroup, 372
 PDFview, 376
 PLCCodeViewer, 377
 Polygon, 379
 Polyline, 382
 ProDiagOverview, 385
 ProtectedAreaNameView, 386
 RangeLabelView, 388
 RangeQualityView, 389
 RecipeView, 390
 Rectangle, 396
 RoundButton, 399
 S7GraphOverview, 404
 Screen, 191, 215
 Screen object (list), 190
 ScreenItem, 192, 218
 ScreenItems (list), 194, 220
 ScreenWindow, 406
 Slider, 410
 SmartClientView object, 414
 SmartTag, 197, 224
 SmartTags (list), 195, 225
 StatusForce, 416
 Switch, 420
 SymbolicIOField, 424
 SymbolLibrary, 429
 SystemDiagnoseView, 432
 TableView object, 348

- TextField, 440
- TrendRulerControl, 444
- TrendView, 454
- TubeArcObject, 460
- TubeDoubleTeeObject, 462
- TubePolyline, 464
- TubeTeeObject, 467
- UserArchiveControl object, 469
- UserView, 481
- WindowSlider, 485
- WLanQualityView, 489
- ZoneLabelView, 490
- ZoneQualityView, 492
- Objects in VBS
 - Alarm object, 199
- Objects:AlarmLogs, 201
- Objects:Dataltem, 203
- Objects:DataLogs, 204
- Objects:HMIRuntime, 208
- Objects:Item, 210
- Objects:Layer, 210
- Objects:Logging, 213
- Objects:Tag, 228
- OcxGuid property (VBS), 842
- OCXState property (VBS), 842
- OcxStateForEs2Rt property (VBS), 842
- OneToOneView method (VBS), 1328
- Online property (VBS), 842
- OnlineTrendControl, 360
- OnValue property (VBS), 843
- OpenAllLogs, 74
- OpenCommandPrompt, 75
- OpenControlPanelDialog, 76
- OpenInternetExplorer, 77
- OpenScreenKeyboard, 77
- OpenTaskManager, 78
- OperationSteps property (VBS), 843
- OperatorAlarms property (VBS), 844
- OperatorMessageID property (VBS), 844
- OperatorMessageIndex property (VBS), 845
- OperatorMessageName property (VBS), 845
- OperatorMessageNumber property (VBS), 846
- OperatorMessageSelected property (VBS), 846
- OperatorMessageSource1 property (VBS), 847
- OperatorMessageSource10 property (VBS), 853
- OperatorMessageSource2 property (VBS), 848
- OperatorMessageSource3 property (VBS), 848
- OperatorMessageSource4 property (VBS), 849
- OperatorMessageSource5 property (VBS), 850
- OperatorMessageSource6 property (VBS), 850
- OperatorMessageSource7 property (VBS), 851
- OperatorMessageSource8 property (VBS), 852

- OperatorMessageSource9 property (VBS), 852
- OperatorMessageSourceType1 property (VBS), 854
- OperatorMessageSourceType10 property (VBS), 860
- OperatorMessageSourceType2 property (VBS), 854
- OperatorMessageSourceType3 property (VBS), 855
- OperatorMessageSourceType4 property (VBS), 856
- OperatorMessageSourceType5 property (VBS), 856
- OperatorMessageSourceType6 property (VBS), 857
- OperatorMessageSourceType7 property (VBS), 858
- OperatorMessageSourceType8 property (VBS), 858
- OperatorMessageSourceType9 property (VBS), 859
- OptionGroup object, 372
- OutputAddressText property (VBS), 860

P

- PaddingBottom property (VBS), 860
- PaddingLeft property (VBS), 861
- PaddingRight property (VBS), 861
- PaddingTop property (VBS), 861
- PageDown, 79
- PageMode property (VBS), 861
- PageModeMessageNumber property (VBS), 862
- PageUp, 79
- Password property (VBS), 862
- PasswordsMustBeEncrypted property (VBS), 863
- PasteRows method (VBS), 1329
- PathHeaderBackColor property (VBS), 863
- PathHeaderFont property (VBS), 863
- PathHeaderTextColor property (VBS), 864
- PDFview object, 376
- PercentageAxis property (VBS), 865
- PercentageAxisAlignment property (VBS), 865
- PercentageAxisColor property (VBS), 866
- Picture property (VBS), 867
- PictureAlignment property (VBS), 867
- PictureAreaBottomMargin property (VBS), 868
- PictureAreaLeftMargin property (VBS), 868
- PictureAreaRightMargin property (VBS), 868
- PictureAreaTopMargin property (VBS), 868
- PictureAutoSizing property (VBS), 868
- PictureDeactivated property (VBS), 868
- PictureList property (VBS), 869
- PictureOff property (VBS), 869
- PictureOn property (VBS), 870
- PictureRotation property (VBS), 870
- PictureSizeMode property (VBS), 871
- PlayCount property (VBS), 871
- PlayEndless property (VBS), 871
- PLC code viewer
 - OpenViewerIECPLByAssignment, 2470

- OpenViewerIECPLByCall, 2464
- OpenViewerIECPLByFCCall, 2467
- OpenViewerS7GraphByBlock, 2461
- PLCCodeViewer object, 377
- PLCFilter property (VBS), 872
- PlcUDTFilter property (VBS), 872
- PointerColor property (VBS), 872
- Points property (VBS), 873
- PointsCount property (VBS), 873
- Polygon object, 379
- Polyline object, 382
- PopupMenuEnabled property (VBS), 874
- PositionFont property (VBS), 874
- pow function, 1526
- Precision property (VBS), 874
- PreferredUseOnAck property (VBS), 875
- Pressed property (VBS), 875
- PreviousColumn method (VBS), 1329
- PreviousTrend method (VBS), 1330
- Print method (VBS), 1330
- PrintJob property (VBS), 875
- PrintReport, 80
- PrintScreen, 80
- ProcessTag property (VBS), 876
- ProcessValue property (VBS), 876
- ProDiagOverview object, 385
- ProhibitDataRecordTagInOnlySimpleView property (VBS), 878
- Properties (VBS)
 - AboveUpperLimitColor, 493
 - AcceptOnExit, 494
 - AcceptOnFull, 494
 - AccessPath, 495
 - ActiveProject, 495
 - ActiveScreen, 496, 497
 - ActualPointIndex, 499
 - ActualPointLeft, 499
 - ActualPointTop, 500
 - AdaptBorder, 501
 - AdaptPicture, 502
 - AdaptScreenToWindow, 502
 - AdaptWindowtoScreen, 503
 - Address, 503
 - AddressEnabled, 504
 - AdressPreview, 504
 - AdvancedButtonPositions, 504
 - AdvancedView, 504
 - Alarm, 504
 - AlarmAreaHeight, 505
 - AlarmAreaWidth, 505
 - AlarmClasses, 505
 - AlarmColor, 505
 - AlarmID, 505
 - AlarmLog, 505
 - AlarmLowerLimit, 506
 - AlarmLowerLimitColor, 507
 - AlarmLowerLimitEnabled, 508
 - AlarmLowerLimitRelative, 508
 - AlarmSource, 509
 - AlarmTextTag, 509
 - AlarmUpperLimit, 509
 - AlarmUpperLimitColor, 510
 - AlarmUpperLimitEnabled, 511
 - AlarmUpperLimitRelative, 511
 - AllFilters, 512
 - AllFiltersForHitlist, 512
 - AllowEdit, 512
 - AllowMenu, 512
 - AllServer, 512
 - AllTagTypesAllowed, 513
 - Analog, 513
 - AngleMax, 514
 - AngleMin, 514
 - AnimationIgnore, 515
 - Appearance, 515
 - ApplyProjectSettings, 515
 - ApplyProjectSettingsForDesignMode, 516
 - ArchiveName, 516
 - ArchiveType, 517
 - AskOperationMotive, 517
 - AspectRatio, 518
 - AssignedFilters, 519
 - AssignedHitlistFilters, 519
 - Assignments, 519
 - AssociatedS7GraphDBName, 519
 - AssociatedS7GraphDBTag, 520
 - Authorization, 520
 - AutoCompleteColumns, 522
 - AutoCompleteRows, 522
 - AutoPosition, 523
 - AutoScroll, 524
 - AutoSelectionColors, 524
 - AutoSelectionRectColor, 525
 - AutoShow, 526
 - AutoSizing, 526
 - AutoStart, 526
 - AvailableStatusBarElements, 527
 - AvailableToolbarButtons, 527
 - AverageLast15Values, 527
 - AxisXBunchCount, 527
 - AxisXMarkCount, 528
 - AxisXNoOfDigits, 528
 - AxisXShowBunchValues, 528
 - AxisXStyle, 528

AxisY1BunchCount, 528
AxisY1MarkCount, 528
AxisY1ShowBunchValues, 528
AxisY2BunchCount, 529
AxisY2MarkCount, 529
AxisY2ShowBunchValues, 529
BackButtonVisible, 529
BackColor, 529
BackColorBottom, 532
BackColorTop, 533
BackFillStyle, 534
BackFlashingColorOff, 536
BackFlashingColorOn, 537
BackFlashingEnabled, 538
BackFlashingRate, 539
BackgroundColor, 541
BackPicture, 541
BackStyle, 541
BarBackColor, 542
BarBackFillStyle, 543
BarBackFlashingColorOff, 544
BarBackFlashingColorOn, 544
BarBackFlashingEnabled, 545
BarBackFlashingRate, 545
BarColor, 545
BarEdgeStyle, 546
BarOrientation, 547
BaseScreenName, 547
BelowLowerLimitColor, 548
BitNumber, 549
BlinkColor, 550
BlinkMode, 550
BlinkSpeed, 551
BlockAlignment, 552
BlockAutoPrecisions, 552
BlockCaption, 553
BlockCount, 553
BlockDateFormat, 554
BlockExponentialFormat, 554
BlockHideText, 555
BlockHideTitleText, 555
BlockId, 556
BlockIndex, 556
BlockLength, 557
BlockName, 557
BlockPrecisions, 558
Blocks, 558
BlockShowDate, 558
BlockShowIcon, 559
BlockShowTitleIcon, 560
BlockTimeFormat, 560
BlockUseSourceFormat, 561
BorderBackColor, 561
BorderBrightColor3D, 564
BorderColor, 565
BorderEnabled, 568
BorderEndStyle, 568
BorderFlashingColorOff, 569
BorderFlashingColorOn, 570
BorderFlashingEnabled, 572
BorderFlashingRate, 573
BorderInnerStyle3D, 575
BorderInnerWidth3D, 576
BorderOuterStyle3D, 576
BorderOuterWidth3D, 577
BorderShadeColor3D, 578
BorderStyle, 579
BorderWidth, 580
BottomMargin, 584
Bounds, 584
BrowserTypeUsed, 584
BufferViewColumnOrder, 584
BufferViewInternalRowOrder, 584
BusyText, 584
ButtonBackColor, 585
ButtonBackFillStyle, 585
ButtonBarElements, 585
ButtonBarHeight, 585
ButtonBarStyle, 585
ButtonBorderBackColor, 585
ButtonBorderColor, 585
ButtonBorderWidth, 586
ButtonCornerRadius, 586
ButtonEdgeStyle, 586
ButtonFirstGradientColor, 586
ButtonFirstGradientOffset, 586
ButtonMiddleGradientColor, 586
ButtonPositions, 586
ButtonSecondGradientColor, 587
ButtonSecondGradientOffset, 587
BV_ColumnWidth_Date, 587
BV_ColumnWidth_Event, 587
BV_ColumnWidth_EventSeverity, 587
BV_ColumnWidth_EventState, 587
BV_ColumnWidth_Number, 587
BV_ColumnWidth_Time, 588
BV_ItemText_Date, 588
BV_ItemText_Event, 588
BV_ItemText_EventSeverity, 588
BV_ItemText_EventState, 588
BV_ItemText_Number, 588
BV_ItemText_Time, 588
BV_ShowItem_Date, 589
BV_ShowItem_Event, 589

BV_ShowItem_EventSeverity, 589
BV_ShowItem_EventState, 589
BV_ShowItem_Number, 589
BV_ShowItem_Time, 589
CameraUrl, 590
CanBeGrouped, 590
Caption, 590
CaptionBackColor, 591
CaptionColor, 592
CaptionFont, 592
CaptionText, 593
CaptionTop, 594
CellCut, 594
CellSpaceBottom, 595
CellSpaceLeft, 596
CellSpaceRight, 596
CellSpaceTop, 597
CenterColor, 598
CenterSize, 598
ChangeMouseCursor, 599
CheckMarkAlignment, 600
CheckMarkCount, 600
ClearOnError, 601
ClearOnFocus, 601
Closeable, 602
Color, 603
ColorChangeHysteresis, 604
ColorChangeHysteresisEnabled, 604
ColumnAdd, 605
ColumnAlias, 605
ColumnAlignment, 606
ColumnAutoPrecisions, 607
ColumnCaption, 607
ColumnCount, 608
ColumnDateFormat, 608
ColumnDMVarName, 609
ColumnExponentialFormat, 609
ColumnFlagNotNull, 610
ColumnFlagUnique, 610
ColumnHideText, 611
ColumnHideTitleText, 612
ColumnIndex, 612
ColumnLeadingZeros, 613
ColumnLength, 613
ColumnMaxValue, 614
ColumnMinValue, 614
ColumnName, 615
ColumnOrder, 616
ColumnPosition, 616
ColumnPrecisions, 616
ColumnReadAccess, 617
ColumnReadOnly, 617
ColumnRemove, 618
ColumnRepos, 618
ColumnResize, 619
Columns, 620
ColumnScrollbar, 620
ColumnSettings, 621
ColumnSettingsBufferView, 621
ColumnShowDate, 621
ColumnShowIcon, 622
ColumnShowTitleIcon, 622
ColumnsMoveable, 623
ColumnSort, 623
ColumnSortIndex, 624
ColumnStartValue, 624
ColumnStringLength, 625
ColumnTextAckGroup, 625
ColumnTextAlarmState, 626
ColumnTextAlarmText, 626
ColumnTextBit, 626
ColumnTextClassName, 626
ColumnTextConnection, 626
ColumnTextDataType, 626
ColumnTextDate, 626
ColumnTextDateTime, 627
ColumnTextDbNumber, 627
ColumnTextDevice, 627
ColumnTextDiagnosable, 627
ColumnTextFormat, 627
ColumnTextGroup, 627
ColumnTextLogTime, 627
ColumnTextNumber, 628
ColumnTextOffset, 628
ColumnTextPassword, 628
ColumnTextTagConnection, 628
ColumnTextTime, 628
ColumnTextTrend, 628
ColumnTextType, 629
ColumnTextUser, 629
ColumnTextValue, 629
ColumnTextWrite, 629
ColumnTextXValue, 629
ColumnTimeFormat, 629
ColumnTitleAlignment, 630
ColumnTitles, 631
ColumnType, 631
ColumnVisible, 632
ColumnWidth, 633
ColumnWriteAccess, 633
ComboBoxFont, 633
CompatibilityMode, 634
ComplexViewToolbar, 634
ComplexViewToolbarBounds, 634

ComponentInfoText, 634
ConfiguredAlarmClasses, 634
ConnectionType, 634
ConnectOnStart, 635
ConnectTrendWindows, 635
ContinousChange, 636
ControlDesignMode, 637
CornerRadius, 638
CornerStyle, 638
Count, 639
CountDivisions, 640
CountOfLinesPerAlarms, 640
CountOfVisibleAlarms, 640
CountSubDivisions, 640
CountVisibleItems, 641
CursorControl, 642
Curves, 643
DangerRangeColor, 643
DangerRangeStart, 644
DangerRangeVisible, 645
DataFormat, 645
DataRecordNameCaption, 647
DataRecordNrCaption, 647
DataSource, 648
DefaultFilterEom, 648
DefaultHitListFilterEom, 648
DefaultMsgFilterSQL, 648
DefaultSort, 649
DefaultSort2, 649
DefaultSort2Column, 650
DeviceStyle, 651
DiagnosticsContext, 651
DiagramAreaHeight, 651
DiagramAreaLeft, 651
DiagramAreaTop, 652
DiagramAreaWidth, 652
DialColor, 652
DialFillStyle, 653
DialPicture, 653
DialSize, 654
Display3D, 654
DisplayButton2Plc, 654
DisplayButtonComparison, 655
DisplayButtonDelete, 655
DisplayButtonFromPlc, 655
DisplayButtonHelp, 655
DisplayButtonNew, 655
DisplayButtonSave, 655
DisplayButtonSaveAs, 655
DisplayCentury, 656
DisplayComboBox, 656
DisplayGridlines, 656
DisplayLabeling, 656
DisplayNumbers, 656
DisplayOptions, 656
DisplaySize, 657
DisplayStatusBar, 657
DisplaySystemTime, 657
DisplayTable, 657
DoubleClickAction, 658
DrawInsideFrame, 658
Drive, 659
EdgeStyle, 660
EditOnFocus, 662
Enabled, 663
EnableDelete, 667
EnableEdit, 667
EnableInsert, 668
EnableNavigateButtons, 668
EnableNavigateKeys, 669
EncryptCommunication, 669
EndAngle, 669
EndLeft, 670
EndStyle, 670
EndTop, 671
EnterButtonVisible, 671
EntryNameCaption, 671
EntryNameColumnWidth, 671
EntryValueColFirst, 671
EntryValueColumnWidth, 671
EntryValueFieldLength, 671
EntryValuePos, 672
ErrorColor, 672
Errorflag, 674
ES2RT_ButtonPositions, 675
ES2RT_ColumnOrder, 675
ES2RT_ColumnWidth, 675
ES2RT_EntryNameColumnWidth, 675
ES2RT_EntryValueColumnWidth, 675
ES2RT_ListAreaHeight, 675
ES2RT_ListAreaWidth, 675
ES2RT_MessageAreaHeight, 676
ES2RT_MessageAreaWidth, 676
ES2RT_StoreAsCheckBack, 676
Es2rtButtonPositions, 676
Es2rtTableBounds, 676
EscButtonVisible, 676
EvenRowBackColor, 677
ExportDelimiter, 677
ExportDirectoryChangeable, 677
ExportDirectoryname, 678
ExportFileExtension, 678
ExportFilename, 679
ExportFilenameChangeable, 680

ExportFormat, 681
ExportFormatGuid, 681
ExportFormatName, 682
ExportParameters, 683
ExportSelection, 683
ExportShowDialog, 684
ExtraHeightOffset, 686
FieldLength, 686
FileName, 686
FillColorMode, 687
FillingDirection, 690
FillPattern, 688
FillPatternColor, 688
FillStyle, 690
Filter, 691
FilterSQL, 691
FilterTag, 692
FilterText, 692
FirstConnectedObject, 692
FirstConnectedObjectIndex, 692
FirstConnectedObjectName, 693
FirstGradientColor, 693
FirstGradientOffset, 693
FitToLargest, 694
FitToSize, 694
FitToSizeLowerRows, 694
FitToSizeUpperRows, 694
FixedAspectRatio, 694
Flashing, 695
FlashingColorOff, 696
FlashingColorOn, 698
FlashingEnabled, 700
FlashingOnLimitViolation, 702
FlashingRate, 702
FlashTransparentColor, 704
Flip, 705
FocusColor, 706
FocusWidth, 707
Font, 708
FontBold, 711
FontItalic, 711
FontName, 712
FontSize, 714
FontUnderline, 714
ForeColor, 715
ForeColorTransparency, 717
Format, 717
FormatPattern, 717
FrameColor, 718
Free, 718
FreePercent, 718
Gradation, 719
GraphDirection, 720
GridlineAxis, 720
GridLineColor, 720
GridlineEnabled, 722
GridlineFillColor, 722
GridlineStyle, 722
GridLineWidth, 722
GSCRuntimeAllowed, 723
HeaderFont, 723
Height, 723
Help text, 727
HiddenInput, 728
HighlightColor, 728
HighLimitColor, 729
Hitlist, 730
HitlistColumnAdd, 730
HitlistColumnCount, 730
HitlistColumnName, 732
HitlistColumnRemove, 732
HitlistColumnRepos, 733
HitlistColumnSort, 733
HitlistColumnSortIndex, 734
HitlistColumnVisible, 735
HitlistDefaultSort, 735
HitlistFilter, 736
HitlistMaxSourceItems, 736
HitlistMaxSourceItemsWarn, 737
HitlistRelTime, 737
HitlistRelTimeFactor, 738
HitlistRelTimeFactorType, 738
HomeButtonVisible, 739
HorizontalAlignment, 739
HorizontalGridLines, 740
HorizontalPictureAlignment, 741
HorizontalScrollBarPosition, 741
HorizontalScrollingEnabled, 742
HotKey, 742
HourNeedleHeight, 742
HourNeedleWidth, 743
IconSpace, 743
Index, 744
IndependentWindow, 745
InfoArea_BackgroundColor, 745
InfoArea_ColumnsMovable, 745
InfoArea_DefaultTextColor, 745
InfoArea_ErrorTextBackgroundColor, 745
InfoArea_ErrorTextColor, 745
InfoArea_FocusFrameColor, 746
InfoArea_FocusFrameWidth, 746
InfoArea_Font, 746
InfoArea_RootNodeText, 746
InfoArea_SelectionBackgroundColor, 746

InfoArea_SelectionForegroundColor, 746
InfoArea_ShowGridLines, 746
InfoArea_TableHeaderBackgroundColor, 747
InfoArea_TableHeaderTextColor, 747
InnerBackColorOff, 747
InnerBackColorOn, 748
InnerDialColor, 748
InnerDialInnerDistance, 748
InnerDialOuterDistance, 749
InnerHeight, 749
InnerWidth, 749
InputAddressText, 750
InputValue, 750
InspectorViewInternalColumnOrder, 751
InspectorViewRowOrder, 751
IntegerDigits, 751
Interval, 752
IsActive, 752
IsImageMiddleAligned, 752
IsMinPasswordValueSet, 752
IsRunningUnderCE, 753
IsVerticalScrollBarEnabled, 753
ItemBorderStyle, 753
ItemText_AKZ, 754
ItemText_Descriptor, 754
ItemText_ErrorText, 754
ItemText_HardwareRevision, 754
ItemText_IMDataVersion, 754
ItemText_InstallationDate, 754
ItemText_LADDR, 754
ItemText_ManufacturerID, 755
ItemText_Name, 755
ItemText_OKZ, 755
ItemText_OperationState, 755
ItemText_OrderID, 755
ItemText_ProfileID, 755
ItemText_Rack, 755
ItemText_RevisionCounter, 756
ItemText_SerialNumber, 756
ItemText_Slot, 756
ItemText_SoftwareRevision, 756
ItemText_SpecificProfileData, 756
ItemText_State, 756
ItemText_Station, 756
ItemText_SubAddress, 757
ItemText_SubSlot, 757
ItemText_SubSystem, 757
ItemText_Type, 757
IV_ShowItem_AKZ, 757
IV_ShowItem_Descriptor, 757
IV_ShowItem_ErrorText, 757
IV_ShowItem_HardwareRevision, 758
IV_ShowItem_IMDataVersion, 758
IV_ShowItem_InstallationDate, 758
IV_ShowItem_LADDR, 758
IV_ShowItem_ManufacturerID, 758
IV_ShowItem_Name, 758
IV_ShowItem_OKZ, 758
IV_ShowItem_OperationState, 759
IV_ShowItem_OrderID, 759
IV_ShowItem_ProfileID, 759
IV_ShowItem_Rack, 759
IV_ShowItem_RevisionCounter, 759
IV_ShowItem_SerialNumber, 759
IV_ShowItem_Slot, 759
IV_ShowItem_SoftwareRevision, 760
IV_ShowItem_SpecificProfileData, 760
IV_ShowItem_State, 760
IV_ShowItem_Station, 760
IV_ShowItem_SubAddress, 760
IV_ShowItem_SubSlot, 760
IV_ShowItem_SubSystem, 760
IV_ShowItem_Type, 761
JumpToLimitsAfterMouseClick, 761
KeyboardOnline, 761
LabelColor, 761
Language, 762
LargeTickLabelingStep, 763
LargeTicksBold, 763
LargeTicksSize, 764
LastConnectedObject, 764
LastConnectedObjectIndex, 765
LastConnectedObjectName, 765
Layer, 767
Left, 773
LeftMargin, 778
LeftOffset, 778
Limit4LowerLimit, 778
Limit4LowerLimitEnabled, 780
Limit4LowerLimitRelative, 780
Limit4LowerLimitColor, 779
Limit4UpperLimit, 781
Limit4UpperLimitColor, 782
Limit4UpperLimitEnabled, 782
Limit4UpperLimitRelative, 783
Limit5LowerLimit, 784
Limit5LowerLimitColor, 784
Limit5LowerLimitEnabled, 785
Limit5LowerLimitRelative, 786
Limit5UpperLimit, 786
Limit5UpperLimitColor, 787
Limit5UpperLimitEnabled, 788
Limit5UpperLimitRelative, 788
LimitRangeCollection, 789

LineAlarmView, 789
LineBackgroundColor, 789
LineColor, 789
LineEdShapeStyle, 790
LinesPerDiagEntry, 793
LineStyle, 793
LineWidth, 793
LineWrap, 794
ListAreaHeight, 794
ListAreaLeft, 795
ListAreaTop, 795
ListAreaWidth, 795
LoadDataImmediately, 795
LocalCursor, 796
Location, 796
LockSquaredExtent, 796
LogOperation, 797
LongDateTimeFormat, 798
LongTermArchiveConsistency, 798
Look3D, 798
LowerLimit, 798
LowLimitColor, 799
Machine, 800
MachineName, 800
MaintainAspectRatio, 800
MaintainOriginalSize, 801
MarginToBorder, 801
MaximumNumberOfTimeAxes, 801
MaximumNumberOfTimeColumns, 801
MaximumNumberOfValueAxes, 802
MaximumNumberOfValueColumns, 802
MaximumValue, 802
MaxNrOfCurves, 803
MaxNumberOfComboBoxCharacters, 803
MaxToolBarRows, 803
MenuButtonVisible, 803
MenuToolBarConfig, 803
MessageAreaHeight, 804
MessageAreaLeft, 804
MessageAreaTop, 804
MessageAreaWidth, 804
MessageBlockAlignment, 804
MessageBlockAutoPrecisions, 805
MessageBlockCaption, 806
MessageBlockCount, 806, 811
MessageBlockDateFormat, 807
MessageBlockExponentialFormat, 807
MessageBlockFlashOn, 808
MessageBlockHideText, 809
MessageBlockHideTitleText, 809
MessageBlockID, 810
MessageBlockLeadingZeros, 811
MessageBlockLength, 812
MessageBlockName, 812
MessageBlockPrecisions, 813
MessageBlockSelected, 813
MessageBlockShowDate, 814
MessageBlockShowIcon, 815
MessageBlockShowTitleIcon, 815
MessageBlockTextID, 816
MessageBlockTimeFormat, 816
MessageBlockType, 817
MessageColumnAdd, 818
MessageColumnCount, 818
MessageColumnIndex, 819
MessageColumnName, 820
MessageColumnRemove, 820
MessageColumnRepos, 821
MessageColumnSort, 821
MessageColumnSortIndex, 822
MessageColumnVisible, 823
MessageListType, 823
MiddleGradientColor, 824
MinimumNumberOfTimeAxes, 824
MinimumNumberOfTimeColumns, 824
MinimumNumberOfValueAxes, 824
MinimumNumberOfValueColumns, 825
MinimumValue, 825
MinNrOfCurves, 825
MinPasswordValue, 826
MinuteNeedleHeight, 826
MinuteNeedleWidth, 826
Mode, 827
MonitorNumber, 828
Moveable, 828
MsgFilterSQL, 829
NameColumnWidth, 833
NavigateTo, 833
NavigationButtons, 833
NavigationPath_Font, 834
NavigationPath_RootText, 834
NavigationPath_TextColor, 834
NavigationpathDiagbufferDetailText, 834
NavigationpathDiagbufferText, 834
NeedleBorderColor, 834
NeedleColor, 835
NeedleFillStyle, 836
NeedleHeight, 836
NoAccessInRuntime, 628, 676
NoHitTest, 837
NormalColor, 837
NormalRangeColor, 837
NormalRangeVisible, 838
NumberOfButtons, 839

NumberOfLines, 839
NumberOfVisibleLines, 839
NumberStyle, 840
OcxGuid, 842
OCXState, 842
OcxStateForEs2Rt, 842
Online, 842
OnValue, 843
OperationSteps, 843
OperatorAlarms, 844
OperatorMessageID, 844
OperatorMessageIndex, 845
OperatorMessageName, 845
OperatorMessageNumber, 846
OperatorMessageSelected, 846
OperatorMessageSource1, 847
OperatorMessageSource10, 853
OperatorMessageSource2, 848
OperatorMessageSource3, 848
OperatorMessageSource4, 849
OperatorMessageSource5, 850
OperatorMessageSource6, 850
OperatorMessageSource7, 851
OperatorMessageSource8, 852
OperatorMessageSource9, 852
OperatorMessageSourceType1, 854
OperatorMessageSourceType10, 860
OperatorMessageSourceType2, 854
OperatorMessageSourceType3, 855
OperatorMessageSourceType4, 856
OperatorMessageSourceType5, 856
OperatorMessageSourceType6, 857
OperatorMessageSourceType7, 858
OperatorMessageSourceType8, 858
OperatorMessageSourceType9, 859
OutputAddressText, 860
PaddingBottom, 860
PaddingLeft, 861
PaddingRight, 861
PaddingTop, 861
PageMode, 861
PageModeMessageNumber, 862
Password, 862
PasswordsMustBeEncrypted, 863
PathHeaderBackColor, 863
PathHeaderFont, 863
PathHeaderTextColor, 864
PercentageAxis, 865
PercentageAxisAlignment, 865
PercentageAxisColor, 866
Picture, 867
PictureAlignment, 867
PictureAreaBottomMargin, 868
PictureAreaLeftMargin, 868
PictureAreaRightMargin, 868
PictureAreaTopMargin, 868
PictureAutoSizing, 868
PictureDeactivated, 868
PictureList, 869
PictureOff, 869
PictureOn, 870
PictureRotation, 870
PictureSizeMode, 871
PlayCount, 871
PlayEndless, 871
PLCFilter, 872
PlcUDTFilter, 872
PointerColor, 872
Points, 873
PointsCount, 873
PopupMenuEnabled, 874
PositionFont, 874
Precision, 874
PreferredUseOnAck, 875
Pressed, 875
PrintJob, 875
ProcessTag, 876
ProcessValue, 876
ProhibitDataRecordTagInOnlySimpleView, 878
Radius, 879
RadiusHeight, 879
RadiusWidth, 880
Recipe, 881
RecipeName, 881
RecipeNameCaption, 881
RecipeNrCaption, 881
RecipeNrColFirst, 882
RecipeNumber, 882
RecordName, 882
RecordNrColFirst, 883
RecordNumber, 883
RelativeFillLevel, 883
RenameButtonVisible, 884
ReSizeable, 884
RightMargin, 885
Rotation, 885
RotationAngle, 885
RotationCenterLeft, 886
RotationCenterTop, 887
RoundCornerHeight, 887
RoundCornerWidth, 888
RowScrollbar, 888
RowTitleAlignment, 889
RowTitles, 890

RTPersistence, 890
RTPersistenceAuthorization, 891
RTPersistenceType, 892
RulerColor, 893
RulerColumns, 894
RulerType, 894
S7Device, 895
ScaleColor, 895
ScaleDenominator, 896
ScaleGradation, 896
ScaleLabelColor, 897
ScaleLabelFieldLength, 897
ScaleLabelFont, 898
ScaleLabelingDoubleLined, 898
ScaleNumerator, 898
ScalePosition, 898
ScaleStart, 899
ScaleTickColor, 899
ScaleTickLabelPosition, 900
ScaleTickLength, 900
ScaleTickPosition, 901
Scaling, 902
ScalingType, 902
ScreenItems, 903
ScreenName, 903
Screens, 904
ScreenScaleMode, 904
ScrollBarOrientation, 905
SecondGradientColor, 905
SecondGradientOffset, 905
SecondNeedleHeight, 905
SecondNeedleWidth, 906
SecurityForSimpleViewEnabled, 906
SegmentColoring, 906
SelectArchiveName, 907
SelectBackColor, 908
SelectedCellColor, 908
SelectedCellForeColor, 909
SelectedID, 910
SelectedIndex, 911
SelectedRowColor, 911
SelectedRowForeColor, 912
SelectedText, 913
SelectedTitleColor, 914
SelectedTitleForeColor, 915
SelectForeColor, 915
SelectionBackColor, 916
SelectionColoring, 917
SelectionForeColor, 918
SelectionRect, 919
SelectionRectColor, 919
SelectionRectWidth, 920
SelectionType, 921
SeparateLineForAlarmText, 922
SeparatorBackColor, 922
SeparatorColor, 923
SeparatorCornerStyle, 923
SeparatorLineEndShapeStyle, 924
SeparatorStyle, 925
SeparatorWidth, 925
ServerNames, 926
ServerPrefix, 926
ServerScale, 927
SetOfVisibleColumns, 927
Shared, 927
ShareSpaceWithSourceControl, 927
ShiftDecimalPoint, 928
ShowAcknowledgeButton, 928
ShowAlarmsFromDate, 928
ShowAlarmsToAcknowledge, 928
ShowBadTagState, 929
ShowBar, 929
ShowCaption, 930
ShowColumnHeaders, 930
ShowControls, 930
ShowDate, 931
ShowDecimalPoint, 931
ShowDropDownButton, 932
ShowDropDownList, 932
ShowFeatureBackward, 932
ShowFeatureForward, 933
ShowFeatureFullScreen, 934
ShowFeatureFullVolume, 934
ShowFeaturePause, 934
ShowFeaturePlay, 935
ShowFeatureStop, 935
ShowFillLevel, 936
ShowFocusRectangle, 937
ShowHelpButton, 937
ShowHorizontalGridlines, 937
ShowInnerDial, 938
ShowLargeTicksOnly, 938
ShowLeadingZeros, 938
ShowLimitLines, 938
ShowLimitMarkers, 939
ShowLimitRanges, 939
ShowLoopInAlarmButton, 939
ShowMilliseconds, 939
ShowNavigationButtons, 940
ShowPathInformation, 940
ShowPeakValuePointer, 940
ShowPendingAlarms, 940
ShowPosition, 941
ShowProcessValue, 941

ShowReadButton, 941
ShowRuler, 941
ShowRulerInAxis, 942
ShowScale, 943
ShowScrollBar, 943
ShowScrollbars, 943
ShowSignForPositiveLabel, 944
ShowSortButton, 944
ShowSortIcon, 945
ShowSplittedView, 946
ShowStatisticRuler, 946
ShowStatusBar, 947
ShowTableGridlines, 947
ShowThumb, 948
ShowTickLabels, 949
ShowTicks, 949
ShowTime, 950
ShowTimeAxis, 950
ShowTimeAxisLabeling, 950
ShowTitle, 950
ShowToolBar, 951
ShowToolBarBackgroundColor, 951
ShowTracker, 951
ShowTrendIcon, 952
ShowTrendIndicator, 953
ShowValueAxis1, 953
ShowValueAxis1Label, 953
ShowValueAxis2, 953
ShowValueAxis2Label, 954
ShowValueTable, 954
ShowWriteButton, 954
ShowY1HlpLine, 954
ShowY2HlpLine, 954
SimpleView, 954
SimpleViewToolBar, 954
Size, 955
Sizeable, 955
SmartTags, 956
SortByTimeDirection, 956
SortByTimeEnabled, 957
SortSequence, 957
SourceControl, 958
SourceControlType, 958
SplittedViewRatio, 959
StartAngle, 959
StartLeft, 960
StartStyle, 960
StartTop, 961
StartValue, 961
StatisticAreaColumns, 962
StatisticResultColumns, 962
StatusBarBackColor, 962
StatusBarElementAdd, 963
StatusBarElementAutoSize, 964
StatusBarElementCount, 965
StatusBarElementIconId, 966
StatusBarElementID, 966
StatusBarElementIndex, 967
StatusBarElementName, 968
StatusBarElementRemove, 969
StatusBarElementRename, 970
StatusBarElementRepos, 971
StatusBarElements, 972
StatusBarElementText, 972
StatusBarElementTooltipText, 973
StatusBarElementUserDefined, 973
StatusBarElementVisible, 974
StatusBarElementWidth, 975
StatusBarFont, 976
StatusBarFontColor, 977
StatusBarShowTooltips, 978
StatusBarText, 979
StatusBarUseBackColor, 979
StatusBarVisible, 980
StatuslineFont, 981
StepBackColor, 981
StepFont, 982
StepSeconds, 983
StepTextColor, 983
Style, 984
StyleItem, 985
StyleSettings, 985
SupportsInplaceEdit, 985
SupportsS7DiagnosticsInSimpleView, 985
SupportsUserDefinedToolBarButtons, 986
SwapFirstWithLastConnection, 986
SwitchOrientation, 986
SymbolTableFont, 986
SysDiagBuffButtonVisible, 987
TabIndex, 987
TabIndexAlpha, 988
TableBackColor, 990
TableColor, 991
TableColor2, 991
TableColumnsWidthAndOrder, 992
TableEvenRowBackColor, 992
TableFont, 993
TableForeColor, 993
TableForeColor2, 994
TableGridLineColor, 994
TableHeaderBackColor, 995
TableHeaderBackFillStyle, 996
TableHeaderBorderBackColor, 996
TableHeaderBorderColor, 996

TableHeaderBorderWidth, 997
TableHeaderCornerRadius, 997
TableHeaderEdgeStyle, 997
TableHeaderFirstGradientColor, 997
TableHeaderFirstGradientOffset, 997
TableHeaderFont, 997
TableHeaderForeColor, 997
TableHeaderMiddleGradientColor, 998
TableHeaderPaddingBottom, 998
TableHeaderPaddingLeft, 998
TableHeaderPaddingRight, 999
TableHeaderPaddingTop, 999
TableHeaderSecondGradientColor, 999
TableHeaderSecondGradientOffset, 999
Tag4DataRecord, 999
Tag4RecipeNumber, 999
TagForExternalTime, 999
TagPrefix, 1000
TcpPortNr, 1002
Template, 1002
Text, 1003
TextAreaBottomMargin, 1003
TextAreaLeftMargin, 1004
TextAreaRightMargin, 1004
TextAreaTopMargin, 1004
TextHandles, 1004
TextList, 1004
TextOff, 1005
TextOn, 1005
TextOrientation, 1006
Texts, 1007
TextualObjectPositions, 1007
TextualObjectsAutoSize, 1007
TextualObjectsBorderBackColor, 1007
TextualObjectsBorderColor, 1008
TextualObjectsBorderWidth, 1008
TextualObjectsCornerRadius, 1008
TextualObjectsEdgeStyle, 1008
TextualObjectsPaddingBottom, 1008
TextualObjectsPaddingLeft, 1008
TextualObjectsPaddingRight, 1008
TextualObjectsPaddingTop, 1009
ThumbBackColor, 1009
ThumbPicture, 1009
TickDistance, 1010
TicksColor, 1010
TickStyle, 1011
TimeAxes, 1012
TimeAxisAdd, 1012
TimeAxisAlignment, 1012
TimeAxisBegin, 1013
TimeAxisBeginTime, 1013
TimeAxisColor, 1014
TimeAxisCount, 1014
TimeAxisCountPoints, 1015
TimeAxisDateFormat, 1015
TimeAxisEnd, 1015
TimeAxisEndTime, 1016
TimeAxisIndex, 1016
TimeAxisInTrendColor, 1017
TimeAxisLabel, 1017
TimeAxisMeasurePoints, 1018
TimeAxisMode, 1018
TimeAxisName, 1019
TimeAxisOnline, 1019
TimeAxisRange, 1020
TimeAxisRangeType, 1020
TimeAxisRemove, 1020
TimeAxisRename, 1021
TimeAxisRepos, 1022
TimeAxisShowDate, 1022
TimeAxisSide, 1023
TimeAxisTimeFormat, 1023
TimeAxisTimeRange, 1023
TimeAxisTimeRangeBase, 1024
TimeAxisTimeRangeFactor, 1024
TimeAxisTrendWindow, 1025
TimeAxisVisible, 1025
TimeBase, 1026
TimeColumnActualize, 1027
TimeColumnAdd, 1028
TimeColumnAlignment, 1028
TimeColumnBackColor, 1029
TimeColumnBeginTime, 1030
TimeColumnCaption, 1030
TimeColumnCount, 1031
TimeColumnDateFormat, 1031
TimeColumnEndTime, 1032
TimeColumnForeColor, 1032
TimeColumnHideText, 1033
TimeColumnHideTitleText, 1034
TimeColumnIndex, 1034
TimeColumnLength, 1035
TimeColumnMeasurePoints, 1035
TimeColumnName, 1036
TimeColumnRangeType, 1036
TimeColumnRemove, 1037
TimeColumnRename, 1037
TimeColumnRepos, 1038
TimeColumns, 1039
TimeColumnShowDate, 1039
TimeColumnShowIcon, 1040
TimeColumnShowTitleIcon, 1040
TimeColumnSort, 1041

TimeColumnSortIndex, 1041
TimeColumnTimeFormat, 1042
TimeColumnTimeRangeBase, 1042
TimeColumnTimeRangeFactor, 1043
TimeColumnUseValueColumnColors, 1044
TimeColumnVisible, 1044
TimeDisplayMode, 1045
TimeStepBase, 1046
TimeStepFactor, 1047
TitleColor, 1048
TitleCut, 1049
TitleDarkShadowColor, 1049
TitleForeColor, 1050
TitleGridLineColor, 1051
TitleLightShadowColor, 1052
TitleSort, 1053
TitleStyle, 1054
Toggle, 1054
Tolerance, 1055
ToleranceColor, 1055
ToleranceLowerLimit, 1056
ToleranceLowerLimitColor, 1057
ToleranceLowerLimitEnabled, 1058
ToleranceLowerLimitRelative, 1058
ToleranceUpperLimit, 1059
ToleranceUpperLimitColor, 1059
ToleranceUpperLimitEnabled, 1060
ToleranceUpperLimitRelative, 1061
ToolBar_ButtonsHeight, 1061
ToolBar_ButtonsWidth, 1062
ToolBarAlignment, 1062
ToolBarBackColor, 1063
ToolBarBackgroundColor, 1064
ToolBarButtonActive, 1064
ToolBarButtonAdd, 1065
ToolBarButtonAuthorization, 1065
ToolBarButtonBeginGroup, 1066
ToolBarButtonClick, 1067
ToolBarButtonCount, 1068
ToolBarButtonEnabled, 1069
ToolBarButtonHeight, 1069
ToolBarButtonHotKey, 1070
ToolBarButtonID, 1070
ToolBarButtonIndex, 1071
ToolBarButtonLocked, 1072
ToolBarButtonName, 1073
ToolBarButtonRemove, 1074
ToolBarButtonRename, 1075
ToolBarButtonRepos, 1076
ToolBarButtons, 1077
ToolBarButtonSettings, 1077
ToolBarButtonsForMigration, 1077
ToolBarButtonTooltipText, 1077
ToolBarButtonUserDefined, 1078
ToolBarButtonVisible, 1079
ToolBarButtonWidth, 1080
ToolBarEnabled, 1080
ToolBarHeight, 1080
ToolBarIconStyle, 1080
ToolBarLeft, 1080
ToolBarShowTooltips, 1080
ToolBarStyle, 1081
ToolBarTop, 1081
ToolBarUseBackColor, 1081
ToolBarUseHotKeys, 1082
ToolBarVisible, 1083
ToolBarWidth, 1084
ToolTipText, 1084
Top, 1085
TopMargin, 1088
TopOffset, 1088
Total, 1089
TransitionHeaderFont, 1089
Transparency, 1090
TransparentColor, 1091
TransparentColorDeactivatedPicture, 1092
TransparentColorPictureOff, 1092
TransparentColorPictureOn, 1093
TrendActualize, 1094
TrendAdd, 1094
TrendAutoRangeBeginTagName, 1095
TrendAutoRangeBeginValue, 1095
TrendAutoRangeEndTagName, 1096
TrendAutoRangeEndValue, 1096
TrendAutoRangeSource, 1097
TrendBeginTime, 1098
TrendColor, 1098
TrendCount, 1099
TrendEndTime, 1100
TrendExtendedColorSet, 1100
TrendFill, 1101
TrendFillColor, 1102
TrendIndex, 1103
TrendIndicatorColor, 1103
TrendLabel, 1104
TrendLineStyle, 1105
TrendLineType, 1105
TrendLineWidth, 1106
TrendLowerLimit, 1107
TrendLowerLimitColor, 1107
TrendLowerLimitColoring, 1108
TrendMeasurePoints, 1109
TrendName, 1109
TrendPointColor, 1110

TrendPointStyle, 1111
TrendPointWidth, 1111
TrendProvider, 1112
TrendProviderCLSID, 1113
TrendRangeType, 1113
TrendRemove, 1114
TrendRename, 1115
TrendRepos, 1115
Trends, 1116
TrendSelectTagName, 1116
TrendSelectTagNameX, 1117
TrendSelectTagNameY, 1117
TrendsForPrinting, 1118
TrendTagName, 1118
TrendTagNameX, 1118
TrendTagNameY, 1119
TrendTimeAxis, 1120
TrendTimeRangeBase, 1120
TrendTimeRangeFactor, 1121
TrendTrendWindow, 1121
TrendUncertainColor, 1122
TrendUncertainColoring, 1123
TrendUpperLimit, 1123
TrendUpperLimitColor, 1124
TrendUpperLimitColoring, 1125
TrendValueAlign, 1125
TrendValueAxis, 1126
TrendValueUnit, 1126
TrendVisible, 1127
TrendWindowAdd, 1127
TrendWindowCoarseGrid, 1128
TrendWindowCoarseGridColor, 1129
TrendWindowCount, 1129
TrendWindowFineGrid, 1130
TrendWindowFineGridColor, 1131
TrendWindowForegroundTrendGrid, 1131
TrendWindowGridInTrendColor, 1132
TrendWindowHorizontalGrid, 1133
TrendWindowIndex, 1133
TrendWindowName, 1134
TrendWindowRemove, 1135
TrendWindowRename, 1135
TrendWindowRepos, 1136
TrendWindowRulerColor, 1136
TrendWindowRulerLayer, 1137
TrendWindowRulerStyle, 1138
TrendWindowRulerWidth, 1139
TrendWindows, 1139
TrendWindowSpacePortion, 1139
TrendWindowStatisticRulerColor, 1140
TrendWindowStatisticRulerStyle, 1141
TrendWindowStatisticRulerWidth, 1142
TrendWindowVerticalGrid, 1142
TrendWindowVisible, 1143
TrendXAxis, 1143
TrendYAxis, 1144
Unit, 1145
UnitColor, 1145
UnitFont, 1146
UnitText, 1146
UnitTop, 1147
UnitViewColumnOrder, 1148
UpdateButtonVisible, 1148
UpperLimit, 1148
UseAutoScaling, 1148
UseButtonFirstGradient, 1149
UseButtonSecondGradient, 1149
UseColumnBackColor, 1149
UseColumnForeColor, 1149
UseCursorKeyScroll, 1150
Used, 1150
UseDesignColorSchema, 1151
UseDesignShadowSettings, 1152
UsedPercent, 1155
UseExponentialFormat, 1155
UseFirstGradient, 1156
UseFlashTransparentColor, 1156
UseMessageColor, 1156
UseScadaRendererStyle, 1158
UseSecondGradient, 1158
UseSelectedTitleColor, 1158
UseSourceBackColors, 1159
UseSourceForeColors, 1160
UseSystemScrollbarWidth, 1160
UseTableColor2, 1160
UseTableHeaderFirstGradient, 1161
UseTableHeaderSecondGradient, 1161
UseTagLimitColors, 1162
UseTransparentColor, 1162
UseTransparentColorDeactivatedPicture, 1163
UseTransparentColorPictureOff, 1163
UseTransparentColorPictureOn, 1164
UseTrendNameAsLabel, 1164
UseTwoHandOperation, 1165
UseUdp, 1165
UV_ColumnWidth_AKZ, 1165
UV_ColumnWidth_Descriptor, 1165
UV_ColumnWidth_InstallationDate, 1166
UV_ColumnWidth_LADDR, 1166
UV_ColumnWidth_Name, 1166
UV_ColumnWidth_OKZ, 1166
UV_ColumnWidth_OperationState, 1166
UV_ColumnWidth_OrderID, 1166
UV_ColumnWidth_ProfileID, 1166

UV_ColumnWidth_Rack, 1167
UV_ColumnWidth_Slot, 1167
UV_ColumnWidth_SoftwareRevision, 1167
UV_ColumnWidth_SpecificProfileData, 1167
UV_ColumnWidth_State, 1167
UV_ColumnWidth_Station, 1167
UV_ColumnWidth_SubAddress, 1167
UV_ColumnWidth_SubSlot, 1168
UV_ColumnWidth_SubSystem, 1168
UV_ColumnWidth_Type, 1168
UV_ShowItem_AKZ, 1168
UV_ShowItem_Descriptor, 1168
UV_ShowItem_InstallationDate, 1168
UV_ShowItem_LADDR, 1168
UV_ShowItem_Name, 1169
UV_ShowItem_OKZ, 1169
UV_ShowItem_OperationState, 1169
UV_ShowItem_OrderID, 1169
UV_ShowItem_ProfileID, 1169
UV_ShowItem_Rack, 1169
UV_ShowItem_Slot, 1169
UV_ShowItem_SoftwareRevision, 1170
UV_ShowItem_SpecificProfileData, 1170
UV_ShowItem_State, 1170
UV_ShowItem_Station, 1170
UV_ShowItem_SubAddress, 1170
UV_ShowItem_SubSlot, 1170
UV_ShowItem_SubSystem, 1170
UV_ShowItem_Type, 1171
ValueAxes, 1172
ValueAxis1AutoRange, 1172
ValueAxis1Begin, 1172
ValueAxis1End, 1172
ValueAxis1LabelLength, 1173
ValueAxis1Style, 1173
ValueAxis2AutoRange, 1173
ValueAxis2Begin, 1173
ValueAxis2End, 1173
ValueAxis2LabelLength, 1173
ValueAxis2Style, 1173
ValueAxisAdd, 1174
ValueAxisAlignment, 1174
ValueAxisAutoPrecisions, 1175
ValueAxisAutorange, 1176
ValueAxisBeginValue, 1176
ValueAxisColor, 1177
ValueAxisCount, 1178
ValueAxisEndValue, 1178
ValueAxisExponentialFormat, 1179
ValueAxisIndex, 1179
ValueAxisInTrendColor, 1180
ValueAxisLabel, 1181
ValueAxisName, 1181
ValueAxisPrecisions, 1182
ValueAxisRemove, 1182
ValueAxisRename, 1183
ValueAxisRepos, 1183
ValueAxisScalingType, 1184
ValueAxisTrendWindow, 1185
ValueAxisVisible, 1185
ValueCaption, 1186
ValueColumnAdd, 1186
ValueColumnAlignment, 1186
ValueColumnAutoPrecisions, 1187
ValueColumnBackColor, 1188
ValueColumnCaption, 1188
ValueColumnCount, 1189
ValueColumnExponentialFormat, 1189
ValueColumnForeColor, 1190
ValueColumnHideText, 1191
ValueColumnHideTitleText, 1191
ValueColumnIndex, 1192
ValueColumnLength, 1192
ValueColumnName, 1193
ValueColumnPrecisions, 1193
ValueColumnProvider, 1194
ValueColumnProviderCLSID, 1195
ValueColumnRemove, 1195
ValueColumnRename, 1196
ValueColumnRepos, 1196
ValueColumns, 1197
ValueColumnSelectTagName, 1197
ValueColumnShowIcon, 1198
ValueColumnShowTitleIcon, 1198
ValueColumnSort, 1199
ValueColumnSortIndex, 1199
ValueColumnTagName, 1200
ValueColumnTimeColumn, 1200
ValueColumnVisible, 1201
ValueColumnWidth, 1202
ValueTableHeight, 1202
ValueTableLeft, 1202
ValueTableTop, 1202
ValueTableWidth, 1202
ValueY1HlpLine, 1202
ValueY2HlpLine, 1202
VerticalAlignment, 1203
VerticalGridLines, 1204
VerticalPictureAlignment, 1204
VerticalScrollBarEnabled, 1205
VerticalScrollBarPosition, 1205
VerticalScrolling, 1205
VerticalScrollingEnabled, 1206
ViewOnly, 1206

ViewType, 1206
 ViewTypeForSaveStream, 1206
 Visible, 1207
 VisibleItems, 1211
 Warning, 1211
 WarningColor, 1212
 WarningLowerLimit, 1212
 WarningLowerLimitColor, 1213
 WarningLowerLimitEnabled, 1214
 WarningLowerLimitRelative, 1215
 WarningRangeColor, 1215
 WarningRangeStart, 1216
 WarningRangeVisible, 1217
 WarningUpperLimit, 1217
 WarningUpperLimitColor, 1218
 WarningUpperLimitEnabled, 1219
 WarningUpperLimitRelative, 1219
 Width, 1220
 WindowCloseEnabled, 1223
 WindowMaximizeEnabled, 1223
 WindowMovingEnabled, 1224
 WindowOnTop, 1225
 WindowsContents, 1225
 WindowSizingEnabled, 1226
 WindowsStartupPosition, 1227
 WindowsStyle, 1227
 XAxes, 1228
 XAxisAdd, 1228
 XAxisAlignment, 1229
 XAxisAutoPrecisions, 1229
 XAxisAutoRange, 1230
 XAxisBeginValue, 1231
 XAxisColor, 1231
 XAxisCount, 1232
 XAxisEndValue, 1232
 XAxisExponentialFormat, 1233
 XAxisIndex, 1234
 XAxisInTrendColor, 1234
 XAxisLabel, 1235
 XAxisName, 1235
 XAxisPrecisions, 1236
 XAxisRemove, 1237
 XAxisRename, 1237
 XAxisRepos, 1238
 XAxisScalingType, 1238
 XAxisTrendWindow, 1239
 XAxisVisible, 1239
 YAxes, 1240
 YAxisAdd, 1240
 YAxisAlignment, 1241
 YAxisAutoPrecisions, 1241
 YAxisAutoRange, 1242

YAxisBeginValue, 1242
 YAxisColor, 1243
 YAxisCount, 1244
 YAxisEndValue, 1244
 YAxisExponentialFormat, 1245
 YAxisIndex, 1245
 YAxisInTrendColor, 1246
 YAxisLabel, 1247
 YAxisName, 1247
 YAxisPrecisions, 1248
 YAxisRemove, 1248
 YAxisRename, 1249
 YAxisRepos, 1249
 YAxisScalingType, 1250
 YAxisTrendWindow, 1250
 YAxisVisible, 1251
 ZeroPoint, 1251
 ZoomFactor, 1252
 Properties (VBS)"; "Name, 830
 Properties (VBS)"; "Object, 840
 Properties in VBS
 ComputerName, 634
 Context, 636
 Instance, 751
 State, 961
 UserName, 1158
 Properties in VBS:CurrentContext, 642
 Properties in VBS:ExtendedZoomingEnable, 685
 Properties in VBS:LayerDeclutteringEnable, 772
 Properties in VBS:Layers, 773
 Properties in VBS:Logging, 797
 Properties in
 VBS:ObjectSizeDeclutteringEnable, 840
 Properties in VBS:ObjectSizeDeclutteringMax, 841
 Properties in VBS:ObjectSizeDeclutteringMin, 841
 Properties in VBS:Path, 863
 Properties in VBS:QualityCode, 878
 Properties in VBS:Tags, 1001
 Properties in VBS:TimeStamp, 1045
 Properties in VBS:Value, 1171
 ProtectedAreaNameView object, 386
 putc function, 1529

Q

qsort function, 1530
 Quality codes of tags, 1547
 QuitHorn method (VBS), 1331
 QuitSelected method (VBS), 1331
 QuitVisible method (VBS), 1332

R

- Radius property (VBS), 879
- RadiusHeight property (VBS), 879
- RadiusWidth property (VBS), 880
- rand function, 1530
- RangeLabelView object, 388
- RangeQualityView object, 389
- ReadTags method (VBS), 1335
- realloc function, 1530
- Recipe
 - Export format, 48
- Recipe data record
 - Export format, 48
- Recipe property (VBS), 881
- RecipeName property (VBS), 881
- RecipeNameCaption property (VBS), 881
- RecipeNrCaption property (VBS), 881
- RecipeNrColFirst property (VBS), 882
- RecipeNumber property (VBS), 882
- RecipeView object, 390
- RecordName property (VBS), 882
- RecordNrColFirst property (VBS), 883
- RecordNumber property (VBS), 883
- Rectangle object, 396
- RelativeFillLevel property (VBS), 883
- remove function, 1529
- rename function, 1529
- RenameButtonVisible property (VBS), 884
- ReportJob, 1433
- ResetBit, 81, 136, 1434
- ResetBitInTag, 82, 137, 1436
- ReSizeable property (VBS), 884
- rewind function, 1529
- RightMargin property (VBS), 885
- Rotation property (VBS), 885
- RotationAngle property (VBS), 885
- RotationCenterLeft property (VBS), 886
- RotationCenterTop property (VBS), 887
- RoundButton object, 399
- RoundCornerHeight property (VBS), 887
- RoundCornerWidth property (VBS), 888
- RowScrollbar property (VBS), 888
- RowTitleAlignment property (VBS), 889
- RowTitles- Property (VBS), 890
- RTPersistence property (VBS), 890
- RTPersistenceAuthorization property (VBS), 891
- RTPersistenceType property (VBS), 892
- RulerColor property (VBS), 893
- RulerColumns property (VBS), 894
- RulerType property (VBS), 894

S

- S7Device property (VBS), 895
- S7GraphOverview object, 404
- SafelyRemoveHardware, 84
- SaveDataRecord, 85
- ScaleDenominator property (VBS), 896
- ScaleGradation property (VBS), 896
- ScaleLabelColor property (VBS), 897
- ScaleLabelFieldLength property (VBS), 897
- ScaleLabelFont property (VBS), 898
- ScaleLabelingDoubleLined property (VBS), 898
- ScaleNumerator property (VBS), 898
- ScalePosition property (VBS), 898
- ScaleStart property (VBS), 899
- ScaleTickColor property (VBS), 899
- ScaleTickLabelPosition property (VBS), 900
- ScaleTickLength property (VBS), 900
- ScaleTickPosition property (VBS), 901
- Scaling property (VBS), 902
- ScalingType property (VBS), 902
- Screen object, 191, 215
- Screen object (list), 190
- ScreenItem, 343
- ScreenItem object, 192, 218
- ScreenItems object (list), 194, 220
- ScreenItems property (VBS), 903
- ScreenName property (VBS), 903
- Screens property (VBS), 904
- ScreenScaleMode property (VBS), 904
- ScreenWindow object, 406
- ScrollBarOrientation property (VBS), 905
- SecondGradientColor property (VBS), 905
- SecondGradientOffset property (VBS), 905
- SecondNeedleHeight property (VBS), 905
- SecondNeedleWidth property (VBS), 906
- SecurityForSimpleViewEnabled property (VBS), 906
- SegmentColoring property (VBS), 906
- SelectArchiveName property (VBS), 907
- SelectBackColor property (VBS), 908
- SelectedCellColor property (VBS), 908
- SelectedCellForeColor property (VBS), 909
- SelectedID property (VBS), 910
- SelectedIndex property (VBS), 911
- SelectedRowColor property (VBS), 911
- SelectedRowForeColor property (VBS), 912
- SelectedStatisticArea method (VBS), 1345
- SelectedText property (VBS), 913
- SelectedTitleColor property (VBS), 914
- SelectedTitleForeColor property (VBS), 915
- SelectForeColor property (VBS), 915

- SelectionBackColor property (VBS), 916
- SelectionColoring property (VBS), 917
- SelectionForeColor property (VBS), 918
- SelectionRect property (VBS), 919
- SelectionRectColor property (VBS), 919
- SelectionRectWidth property (VBS), 920
- SelectionType property (VBS), 921
- SelectRow method (VBS), 1346
- SendEmail, 86
- SeparateLineForAlarmText property (VBS), 922
- SeparatorBackColor property (VBS), 922
- SeparatorColor property (VBS), 923
- SeparatorCornerStyle property (VBS), 923
- SeparatorLineEndShapeStyle property (VBS), 924
- SeparatorStyle property (VBS), 925
- SeparatorWidth property (VBS), 925
- ServerExport method (VBS), 1346
- ServerImport method (VBS), 1347
- ServerNames property (VBS), 926
- ServerPrefix property (VBS), 926
- ServerScale property (VBS), 927
- Set_Focus, 1437
- SetAcousticSignal, 87
- SetAlarmReportingMode, 88
- SetAlarmReportMode, 88
- SetAndGetBrightness, 89
- SetBit, 89, 139, 1438
- SetBitInTag, 91, 141, 1439
- SetBrightness, 93
- setbuf function, 1529
- SetConnectionMode, 94
- SetDataRecordTagsToPLC, 95
- SetDataRecordToPLC, 96
- SetDaylightSavingTime, 97
- SetDeviceMode, 98
- SetDisplayMode, 98
- SetHTML method (VBS), 1347
- SetLanguage, 99, 143, 1441
- SetOfVisibleColumns property (VBS), 927
- SetPLCMode, 100
- SetPropBOOL, 1444
- SetPropChar, 1445
- SetPropDouble, 1447
- SetPropertyByConstant, 143, 1449
- SetPropertyByProperty, 145, 1451
- SetPropertyByTag, 147, 1453
- SetPropertyByTagIndirect, 149, 1455
- SetPropLong, 1448
- SetRecipeTags, 101
- SetScreenKeyboardMode, 102
- SetStartAtBoot, 120
- SetTag, 103, 151, 1474
- SetTagBit, 1457
- SetTagBitState, 1464
- SetTagBitStateWait, 1466
- SetTagBitWait, 1472
- SetTagByProperty, 152, 1475
- SetTagByte, 1457
- SetTagByteState, 1464
- SetTagByteStateWait, 1466
- SetTagByteWait, 1472
- SetTagChar, 1457
- SetTagCharState, 1464
- SetTagCharStateWait, 1467
- SetTagCharWait, 1472
- SetTagDateTime, 1457, 1459
- SetTagDouble, 1457
- SetTagDoubleState, 1464
- SetTagDoubleStateWait, 1467
- SetTagDoubleWait, 1472
- SetTagDWord, 1457
- SetTagDWordState, 1464
- SetTagDWordStateWait, 1467
- SetTagDWordWait, 1473
- SetTagFloat, 1457
- SetTagFloatState, 1464
- SetTagFloatStateWait, 1467
- SetTagFloatWait, 1473
- SetTagIndirect, 154, 1478
- SetTagIndirectByProperty, 153, 155, 1477, 1479
- SetTagIndirectByTagIndirect, 156, 1480
- SetTagIndirectWithOperatorInputAlarm, 157, 1481
- SetTagMultiStateWait, 1460
- SetTagMultiWait, 1461
- SetTagRaw, 1457
- SetTagRawState, 1464
- SetTagRawStateWait, 1467
- SetTagRawWait, 1473
- SetTagSByte, 1457
- SetTagSByteState, 1464
- SetTagSByteStateWait, 1467
- SetTagSByteWait, 1473
- SetTagSDWord, 1457
- SetTagSDWordState, 1465
- SetTagSDWordStateWait, 1467
- SetTagSDWordWait, 1473
- SetTagSWord, 1457
- SetTagSWordState, 1465
- SetTagSWordStateWait, 1467
- SetTagSWordWait, 1473
- SetTagValue, 1469
- SetTagValueWait, 1470
- SetTagWord, 1457
- SetTagWordState, 1465

- SetTagWordStateWait, 1467
- SetTagWordWait, 1473
- setvbuf function, 1529
- Shared property (VBS), 927
- ShareSpaceWithSourceControl property (VBS), 927
- ShiftDecimalPoint property (VBS), 928
- ShowAcknowledgeButton property (VBS), 928
- ShowAlarmsFromDate property (VBS), 928
- ShowAlarmsToAcknowledge property (VBS), 928
- ShowAlarmWindow, 106, 113
- ShowBadTagState property (VBS), 929
- ShowBar property (VBS), 929
- ShowBlockInTiaPortalFromAlarm
 - ShowBlockInTIAPortalFromAlarm, 159
- ShowCaption property (VBS), 930
- ShowColumnHeaders property (VBS), 930
- ShowColumnSelection method (VBS), 1349
- ShowComment method (VBS), 1349
- ShowControls property (VBS), 930
- ShowDate property (VBS), 931
- ShowDecimalPoint property (VBS), 931
- ShowDisplayOptionsDialog method (VBS), 1350
- ShowDropDownButton property (VBS), 932
- ShowDropDownList property (VBS), 932
- ShowEmergencyQuitDialog method (VBS), 1350
- ShowFeatureBackward property (VBS), 932
- ShowFeatureForward property (VBS), 933
- ShowFeatureFullScreen property (VBS), 934
- ShowFeatureFullVolume property (VBS), 934
- ShowFeaturePause property (VBS), 934
- ShowFeaturePlay property (VBS), 935
- ShowFeatureStop property (VBS), 935
- ShowFillLevel property (VBS), 936
- ShowFocusRectangle property (VBS), 937
- ShowHelp method (VBS), 1351
- ShowHelpButton property (VBS), 937
- ShowHideList method (VBS), 1351
- ShowHitList method (VBS), 1352
- ShowHorizontalGridlines property (VBS), 937
- ShowInfoText method (VBS), 1352
- ShowInnerDial property (VBS), 938
- ShowLargeTicksOnly property (VBS), 938
- ShowLeadingZeros property (VBS), 938
- ShowLimitLines property (VBS), 938
- ShowLimitMarkers property (VBS), 939
- ShowLimitRanges property (VBS), 939
- ShowLockDialog method (VBS), 1353
- ShowLockList method (VBS), 1353
- ShowLogonDialog, 160
- ShowLongTermArchiveList method (VBS), 1354
- ShowLoopInAlarmButton property (VBS), 939
- ShowMessageList method (VBS), 1354
- ShowMilliseconds property (VBS), 939
- ShowNavigationButtons property (VBS), 940
- ShowOperatorNotes, 107
- ShowPathInformation property (VBS), 940
- ShowPeakValuePointer property (VBS), 940
- ShowPendingAlarms property (VBS), 940
- ShowPercentageAxis method (VBS), 1355
- ShowPLCCCodeViewFromAlarm
 - ShowPLCCCodeViewFromAlarm, 160
- ShowPopupScreen, 108, 109
- ShowPosition property (VBS), 941
- ShowProcessValue property (VBS), 941
- ShowPropertyDialog method (VBS), 1355
- ShowReadButton property (VBS), 941
- ShowRuler property (VBS), 941
- ShowRulerInAxis property (VBS), 942
- ShowScale property (VBS), 943
- ShowScrollBar property (VBS), 943
- ShowScrollbars property (VBS), 943
- ShowSelectArchive method (VBS), 1356
- ShowSelection method (VBS), 1356
- ShowSelectionDialog method (VBS), 1357
- ShowSelectTimeBase method (VBS), 1357
- ShowShortTermArchiveList method (VBS), 1358
- ShowSignForPositiveLabel property (VBS), 944
- ShowSlideInScreen, 111
- ShowSoftwareVersion, 112
- ShowSort method (VBS), 1358
- ShowSortButton property (VBS), 944
- ShowSortDialog method (VBS), 1359
- ShowSortIcon property (VBS), 945
- ShowSplittedView property (VBS), 946
- ShowStatisticRuler property (VBS), 946
- ShowStatusBar property (VBS), 947
- ShowSystemAlarm, 113
- ShowSystemDiagnosticsWindow, 113
- ShowTableGridlines property (VBS), 947
- ShowTagSelection method (VBS), 1359
- ShowThumb property (VBS), 948
- ShowTickLabels property (VBS), 949
- ShowTicks property (VBS), 949
- ShowTime property (VBS), 950
- ShowTimeAxis property (VBS), 950
- ShowTimeAxisLabeling property (VBS), 950
- ShowTimebaseDialog method (VBS), 1360
- ShowTimeSelection method (VBS), 1360
- ShowTitle property (VBS), 950
- ShowToolBar property (VBS), 951
- ShowToolBarBackgroundColor property (VBS), 951
- ShowTracker property (VBS), 951
- ShowTrendIcon property (VBS), 952
- ShowTrendIndicator property (VBS), 953

-
- ShowTrendSelection method (VBS), 1361
 - ShowValueAxis1 property (VBS), 953
 - ShowValueAxis1Label property (VBS), 953
 - ShowValueAxis2 property (VBS), 953
 - ShowValueAxis2Label property (VBS), 954
 - ShowValueTable property (VBS), 954
 - ShowWriteButton property (VBS), 954
 - ShowY1HlpLine property (VBS), 954
 - ShowY2HlpLine property (VBS), 954
 - SimpleView property (VBS), 954
 - SimpleViewToolbar property (VBS), 954
 - sin function, 1526
 - sinh function, 1526
 - Size property (VBS), 955
 - Sizeable property (VBS), 955
 - Slider object, 410
 - SmartClientView object, 414
 - SmartServer, 40
 - SmartTag object, 197, 224
 - SmartTags object (list), 195, 225
 - SmartTags property (VBS), 956
 - SortByTimeDirection property (VBS), 956
 - SortByTimeEnabled property (VBS), 957
 - SortSequence property (VBS), 957
 - SourceControl property (VBS), 958
 - SourceControlType property (VBS), 958
 - SplittedViewRatio property (VBS), 959
 - sqrt function, 1526
 - srand function, 1530
 - StartAngle property (VBS), 959
 - StartLeft property (VBS), 960
 - StartLogging, 114
 - StartNextLog, 115
 - StartProgram, 115, 1482
 - StartSequenceLog, 115
 - StartStopUpdate method (VBS), 1361
 - StartStyle property (VBS), 960
 - StartTop property (VBS), 961
 - StartValue property (VBS), 961
 - StatisticAreaColumns property (VBS), 962
 - StatisticResultColumns property (VBS), 962
 - StatusbarBackColor property (VBS), 962
 - StatusbarElementAdd property (VBS), 963
 - StatusbarElementAutoSize property (VBS), 964
 - StatusbarElementCount property (VBS), 965
 - StatusbarElementIconId property (VBS), 966
 - StatusbarElementID property (VBS), 966
 - StatusbarElementIndex property (VBS), 967
 - StatusbarElementName property (VBS), 968
 - StatusbarElementRemove property (VBS), 969
 - StatusbarElementRename property (VBS), 970
 - StatusbarElementRepos property (VBS), 971
 - StatusbarElements property (VBS), 972
 - StatusbarElementText property (VBS), 972
 - StatusbarElementTooltipText property (VBS), 973
 - StatusbarElementUserDefined property (VBS), 973
 - StatusbarElementVisible property (VBS), 974
 - StatusbarElementWidth property (VBS), 975
 - StatusbarFont property (VBS), 976
 - StatusbarFontColor property (VBS), 977
 - StatusbarShowTooltips property (VBS), 978
 - StatusbarText property (VBS), 979
 - StatusbarUseBackColor property (VBS), 979
 - StatusbarVisible property (VBS), 980
 - StatusForce object, 416
 - StatuslineFont property (VBS), 981
 - stdio functions
 - clearerr function, 1529
 - fclose function, 1528
 - feof function, 1528
 - ferror function, 1528
 - fflush function, 1528
 - fgetc function, 1528
 - fgetpos function, 1528
 - fgets function, 1528
 - fopen function, 1528
 - fputc function, 1529
 - fputs function, 1529
 - fread function, 1529
 - freopen function, 1528
 - fseek function, 1529
 - fsetpos function, 1529
 - ftell function, 1529
 - fwrite function, 1529
 - getc function, 1529
 - putc function, 1529
 - remove function, 1529
 - rename function, 1529
 - rewind function, 1529
 - setbuf function, 1529
 - setvbuf function, 1529
 - tmpfile function, 1528
 - tmpnam function, 1528
 - ungetc function, 1529
 - vfprintf function, 1529
 - vsprintf function, 1529
 - stdlib functions
 - abort function, 1530
 - abs function, 1529
 - atof function, 1529
 - atoi function, 1529
 - atol function, 1530
 - bsearch function, 1530
 - calloc function, 1530

- div function, 1529
- exit function, 1530
- free function, 1530
- getenv function, 1529
- labs function, 1530
- ldiv function, 1529
- malloc function, 1530
- qsort function, 1530
- rand function, 1530
- realloc function, 1530
- srand function, 1530
- strtod function, 1529
- strtol function, 1530
- strtoul function, 1530
- system function, 1530
- StepBackColor property (VBS), 981
- StepFont property (VBS), 982
- StepSeconds property (VBS), 983
- StepTextColor property (VBS), 983
- Stop method (VBS), 1362
- StopLogging, 117
- StopRuntime, 118, 161, 1483
- strcat function, 1530
- strchr function, 1530
- strcmp function, 1531
- strcpy function, 1530
- strcspn function, 1531
- strerror function, 1530
- strftime function, 1531
- String functions
 - strcat function, 1530
 - strchr function, 1530
 - strcmp function, 1531
 - strcpy function, 1530
 - strcspn function, 1531
 - strerror function, 1530
 - strlen function, 1531
 - strncat function, 1530
 - strncmp function, 1531
 - strncpy function, 1530
 - strpbrk function, 1530
 - strrchr function, 1530
 - strspn function, 1531
 - strstr function, 1530
 - strtok function, 1530
- strlen function, 1531
- strncat function, 1530
- strncmp function, 1531
- strncpy function, 1530
- strpbrk function, 1530
- strrchr function, 1530
- strspn function, 1531
- strstr function, 1530
- strtod function, 1529
- strtok function, 1530
- strtol function, 1530
- strtoul function, 1530
- Style property (VBS), 984
- StyleItem property (VBS), 985
- StyleSettings property (VBS), 985
- SupportsInplaceEdit property (VBS), 985
- SupportsS7DiagnosticsInSimpleView property (VBS), 985
- SupportsUserDefinedToolbarButtons property (VBS), 986
- SwapFirstWithLastConnection property (VBS), 986
- Switch object, 420
- SwitchOrientation property (VBS), 986
- SymbolicIOField object, 424
- SymbolLibrary object, 429
- SymbolTableFont property (VBS), 986
- SysDiagBuffButtonVisible property (VBS), 987
- System function,
 - SafelyRemoveHardware, 84
 - SetStartAtBoot, 120
 - WinACMPGetVersion, 119
 - WinACMPSetStartMode, 120
- System functions
 - AcknowledgeAlarm, 23
 - ActivatePLCCodeView, 24
 - ActivatePreviousScreen, 25
 - ActivateScreen, 26, 121, 1375
 - ActivateScreenByNumber, 27
 - ActivateScreenInScreenWindow, 122, 1376
 - ActivateSystemDiagnosticsView, 28
 - ArchiveLogFile, 29
 - BackupRAMFileSystem, 30
 - CalibrateTouchScreen, 31
 - ChangeConnection, 32
 - ChangeConnectionEIP, 33
 - ClearAlarmBuffer, 35
 - ClearAlarmBufferProTool, 36
 - ClearDataRecord, 37
 - ClearDataRecordMemory, 38
 - ClearLog, 39
 - CloseAllLogs, 40
 - ControlSmartServer, 40
 - ControlWebServer, 41
 - DecreaseTag, 43, 123, 1379
 - EditAlarm, 44
 - Encode, 44
 - EncodeEx, 45
 - ExportDataRecords, 46
 - ExportDataRecordsWithChecksum, 48

ExportImportUserAdministration, 51, 124
 GetBrightness, 52
 GetDataRecordFromPLC, 52
 GetDataRecordName, 54
 GetDataRecordTagsFromPLC, 56
 GetGroupNumber, 56
 GetPassword, 57
 GetUserName, 58
 GoToEnd, 58
 GoToHome, 59
 ImportDataRecords, 59
 ImportDataRecordsWithChecksum, 61
 IncreaseTag, 63, 128, 1422
 InvertBit, 65, 130, 1427
 InvertBitInTag, 66, 132, 1428
 InvertLinearScaling, 63, 129, 1425
 LinearScaling, 68, 133, 1432
 LoadDataRecord, 70
 Logoff, 71
 Logon, 71
 LookupText, 72, 135
 NotifyUserAction, 73
 OpenAllLogs, 74
 OpenCommandPrompt, 75
 OpenControlPanelDialog, 76
 OpenInternetExplorer, 77
 OpenScreenKeyboard, 77
 OpenTaskManager, 78
 PageDown, 79
 PageUp, 79
 PrintReport, 80
 PrintScreen, 80
 ResetBit, 81, 136, 1434
 ResetBitInTag, 82, 137, 1436
 SaveDataRecord, 85
 SendEmail, 86
 SetAcousticSignal, 87
 SetAlarmReportMode, 88
 SetAndGetBrightness, 89
 SetBit, 89, 139, 1438
 SetBitInTag, 91, 141, 1439
 SetBrightness, 93
 SetConnectionMode, 94
 SetDataRecordTagsToPLC, 95
 SetDataRecordToPLC, 96
 SetDaylightSavingTime, 97
 SetDeviceMode, 98
 SetDisplayMode, 98
 SetLanguage, 99, 143
 SetPropertyByConstant, 143, 1449
 SetPropertyByProperty, 145, 1451
 SetPropertyByTag, 147, 1453

SetPropertyByTagIndirect, 149, 1455
 SetRecipeTags, 101
 SetScreenKeyboardMode, 102
 SetTag, 103, 151, 1474
 SetTagByProperty, 152, 1475
 SetTagIndirect, 154, 1478
 SetTagIndirectByProperty, 153, 155, 1477, 1479
 SetTagIndirectByTagIndirect, 156, 1480
 SetTagIndirectWithOperatorInputAlarm, 1481
 SetTagWithOperatorInputAlarm, 157, 1481
 ShowAlarmWindow, 106
 ShowBlockInTiaPortalFromAlarm, 159
 ShowLogonDialog, 160
 ShowOperatorNotes, 107
 ShowPLCCodeViewFromAlarm, 160
 ShowPopupScreen, 108, 109
 ShowSlideInScreen, 111
 ShowSoftwareVersion, 112
 ShowSystemAlarm, 113
 ShowSystemDiagnosticsWindow, 113
 StartLogging, 114
 StartNextLog, 115
 StartProgram, 115
 StopLogging, 117
 StopRuntime, 118, 161, 1483
 TerminatePROFIsafe, 119
 SystemDiagnoseView object, 432

T

TabIndex property (VBS), 987
 TabIndexAlpha property (VBS), 988
 TableBackColor property (VBS), 990
 TableColor property (VBS), 991
 TableColor2 property (VBS), 991
 TableColumnsWidthAndOrder property (VBS), 992
 TableEvenRowBackColor property (VBS), 992
 TableFont property (VBS), 993
 TableForeColor property (VBS), 993
 TableForeColor2 property (VBS), 994
 TableGridLineColor property (VBS), 994
 TableHeaderBackColor property (VBS), 995
 TableHeaderBackFillStyle property (VBS), 996
 TableHeaderBorderBackColor property (VBS), 996
 TableHeaderBorderColor property (VBS), 996
 TableHeaderBorderWidth property (VBS), 997
 TableHeaderCornerRadius property (VBS), 997
 TableHeaderEdgeStyle property (VBS), 997
 TableHeaderFirstGradientColor property (VBS), 997
 TableHeaderFirstGradientOffset property (VBS), 997
 TableHeaderFont property (VBS), 997
 TableHeaderForeColor property (VBS), 997

- TableHeaderMiddleGradientColor property (VBS), 998
- TableHeaderPaddingBottom property (VBS), 998
- TableHeaderPaddingLeft property (VBS), 998
- TableHeaderPaddingRight property (VBS), 999
- TableHeaderPaddingTop property (VBS), 999
- TableHeaderSecondGradientColor property (VBS), 999
- TableHeaderSecondGradientOffset property (VBS), 999
- TableView object, 348
- Tag
 - Quality codes, 1547
- Tag object, 228
- Tag4DataRecord property (VBS), 999
- Tag4RecipeNumber property (VBS), 999
- TagForExternalTime property (VBS), 999
- TagPrefix property (VBS), 1000
- Tags, 231, 232
- Tags object (list), 232
- tan function, 1526
- tanh function, 1526
- TcpPortNr property (VBS), 1002
- Template property (VBS), 1002
- TerminatePROFIsafe, 119
- Text property (VBS), 1003
- TextAreaBottomMargin property (VBS), 1003
- TextAreaLeftMargin property (VBS), 1004
- TextAreaRightMargin property (VBS), 1004
- TextAreaTopMargin property (VBS), 1004
- TextField object, 440
- TextHandles property (VBS), 1004
- TextList property (VBS), 1004
- TextOff property (VBS), 1005
- TextOn property (VBS), 1005
- TextOrientation property (VBS), 1006
- Texts property (VBS), 1007
- TextualObjectPositions property (VBS), 1007
- TextualObjectsAutoSize property (VBS), 1007
- TextualObjectsBorderBackColor property (VBS), 1007
- TextualObjectsBorderColor property (VBS), 1008
- TextualObjectsBorderWidth property (VBS), 1008
- TextualObjectsCornerRadius property (VBS), 1008
- TextualObjectsEdgeStyle property (VBS), 1008
- TextualObjectsPaddingBottom property (VBS), 1008
- TextualObjectsPaddingLeft property (VBS), 1008
- TextualObjectsPaddingRight property (VBS), 1008
- TextualObjectsPaddingTop property (VBS), 1009
- ThumbBackColor property (VBS), 1009
- ThumbPicture property (VBS), 1009
- TickDistance property (VBS), 1010
- TicksColor property (VBS), 1010
- TickStyle property (VBS), 1011
- time function, 1531
- time functions
 - asctime function, 1531
 - ctime function, 1531
 - difftime function, 1531
 - gmtime function, 1531
 - localtime function, 1531
 - mktime function, 1531
 - strftime function, 1531
 - time function, 1531
- TimeAxes property (VBS), 1012
- TimeAxisAdd property (VBS), 1012
- TimeAxisAlignment property (VBS), 1012
- TimeAxisBegin property (VBS), 1013
- TimeAxisBeginTime property (VBS), 1013
- TimeAxisColor property (VBS), 1014
- TimeAxisCount property (VBS), 1014
- TimeAxisCountPoints property (VBS), 1015
- TimeAxisDateFormat property (VBS), 1015
- TimeAxisEnd property (VBS), 1015
- TimeAxisEndTime property (VBS), 1016
- TimeAxisIndex property (VBS), 1016
- TimeAxisInTrendColor property (VBS), 1017
- TimeAxisLabel property (VBS), 1017
- TimeAxisMeasurePoints property (VBS), 1018
- TimeAxisMode property (VBS), 1018
- TimeAxisName property (VBS), 1019
- TimeAxisOnline property (VBS), 1019
- TimeAxisRange property (VBS), 1020
- TimeAxisRangeType property (VBS), 1020
- TimeAxisRemove property (VBS), 1020
- TimeAxisRename property (VBS), 1021
- TimeAxisRepos property (VBS), 1022
- TimeAxisShowDate property (VBS), 1022
- TimeAxisSide property (VBS), 1023
- TimeAxisTimeFormat property (VBS), 1023
- TimeAxisTimeRange property (VBS), 1023
- TimeAxisTimeRangeBase property (VBS), 1024
- TimeAxisTimeRangeFactor property (VBS), 1024
- TimeAxisTrendWindow property (VBS), 1025
- TimeAxisVisible property (VBS), 1025
- TimeBase property (VBS), 1026
- TimeColumnActualize property (VBS), 1027
- TimeColumnAdd property (VBS), 1028
- TimeColumnAlignment property (VBS), 1028
- TimeColumnBackColor property (VBS), 1029
- TimeColumnBeginTime property (VBS), 1030
- TimeColumnCaption property (VBS), 1030
- TimeColumnCount property (VBS), 1031
- TimeColumnDateFormat property (VBS), 1031

- TimeColumnEndTime property (VBS), 1032
TimeColumnForeColor property (VBS), 1032
TimeColumnHideText property (VBS), 1033
TimeColumnHideTitleText property (VBS), 1034
TimeColumnIndex property (VBS), 1034
TimeColumnLength property (VBS), 1035
TimeColumnMeasurePoints property (VBS), 1035
TimeColumnName property (VBS), 1036
TimeColumnRangeType property (VBS), 1036
TimeColumnRemove property (VBS), 1037
TimeColumnRename property (VBS), 1037
TimeColumnRepos property (VBS), 1038
TimeColumns property (VBS), 1039
TimeColumnShowDate property (VBS), 1039
TimeColumnShowIcon property (VBS), 1040
TimeColumnShowTitleIcon property (VBS), 1040
TimeColumnSort property (VBS), 1041
TimeColumnSortIndex property (VBS), 1041
TimeColumnTimeFormat property (VBS), 1042
TimeColumnTimeRangeBase property (VBS), 1042
TimeColumnTimeRangeFactor property (VBS), 1043
TimeColumnUseValueColumnColors property (VBS), 1044
TimeColumnVisible property (VBS), 1044
TimeDisplayMode property (VBS), 1045
TimeStepBase property (VBS), 1046
TimeStepFactor property (VBS), 1047
TitleColor property (VBS), 1048
TitleCut property (VBS), 1049
TitleDarkShadowColor property (VBS), 1049
TitleForeColor property (VBS), 1050
TitleGridLineColor property (VBS), 1051
TitleLightShadowColor property (VBS), 1052
TitleSort property (VBS), 1053
TitleStyle property (VBS), 1054
tmpfile function, 1528
tmpnam function, 1528
Toggle property (VBS), 1054
Tolerance property (VBS), 1055
ToleranceColor property (VBS), 1055
ToleranceLowerLimit property (VBS), 1056
ToleranceLowerLimitColor property (VBS), 1057
ToleranceLowerLimitEnabled property (VBS), 1058
ToleranceLowerLimitRelative property (VBS), 1058
ToleranceUpperLimit property (VBS), 1059
ToleranceUpperLimitColor property (VBS), 1059
ToleranceUpperLimitEnabled property (VBS), 1060
ToleranceUpperLimitRelative property (VBS), 1061
tolower function, 1525
ToolBar_ButtonsHeight property (VBS), 1061
ToolBar_ButtonsWidth property (VBS), 1062
ToolBarAlignment property (VBS), 1062
ToolBarBackColor property (VBS), 1063
ToolBarBackgroundColor property (VBS), 1064
ToolBarButtonActive property (VBS), 1064
ToolBarButtonAdd property (VBS), 1065
ToolBarButtonAuthorization property (VBS), 1065
ToolBarButtonBeginGroup property (VBS), 1066
ToolBarButtonClick property (VBS), 1067
ToolBarButtonCount property (VBS), 1068
ToolBarButtonEnabled property (VBS), 1069
ToolBarButtonHeight property (VBS), 1069
ToolBarButtonHotKey property (VBS), 1070
ToolBarButtonID property (VBS), 1070
ToolBarButtonIndex property (VBS), 1071
ToolBarButtonLocked property (VBS), 1072
ToolBarButtonName property (VBS), 1073
ToolBarButtonRemove property (VBS), 1074
ToolBarButtonRename property (VBS), 1075
ToolBarButtonRepos property (VBS), 1076
ToolBarButtons property (VBS), 1077
ToolBarButtonSettings property (VBS), 1077
ToolBarButtonsForMigration property (VBS), 1077
ToolBarButtonTooltipText property (VBS), 1077
ToolBarButtonUserDefined property (VBS), 1078
ToolBarButtonVisible property (VBS), 1079
ToolBarButtonWidth property (VBS), 1080
ToolBarEnabled property (VBS), 1080
ToolBarHeight property (VBS), 1080
ToolBarIconStyle property (VBS), 1080
ToolBarLeft property (VBS), 1080
ToolBarShowTooltips property (VBS), 1080
ToolBarStyle property (VBS), 1081
ToolBarTop property (VBS), 1081
ToolBarUseBackColor property (VBS), 1081
ToolBarUseHotKeys property (VBS), 1082
ToolBarVisible property (VBS), 1083
ToolBarWidth property (VBS), 1084
ToolTipText property (VBS), 1084
Top property (VBS), 1085
TopMargin property (VBS), 1088
TopOffset property (VBS), 1088
Total property (VBS), 1089
toupper function, 1525
Trace methods, 1362
TransitionHeaderFont property (VBS), 1089
Transparency property (VBS), 1090
TransparentColor property (VBS), 1091
TransparentColorDeactivatedPicture property (VBS), 1092
TransparentColorPictureOff property (VBS), 1092
TransparentColorPictureOn property (VBS), 1093
TrendActualize property (VBS), 1094
TrendAdd property (VBS), 1094

- TrendAutoRangeBeginTagName property (VBS), 1095
 - TrendAutoRangeBeginValue property (VBS), 1095
 - TrendAutoRangeEndTagName property (VBS), 1096
 - TrendAutoRangeEndValue property (VBS), 1096
 - TrendAutoRangeSource property (VBS), 1097
 - TrendBeginTime property (VBS), 1098
 - TrendColor property (VBS), 1098
 - TrendCount property (VBS), 1099
 - TrendEndTime property (VBS), 1100
 - TrendExtendedColorSet property (VBS), 1100
 - TrendFill property (VBS), 1101
 - TrendFillColor property (VBS), 1102
 - TrendIndex property (VBS), 1103
 - TrendIndicatorColor property (VBS), 1103
 - TrendLabel property (VBS), 1104
 - TrendLineStyle property (VBS), 1105
 - TrendLineType property (VBS), 1105
 - TrendLineWidth property (VBS), 1106
 - TrendLowerLimit property (VBS), 1107
 - TrendLowerLimitColor property (VBS), 1107
 - TrendLowerLimitColoring property (VBS), 1108
 - TrendMeasurePoints property (VBS), 1109
 - TrendName property (VBS), 1109
 - TrendPointColor property (VBS), 1110
 - TrendPointStyle property (VBS), 1111
 - TrendPointWidth property (VBS), 1111
 - TrendProvider property (VBS), 1112
 - TrendProviderCLSID property (VBS), 1113
 - TrendRangeType property (VBS), 1113
 - TrendRemove property (VBS), 1114
 - TrendRename property (VBS), 1115
 - TrendRepos property (VBS), 1115
 - TrendRulerControl, 444
 - Trends property (VBS), 1116
 - TrendSelectTagName property (VBS), 1116
 - TrendSelectTagNameX property (VBS), 1117
 - TrendSelectTagNameY property (VBS), 1117
 - TrendsForPrinting property (VBS), 1118
 - TrendTagName property (VBS), 1118
 - TrendTagNameX property (VBS), 1118
 - TrendTagNameY property (VBS), 1119
 - TrendTimeAxis property (VBS), 1120
 - TrendTimeRangeBase property (VBS), 1120
 - TrendTimeRangeFactor property (VBS), 1121
 - TrendTrendWindow property (VBS), 1121
 - TrendUncertainColor property (VBS), 1122
 - TrendUncertainColoring property (VBS), 1123
 - TrendUpperLimit property (VBS), 1123
 - TrendUpperLimitColor property (VBS), 1124
 - TrendUpperLimitColoring property (VBS), 1125
 - TrendValueAlign property (VBS), 1125
 - TrendValueAxis property (VBS), 1126
 - TrendValueUnit property (VBS), 1126
 - TrendView object, 454
 - TrendVisible property (VBS), 1127
 - TrendWindowAdd property (VBS), 1127
 - TrendWindowCoarseGrid property (VBS), 1128
 - TrendWindowCoarseGridColor property (VBS), 1129
 - TrendWindowCount property (VBS), 1129
 - TrendWindowFineGrid property (VBS), 1130
 - TrendWindowFineGridColor property (VBS), 1131
 - TrendWindowForegroundTrendGrid property (VBS), 1131
 - TrendWindowGridInTrendColor property (VBS), 1132
 - TrendWindowHorizontalGrid property (VBS), 1133
 - TrendWindowIndex property (VBS), 1133
 - TrendWindowName property (VBS), 1134
 - TrendWindowRemove property (VBS), 1135
 - TrendWindowRename property (VBS), 1135
 - TrendWindowRepos property (VBS), 1136
 - TrendWindowRulerColor property (VBS), 1136
 - TrendWindowRulerLayer property (VBS), 1137
 - TrendWindowRulerStyle property (VBS), 1138
 - TrendWindowRulerWidth property (VBS), 1139
 - TrendWindows property (VBS), 1139
 - TrendWindowSpacePortion property (VBS), 1139
 - TrendWindowStatisticRulerColor property (VBS), 1140
 - TrendWindowStatisticRulerStyle property (VBS), 1141
 - TrendWindowStatisticRulerWidth property (VBS), 1142
 - TrendWindowVerticalGrid property (VBS), 1142
 - TrendWindowVisible property (VBS), 1143
 - TrendXAxis property (VBS), 1143
 - TrendYAxis property (VBS), 1144
 - TriggerOperatorEvent, 1485
 - TubeArcObject, 460
 - TubeDoubleTeeObject, 462
 - TubePolyline, 464
 - TubeTeeObject, 467
- ## U
- uaArchiveClose, 1486
 - uaArchiveDelete, 1487
 - uaArchiveExport, 1488
 - uaArchiveGetCount, 1489
 - uaArchiveGetFieldLength, 1489
 - uaArchiveGetFieldName, 1490
 - uaArchiveGetFields, 1491

uaArchiveGetFieldType, 1491
uaArchiveGetFieldValueDate, 1492
uaArchiveGetFieldValueDouble, 1493
uaArchiveGetFieldValueFloat, 1494
uaArchiveGetFieldValueLong, 1494
uaArchiveGetFieldValueString, 1495
uaArchiveGetFilter, 1496
uaArchiveGetID, 1497
uaArchiveGetName, 1497
uaArchiveGetSort, 1498
uaArchiveImport, 1499, 2306
uaArchiveInsert, 1500
uaArchiveMoveFirst, 1500
uaArchiveMoveLast, 1501
uaArchiveMoveNext, 1501
uaArchiveMovePrevious, 1502
uaArchiveOpen, 1503
uaArchiveReadTagValues, 1503
uaArchiveReadTagValuesByName, 1504
uaArchiveRequery, 1505
uaArchiveSetFieldValueDate, 1506
uaArchiveSetFieldValueDouble, 1507
uaArchiveSetFieldValueFloat, 1508
uaArchiveSetFieldValueLong, 1508
uaArchiveSetFieldValueString, 1509
uaArchiveSetFilter, 1510
uaArchiveSetSort, 1511
uaArchiveUpdate, 1512, 2308
uaArchiveWriteTagValues, 1512
uaArchiveWriteTagValuesByName, 1513
uaConnect, 1514, 2288
uaDisconnect, 1515, 2290
uaGetArchive, 1515
uaGetField, 1516
uaGetLastError, 1517
uaGetLastHResult, 1519
uaGetNumArchives, 1519
uaGetNumFields, 1520
uaQueryArchive, 1520, 2291
uaQueryArchiveByName, 1521, 2293
UaQueryConfiguration, 1522, 2295
uaReleaseArchive, 1523, 2296
uaReleaseConfiguration, 1523
ungetc function, 1529
UnhideAlarm method (VBS), 1363
Unit property (VBS), 1145
UnitColor property (VBS), 1145
UnitFont-property (VBS), 1146
UnitText property (VBS), 1146
UnitTop property (VBS), 1147
UnitViewColumnOrder property (VBS), 1148
UnlockAlarm method (VBS), 1363
UnselectRow, 1364
UnselectRow- Method (VBS), 1364
UpdateButtonVisible property (VBS), 1148
UpperLimit property (VBS), 1148
UseAutoScaling property (VBS), 1148
UseButtonFirstGradient property (VBS), 1149
UseButtonSecondGradient property (VBS), 1149
UseColumnBackColor property (VBS), 1149
UseColumnForeColor property (VBS), 1149
UseCurserKeyScroll property (VBS), 1150
Used property (VBS), 1150
UseDesignColorSchema property (VBS), 1151
UseDesignShadowSettings property (VBS), 1152
UsedPercent property (VBS), 1155
UseEyponentialFormat property (VBS), 1155
UseFirstGradient property (VBS), 1156
UseFlashTransparentColor property (VBS), 1156
UseMessageColor property (VBS), 1156
UserArchiveControl object, 469
UserView object, 481
UseScadaRendererStyle property (VBS), 1158
UseSecondGradient property (VBS), 1158
UseSelectedTitleColor property (VBS), 1158
UseSourceBackColors property (VBS), 1159
UseSourceForeColors property (VBS), 1160
UseSystemScrollbarWidth property (VBS), 1160
UseTableColor2 property (VBS), 1160
UseTableHeaderFirstGradient property (VBS), 1161
UseTableHeaderSecondGradient property (VBS), 1161
UseTagLimitColors property (VBS), 1162
UseTransparentColor property (VBS), 1162
UseTransparentColorDeactivatedPicture property (VBS), 1163
UseTransparentColorPictureOff property (VBS), 1163
UseTransparentColorPictureOn property (VBS), 1164
UseTrendNameAsLabel property (VBS), 1164
UseTwoHandOperation property (VBS), 1165
UseUdp property (VBS), 1165
UV_ColumnWidth_AKZ property (VBS), 1165
UV_ColumnWidth_Descriptor property (VBS), 1165
UV_ColumnWidth_InstallationDate property (VBS), 1166
UV_ColumnWidth_LADDR property (VBS), 1166
UV_ColumnWidth_Name property (VBS), 1166
UV_ColumnWidth_OKZ property (VBS), 1166
UV_ColumnWidth_OperationState property (VBS), 1166
UV_ColumnWidth_OrderID property (VBS), 1166
UV_ColumnWidth_ProfileID property (VBS), 1166

UV_ColumnWidth_Rack property (VBS), 1167
UV_ColumnWidth_Slot property (VBS), 1167
UV_ColumnWidth_SoftwareRevision property (VBS), 1167
UV_ColumnWidth_SpecificProfileData property (VBS), 1167
UV_ColumnWidth_State property (VBS), 1167
UV_ColumnWidth_Station property (VBS), 1167
UV_ColumnWidth_SubAddress property (VBS), 1167
UV_ColumnWidth_SubSlot property (VBS), 1168
UV_ColumnWidth_SubSystem property (VBS), 1168
UV_ColumnWidth_Type property (VBS), 1168
UV_ShowItem_AKZ property (VBS), 1168
UV_ShowItem_Descriptor property (VBS), 1168
UV_ShowItem_InstallationDate property (VBS), 1168
UV_ShowItem_LADDR property (VBS), 1168
UV_ShowItem_Name property (VBS), 1169
UV_ShowItem_OKZ property (VBS), 1169
UV_ShowItem_OperationState property (VBS), 1169
UV_ShowItem_OrderID property (VBS), 1169
UV_ShowItem_ProfileID property (VBS), 1169
UV_ShowItem_Rack property (VBS), 1169
UV_ShowItem_Slot property (VBS), 1169
UV_ShowItem_SoftwareRevision property (VBS), 1170
UV_ShowItem_SpecificProfileData property (VBS), 1170
UV_ShowItem_State property (VBS), 1170
UV_ShowItem_Station property (VBS), 1170
UV_ShowItem_SubAddress property (VBS), 1170
UV_ShowItem_SubSlot property (VBS), 1170
UV_ShowItem_SubSystem property (VBS), 1170
UV_ShowItem_Type property (VBS), 1171

V

ValueAxes property (VBS), 1172
ValueAxis1AutoRange property (VBS), 1172
ValueAxis1Begin property (VBS), 1172
ValueAxis1End property (VBS), 1172
ValueAxis1LabelLength property (VBS), 1173
ValueAxis1Style property (VBS), 1173
ValueAxis2AutoRange property (VBS), 1173
ValueAxis2Begin property (VBS), 1173
ValueAxis2End property (VBS), 1173
ValueAxis2LabelLength property (VBS), 1173
ValueAxis2Style property (VBS), 1173
ValueAxisAdd property (VBS), 1174
ValueAxisAlignment property (VBS), 1174
ValueAxisAutoPrecisions property (VBS), 1175

ValueAxisAutorange property (VBS), 1176
ValueAxisBeginValue property (VBS), 1176
ValueAxisColor property (VBS), 1177
ValueAxisCount property (VBS), 1178
ValueAxisEndValue property (VBS), 1178
ValueAxisExponentialFormat property (VBS), 1179
ValueAxisIndex-property (VBS), 1179
ValueAxisInTrendColor property (VBS), 1180
ValueAxisLabel property (VBS), 1181
ValueAxisName property (VBS), 1181
ValueAxisPrecisions property (VBS), 1182
ValueAxisRemove property (VBS), 1182
ValueAxisRename property (VBS), 1183
ValueAxisRepos property (VBS), 1183
ValueAxisScalingType property (VBS), 1184
ValueAxisTrendWindow property (VBS), 1185
ValueAxisVisible property (VBS), 1185
ValueCaption property (VBS), 1186
ValueColumnAdd property (VBS), 1186
ValueColumnAlignment property (VBS), 1186
ValueColumnAutoPrecisions property (VBS), 1187
ValueColumnBackColor property (VBS), 1188
ValueColumnCaption property (VBS), 1188
ValueColumnCount property (VBS), 1189
ValueColumnExponentialFormat property (VBS), 1189
ValueColumnForeColor property (VBS), 1190
ValueColumnHideText property (VBS), 1191
ValueColumnHideTitleText property (VBS), 1191
ValueColumnIndex-property (VBS), 1192
ValueColumnLength property (VBS), 1192
ValueColumnName property (VBS), 1193
ValueColumnPrecisions property (VBS), 1193
ValueColumnProvider property (VBS), 1194
ValueColumnProviderCLSID property (VBS), 1195
ValueColumnRemove property (VBS), 1195
ValueColumnRename property (VBS), 1196
ValueColumnRepos property (VBS), 1196
ValueColumns property (VBS), 1197
ValueColumnSelectTagName property (VBS), 1197
ValueColumnShowIcon property (VBS), 1198
ValueColumnShowTitleIcon property (VBS), 1198
ValueColumnSort property (VBS), 1199
ValueColumnSortIndex property (VBS), 1199
ValueColumnTagName property (VBS), 1200
ValueColumnTimeColumn property (VBS), 1200
ValueColumnVisible property (VBS), 1201
ValueColumnWidth property (VBS), 1202
ValueTableHeight property (VBS), 1202
ValueTableLeft property (VBS), 1202
ValueTableTop property (VBS), 1202
ValueTableWidth property (VBS), 1202

ValueY1HlpLine property (VBS), 1202
 ValueY2HlpLine property (VBS), 1202
 VBS

Object model, 186
 Reference, 186

VerticalAlignment property (VBS), 1203
 VerticalGridLines property (VBS), 1204
 VerticalPictureAlignment property (VBS), 1204
 VerticalScrollBarEnabled property (VBS), 1205
 VerticalScrollBarPosition property (VBS), 1205
 VerticalScrolling property (VBS), 1205
 VerticalScrollingEnabled property (VBS), 1206
 vfprintf function, 1529
 ViewOnly property (VBS), 1206
 ViewType property (VBS), 1206
 ViewTypeForSaveStream property (VBS), 1206
 Visible property (VBS), 1207
 VisibleItems property (VBS), 1211
 vsprintf function, 1529

W

Warning property (VBS), 1211
 WarningColor property (VBS), 1212
 WarningLowerLimit property (VBS), 1212
 WarningLowerLimitColor property (VBS), 1213
 WarningLowerLimitEnabled property (VBS), 1214
 WarningLowerLimitRelative property (VBS), 1215
 WarningRangeColor property (VBS), 1215
 WarningRangeStart property (VBS), 1216
 WarningRangeVisible property (VBS), 1217
 WarningUpperLimit property (VBS), 1217
 WarningUpperLimitColor property (VBS), 1218
 WarningUpperLimitEnabled property (VBS), 1219
 WarningUpperLimitRelative property (VBS), 1219
 Width property (VBS), 1220
 WinACMPGetVersion, 119
 WinACMPSetStartAtBoot, 120
 WinACMPSetStartMode, 120
 WinACSetStartMode, 120
 WinCC
 WinCC MediaPlayer, 343
 WindowCloseEnabled property (VBS), 1223
 WindowMaximizeEnabled property (VBS), 1223
 WindowMovingEnabled property (VBS), 1224
 WindowOnTop property (VBS), 1225
 WindowsContents property (VBS), 1225
 WindowSizingEnabled property (VBS), 1226
 WindowSlider object, 485
 WindowsStartupPosition property (VBS), 1227
 WindowsStyle property (VBS), 1227
 WlanQualityView object, 489

WriteTag method (VBS), 1368

X

XAxes property (VBS), 1228
 XAxisAdd property (VBS), 1228
 XAxisAlignment property (VBS), 1229
 XAxisAutoPrecisions property (VBS), 1229
 XAxisAutoRange property (VBS), 1230
 XAxisBeginValue property (VBS), 1231
 XAxisColor property (VBS), 1231
 XAxisCount property (VBS), 1232
 XAxisEndValue property (VBS), 1232
 XAxisExponentialFormat property (VBS), 1233
 XAxisIndex property (VBS), 1234
 XAxisInTrendColor property (VBS), 1234
 XAxisLabel property (VBS), 1235
 XAxisName property (VBS), 1235
 XAxisPrecisions property (VBS), 1236
 XAxisRemove property (VBS), 1237
 XAxisRename property (VBS), 1237
 XAxisRepos property (VBS), 1238
 XAxisScalingType property (VBS), 1238
 XAxisTrendWindow property (VBS), 1239
 XAxisVisible property (VBS), 1239

Y

YAxes property (VBS), 1240
 YAxisAdd property (VBS), 1240
 YAxisAlignment property (VBS), 1241
 YAxisAutoPrecisions property (VBS), 1241
 YAxisAutoRange property (VBS), 1242
 YAxisBeginValue property (VBS), 1242
 YAxisColor property (VBS), 1243
 YAxisCount property (VBS), 1244
 YAxisEndValue property (VBS), 1244
 YAxisExponentialFormat property (VBS), 1245
 YAxisIndex property (VBS), 1245
 YAxisInTrendColor property (VBS), 1246
 YAxisLabel property (VBS), 1247
 YAxisName property (VBS), 1247
 YAxisPrecisions property (VBS), 1248
 YAxisRemove property (VBS), 1248
 YAxisRename property (VBS), 1249
 YAxisRepos property (VBS), 1249
 YAxisScalingType property (VBS), 1250
 YAxisTrendWindow property (VBS), 1250
 YAxisVisible property (VBS), 1251

Z

ZeroPoint property (VBS), 1251
ZoneLabelView object, 490
ZoneQualityView object, 492
ZoomArea method (VBS), 1369
ZoomFactor property (VBS), 1252
ZoomInOut method (VBS), 1369
ZoomInOutTime-, 1370
ZoomInOutValues method (VBS), 1370
ZoomInOutX method (VBS), 1371
ZoomInOutY method (VBS), 1371
ZoomMove method (VBS), 1372