SIEMENS

Introduction 1

Information model 2

Address space 3

New Properties with Type Definition A

SIMATIC

SIMATIC Process Historian 2020 SIMATIC Process Historian 2020 OPC UA Information model

System Manual

02/2021
A5E35762138-AA

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.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E35762138-AA
Ⓟ 04/2021 Subject to change

Copyright © Siemens AG 2021.
All rights reserved

Table of contents

1 Introduction 5

2 Information model 7

2.1 Information model for PH Data Access 10
2.1.1 PHSystem 11
2.1.2 DataSourceSystem 12
2.1.3 DataSourceNode 12
2.1.4 Tag 13
2.1.5 EquipmentFolder 14
2.1.6 PO 14
2.1.7 HmiLoggingTag 14
2.1.8 Group 16

2.2 Information model for PH Events 16
2.2.1 PH message configuration 16
2.2.2 Message definition 18
2.2.3 Message Class 19
2.2.4 Message Type 20
2.2.5 PH Runtime Messages 20
2.2.6 PHEventType 22
2.2.7 WinCCEventType 22
2.2.8 WinCCAlarmConditionType 24
2.2.9 HmiEventType 25
2.2.10 HmiAlarmType 27
2.2.11 WinCC Message Attributes 29
2.2.12 Mapping of WinCC Message Attributes 30
2.2.13 PCS neo Message Attributes 32
2.2.14 Mapping of PCS neo Message Attributes 33

3 Address space 37

A New Properties with Type Definition 39

A.1 New Properties with Type Definition 39

Introduction

1

This document defines the information model of the OPC UA Server for Process Historian 2020 (OPC UA PH Server). The intended audience is software developers or architects that want to access following Process Historian Data in OPC UA clients:

- PH configuration data:
  - Read the Project Structure, Data Sources (DSS and DSN), Tag/Message Configuration Data
  - Read Meta-Information: Additional Information or Tag Value Contexts
- PH Historical Data:
  - Read archived tag values (Historical Data Access)
  - Read archived events (Historical Event Access)

It is assumed that the reader has basic knowledge of the OPC UA specification, in particular of Part 3 (Address Space Model) and Part 5 (Information Model).

Information model

2

The following paragraphs specify the elements of the OPC UA PH information model.

- All types defined for OPC UA PH belong to following namespace:
  - urn:Siemens/UA/Types/PH
- Only PCS neo alarm defined for OPC UA PH belong to following namespace:
  - http://Siemens.com/Ua/CHROM/Alarms
- Only local PH System exposed by the OPC UA PH Server has its own namespace:
  - http://Siemens/Opc/Ua/PHSystem

  OPC UA PH node instances (e.g. Tags) belong to the namespace of PH System.

Process Historian 2020 enables applications like SIMATIC PCS neo and SIMATIC WinCC to configure tags and messages and, at runtime, to archive tag values and events.

Fig. 1 shows an overview of the PH data model.

The PH system is the top level object representing a concrete PH server installation; the PH server may be redundant.

Each PH system contains:

- Organizational data to structure the information stored in PH by the applications
- Configuration data, i.e. tag configurations and message definitions

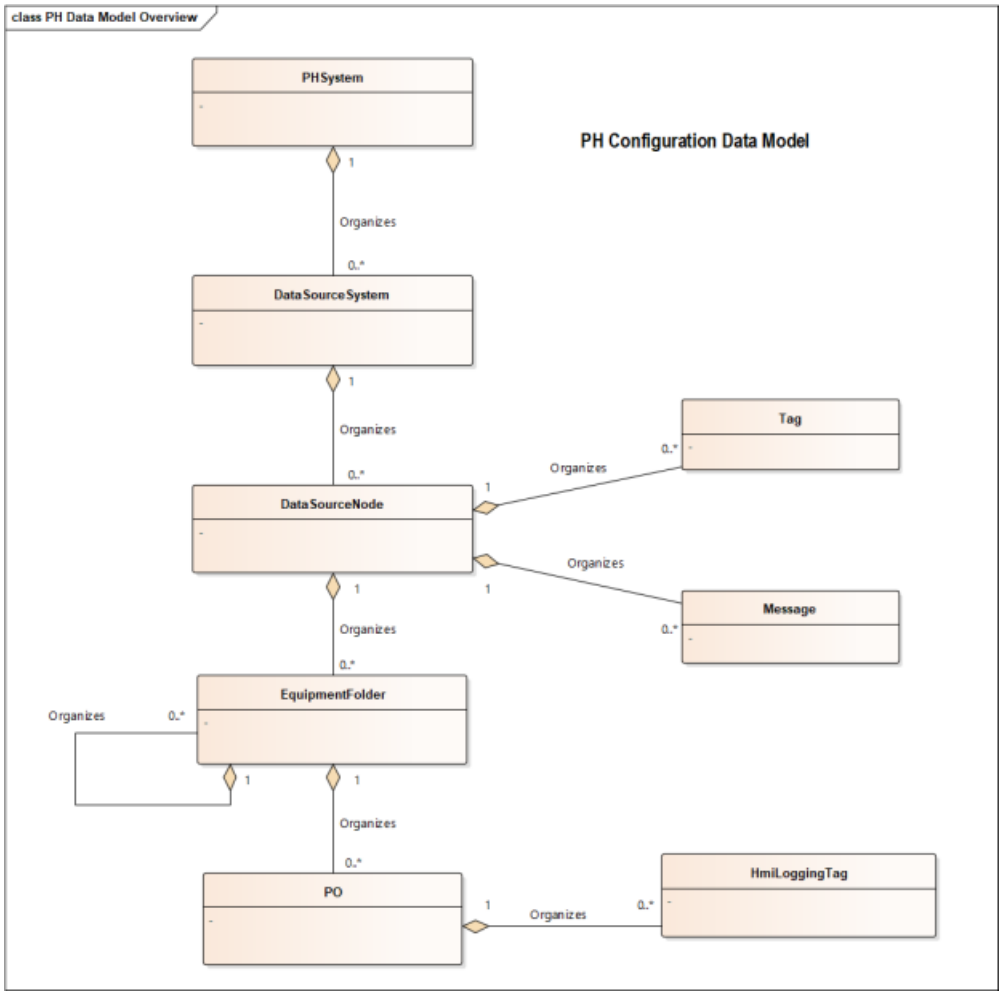


Figure 2-1 Fig. 1 Process Historian Data Model - Overview

PH Data Organization

Prior to configure and archive data, PH applications have to structure the information using following objects:

- Data Source System (DSS)

A Data Source System represents a (logical) subsystem that archives data or messages in a PH Server, e.g. a WinCC OS. This subsystem may be a standalone or a redundant system that provides data at runtime

- Data Source Node (DSN)

A Data Source Node represents a (concrete) server of a Data Source System. Thus, a redundant DSS has several Data Source Nodes.

Tag and message configurations are always related to a DSN that provides the runtime data to be archived in PH.

PH Data Configuration

Applications have to configure the data to be archived in PH by the Data Source Node(s):

Tag Configuration	A PH tag contains the configuration of a tag whose values are to be archived in PH (e.g. archiving cycle, scan cycle)
Message Definition	A PH Message Definition contains the configuration of messages to be archived in PH (e.g. priority, PLC Nr)

PH Runtime Data

PH runtime data is composed of the tag values and the events archived in PH.

It can be retrieved by OPC UA clients via the *HistoryRead* service.

2.1 Information model for PH Data Access

Information Model For PH Data Access

The next figure shows how the PH Tag Configuration is mapped to OPC UA:

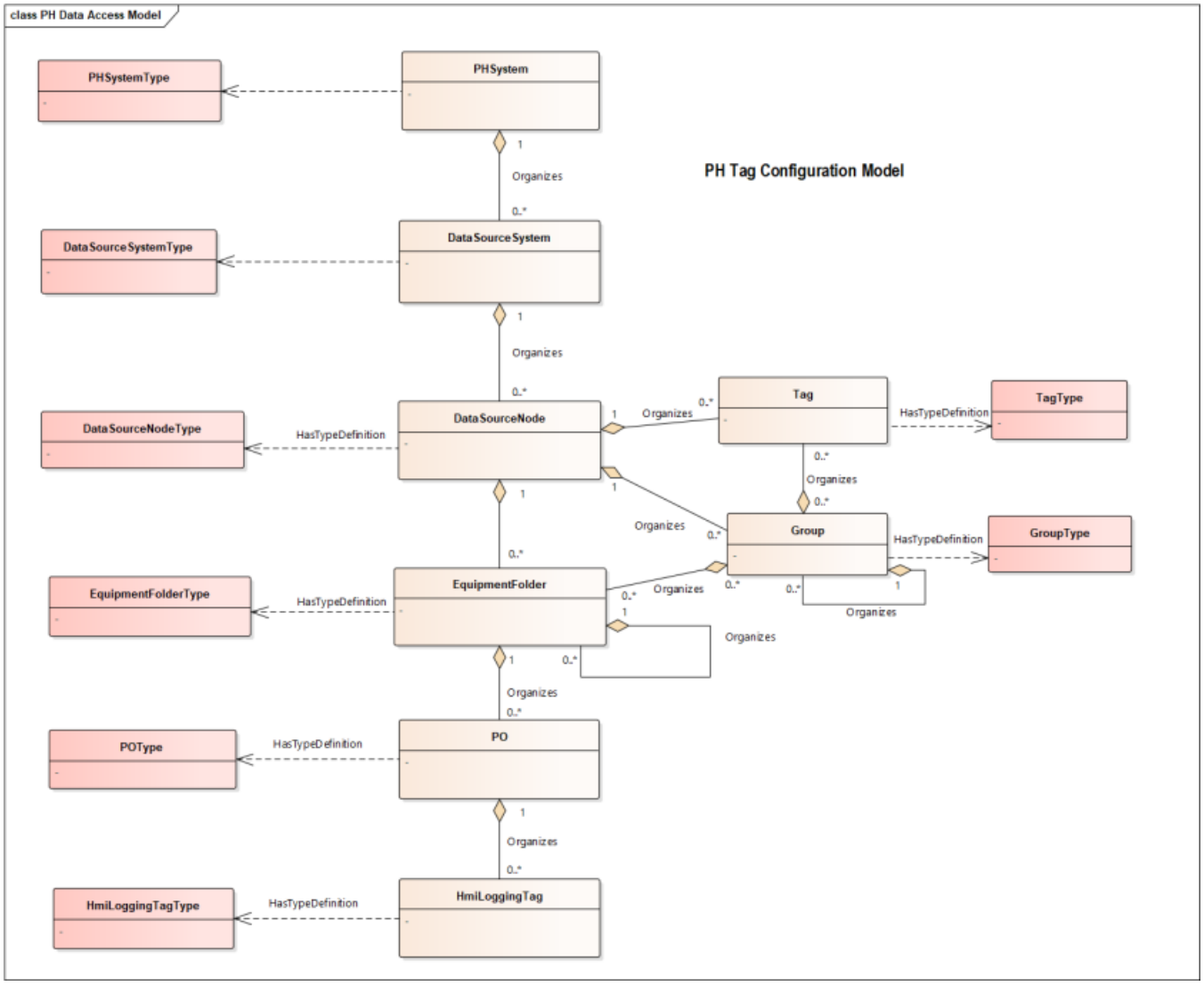


Figure 2-2 Fig. 2: OPC UA Info Model for PH Data Access

As shown in Fig. 2, following OPC UA PH types are needed to map PH Business Objects to OPC UA nodes for PH Data Access:

OPC UA Type	extends	PH Object Mapping
PHSystemType	FolderType	represents a connection to a PH Server
DataSourceSystemType	FolderType	represents a PH Data Source System
DataSourceNodeType	FolderType	represents a PH DataSourceNode
GroupType	FolderType	represents a PH Group

OPC UA Type	extends	PH Object Mapping
EquipementFolderType	FolderType	represents a PH EquipementFolder
POType	FolderType	represents a PH Process Object
TagType	BaseDataVariableType	represents a PH Tag
HmiLoggingTagType	BaseDataVariableType	represents a PH Hmi Logging Tag

Each Data Source Node contains the tag configuration of the related WinCC or PCS neo server. The PH Tag Configuration the nodes are grouped in folders "Tags" and "HMILoggingTags" for a better usability. And already grouped tags will show under the group, as shown in Fig. 3.

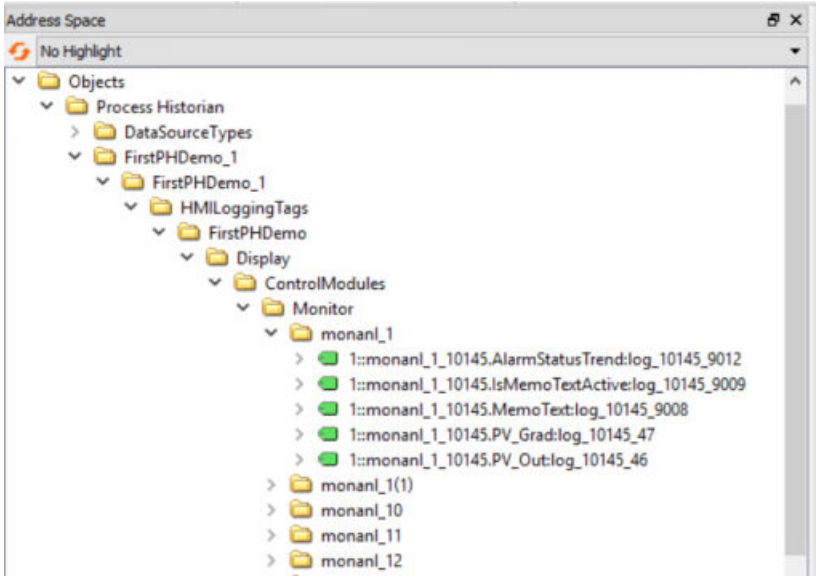


Figure 2-3 Fig. 3: PH Tag Configuration

2.1.1 PHSystem

A PHSystem node is an OPC UA object instance of PHSystemType that represents a connection to a particular PH server (that may be redundant):

PHSystem	Definition		
Attributes			
BrowseName	"Process Historian"		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	PHSystemType	ObjectType	OPC UA type of the node
HasProperty	"State"	Variable / UInt32	status of PH Server connection
HasComponent	"DataSourceTypes"	Folder	contains the Data Source Types defined in PH (e.g. "WinccClassic"; "Cpm")
HasComponent	"GroupTypes"	Folder	contains the Group Types defined in PH (e.g. "ALG Group"; "TLG Group")
HasComponent	"TagTypes"	Folder	contains the Tag Types defined in PH(e.g. "RAW"; "KPI")

PHSystem	**Definition**		
HasComponent	"UnitsofMeasure-ment"	Folder	contains the Measurement Units defined in PH
Organizes	<DataSourceSystem>	Object / DataSourceSys-temType	a PH System subsystem which contains archived data
[...]			

2.1.2 DataSourceSystem

A DataSourceSystem (DSS) node is an instance of DataSourceSystemType that maps a PH DataSourceSystem object to an OPC UA Folder:

The Data Source types available in a particular PH Server are mapped to OPC UA simple objects that are "organized by" the "DataSourceTypes" folder of the PHSystem node.

PH applications may create new DSS types.

DataSourceSystem Definition			
Attributes	Value		
BrowseName	<DataSourceSystem Name>		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	DataSourceSystemType	ObjectType	OPC UA type of the node
Organizes	<DataSourceNode>	Object / DataSourceNo-deType	Related DataSource Node
HasDataSourceType	<PH DataSourceType>	Base Object	PH DataSource type (e.g. "WinccClassic")
[...]			

2.1.3 DataSourceNode

A DataSourceNode (DSN) node is an instance of DataSourceNodeType that maps a PH DataSourceNode to an OPC UA Folder:

A DataSourceNode represents a (concrete) server of a DataSourceSystem that may be redundant: e.g. a PCS 7 project (DSS) may have 2 redundant PCS 7 OS (i.e. 2 DSN), each DSN archiving data and/or messages in the PH Server.

DataSourceNode	**Definition**		
Attributes	Value		
BrowseName	<DataSourceNode Name>		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	DataSourceNodeType	ObjectType	OPC UA type of the node
HasProperty	"NodeNumber"	Variable / Byte	DSN number (0..255) in the related DSS
Organizes	<HmiLoggingTags>	Object / FolderType	Related HmLoggingTags

DataSourceNode	Definition		
Organizes	<Tags>	Object / FolderType	Related Tags
Organizes	<Group Name>	Object / GroupType	Related PH Groups
Organizes	<Messages>	Object / FolderType	Related Messages
Organizes	<MessageTypes>	Object / FolderType	Related MessageTypes
Organizes	<MessageClasses>	Object / FolderType	Related MessageClasses
[...]			

2.1.4 Tag

A Tag node is an instance of TagType that maps a PH Tag to an OPC UA Variable.

A Tag represents the configuration of a particular tag (e.g. a WinCC tag) whose values are archived in the PH Server hence it is *HistoryReadable*.

A Tag may have a PH specific "TagType". Some tag types are predefined in PH. The tag types available in a particular PH Server are mapped to OPC UA simple objects that are "organized by" the "TagTypes" folder of the PHSystem node.

PH applications may create new PH tag types.

Tag		Definition	
Attributes	Value		
BrowseName	<Tag Name>		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	TagType	ObjectType	OPC UA type of the node
HasProperty	"ClientId"	Variable / ByteString	Tag unique Id
HasProperty	"Created"	Variable / DateTime	Date/Time of object's creation (in PH)
HasProperty	"Aggregated"	Variable / Boolean	Aggregated value
HasProperty	"ArchivingCycle"	Variable / UInt64	Archiving Cycle value
HasProperty	"Compressed"	Variable / Boolean	Compressed value
HasProperty	"CompressionCycle"	Variable / UInt64	Compression cycle value
HasProperty	"ScanCycle"	Variable / UInt64	Scan cycle value
HasProperty	"CustomData"	Variable / String	Custom Data (XML) value
HasProperty	"ProcessPontName"	Variable / String	Process Point name
HasProperty	"TechnicalName"	Variable / String	Technical name
HasTagType	<PH TagType>	Base Object	PH Tag type (e.g. "KPI")
HasUnitofMeasure-ment	<PH UnitsofMeasurementType>	Base Object	PH UnitsofMeasurement type (e.g. "Custom")
[...]			

2.1.5 EquipmentFolder

HmiLoggingTags folder organizes the equipment folders which shows the equipment hierarchy of HmiLoggingTag.

EquipmentFolder	**Definition**		
Attributes	Value		
BrowseName	<EquipmentFolder Name>		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	EquipmentFolderType	ObjectType	OPC UA type of the node
HasProperty	"EquipmentFolderType"	Variable / Integer	Node ID
HasProperty	"EquipmentProperty"	Variable / String	Value of the property type of equipment folder
Organizes	<PO>	Object / POType	Related Process Object

2.1.6 PO

Equipment folder organizes the Process Objects which shows the equipment hierarchy of HmiLoggingTag. Process Objects organizes the HmiLoggingTag.

PO	**Definition**		
Attributes	Value		
BrowseName	<Tag Name>		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	POType	ObjectType	OPC UA type of the node
HasProperty	"EquipmentNodeType"	Variable / Integer	Node ID
HasProperty	"EquipmentProperty"	Variable / String	Value of the property type of process object
HasProperty	"POId"	Variable / ByteString	PO ID
Organizes	<HmiLoggingTag>	Object / HmiLoggingTagType	Related HmiLoggingTag

2.1.7 HmiLoggingTag

HmiLoggingTag

A HmiLoggingTag node is an instance of HmiLoggingTagType that maps a PH HmiLoggingTag to an OPC UA Variable.

A HmiLoggingTag represents the configuration of a particular tag (e.g. a WinCC HMI tag) whose values are archived in the PH Server hence it is *HistoryReadable*.

HmiLoggingTag		**Definition**	
Attributes	Value		
BrowseName	<Tag Name>		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	TagType	ObjectType	OPC UA type of the node
HasProperty	"Created"	Variable / DateTime	Date/Time of tag creation (in PH)
HasProperty	"AggregationConfigDelay"	Variable / Int64	Aggregation Configuration Delay value
HasProperty	"AggregationConfigMode"	Variable / UInt8	Aggregation Configuration Mode value
HasProperty	"AggregationConfigPercentDataBad"	Variable / UInt8	Aggregation Configuration Percent Data Bad
HasProperty	"AggregationConfigPercentDataGood"	Variable / UInt8	Aggregation Configuration Percent Data Good
HasProperty	"AggregationConfigTreat"	Variable / UInt8	Aggregation Configuration Treat
HasProperty	"UncertainAsBad"	Variable / Boolean	Uncertain As Bad
HasProperty	"DisableAtStartup"	Variable / Boolean	Archiving Disabled at Startup
HasProperty	"Enabled"	Variable / Boolean	Tag enabled for archiving
HasProperty	"LimitsConfigLowLimit"	Variable / Double	Configuration Low Limit value
HasProperty	"LimitsConfigHighLimit"	Variable / Double	Configuration High Limit value
HasProperty	"LimitsConfigLimitScope"	Variable / UInt8	Configuration Limit Scope
HasProperty	"LoggingCycleFactor"	Variable / UInt8	Logging Cycle Factor
HasProperty	"LoggingCycleInterval"	Variable / Int64	Logging Cycle Interval value
HasProperty	"LoggingCycleIntervalUnit"	Variable / UInt8	Logging Cycle Interval unit value
HasProperty	"LoggingCycleStartingTime"	Variable / DateTime	Logging Cycle Starting Time
HasProperty	"LoggingCycleStartingTimeKind"	Variable / UInt8	Logging Cycle Starting Time Kind
HasProperty	"LoggingMode"	Variable / UInt8	Logging Mode value
HasProperty	"LoggingTagId"	Variable / ByteString	Logging Tag Id value
HasProperty	"LogId"	Variable /String	Log Id value
HasProperty	"SmoothingConfigDeltaValue"	Variable / Double	Configuration Delta value
HasProperty	"SmoothingConfigDeltaValueMode"	Variable / UInt8	Smoothing Config Delta Value Mode
HasProperty	"SmoothingConfigMinTime"	Variable / Int64	Smoothing Config Min Time
HasProperty	"SmoothingConfigMaxTime"	Variable / Int64	Smoothing Config Max Time
HasProperty	"SmoothingConfigSmoothingMode"	Variable / UInt8	Smoothing Config Smoothing Mode
HasProperty	"SourceLoggingTag"	Variable / Byte String	Source Logging Tag Id
HasProperty	"SourceClientId"	Variable / ByteString	Source unique Id
HasProperty	"TechnicalName"	Variable / String	Technical name
HasTagType	<PH TagType>	Base Object	PH Tag type (e.g. "KPI")
[...]			

2.1.8 Group

A Group node is an instance of GroupType that maps a PH Group to an OPC UA Folder:

Group		**Definition**	
Attributes	Value		
BrowseName	<Group Name>		
[...]			
References	Browse Name	Node Class / (Data)Type	Description
HasTypeDefinition	GroupType	ObjectType	OPC UA type of the node
HasProperty	"Created"	Variable / DateTime	Date/Time of object's creation (in PH)
Organizes	<HmiLoggingTag Name>	Object / TagType	Related HmiLoggingTag
Organizes	<Tag Name>	Object / TagType	Related Tag
Organizes	<AdditionalInfo>	Object / AdditionalInfoType	Related Additional Info
HasGroupType	<PH GroupType>	Base Object	PH group type (e.g. "TLG Archive")
[...]	[...]		

Groups are useful to organize the tags within a DataSourceNode. For ex. WinCC Tag Logging Archives are mapped to PH Groups with the specific GroupType “TLG Archive”.

A Group may have a PH specific “Group type”. Some group types are predefined in PH: the group types available in a particular PH Server are mapped to OPC UA simple objects that are “organized by” the “GroupTypes” folder of the PHSystem node.

PH applications may create new PH group types.

2.2 Information model for PH Events

The OPC UA PH server enables clients to read messages archived in PH by WinCC or other applications (HistoryRead service). Following paragraphs describe how these messages are mapped to OPC UA events. This mapping is based on the standard OPC UA event model. Further information about this model can be found in [UA_Part9: Alarms&Conditions (https://opcfoundation.org/developer-tools/specifications-unified-architecture)].

As shown in Fig. 4 in next topic, following OPC UA PH types are needed to map PH Business Objects to OPC UA nodes for PH Data Access:

OPC UA Type	**Extends**	**PH Object Mapping**
MessageDefinitionVariableType	BaseDataVariableType	Represents a PH Message
MessageClassVariableType	BaseDataVariableType	Represents a PH Message Class
MessageTypeVariableType	BaseDataVariableType	Represents a PH Message Type

2.2.1 PH message configuration

The PH message configuration partially mirrors the configuration of the WinCC message system, based on message definitions grouped in message classes and types, as described in the SIMATIC WinCC documentation.

The next figure shows how the PH Message Configuration is mapped to OPC UA:

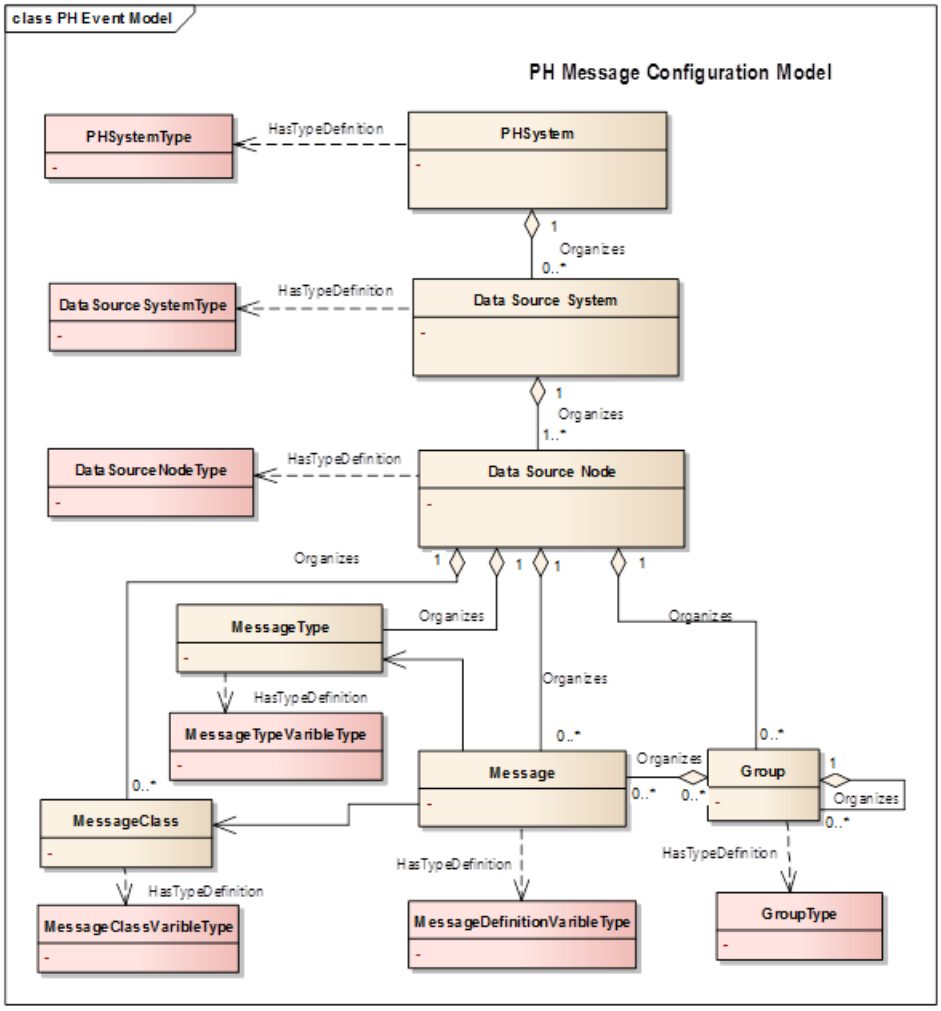


Figure 2-4 Fig. 4 PH Message Configuration Model

Each Data Source Node contains the message configuration of the related WinCC server. This message configuration is exposed in subfolders “Messages”, “MessageClasses” and “MessageTypes” for a better usability. And already grouped messages will show under the group, as shown in Fig. 5.

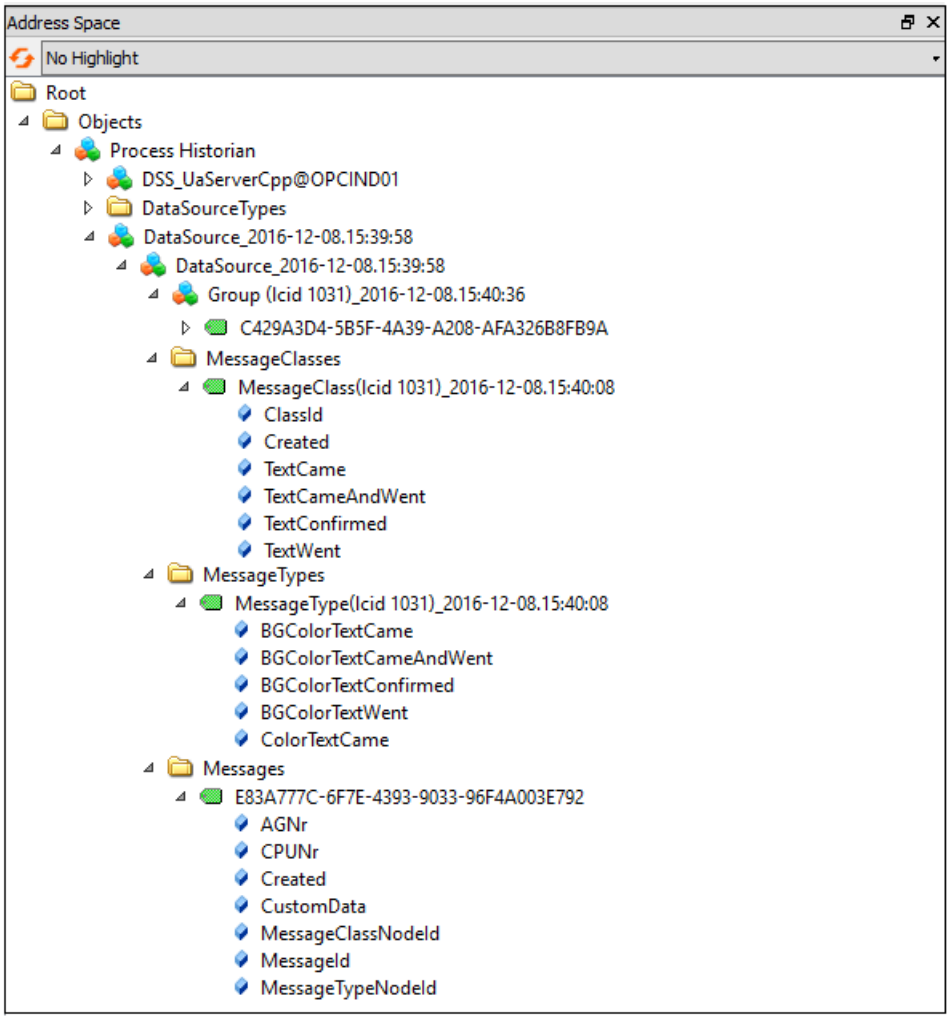


Figure 2-5 Fig. 5 PH Message Configuration Nodes

2.2.2 Message definition

Message Definitions are mapped to OPC UA variables of type **MessageDefinitionVariableType** defined below: a MessageDefinition represents the configuration of a particular WinCC message (mostly via the AlarmLogging editor).

Message Definitions may also be created by other applications archiving messages in PH: in that case, the message class and the message type may be undefined.

MessageDefinitionVariableType			**Definition**		
Attributes	Value				
BrowseName	<MessageDefinition Name>				
IsAbstact	False				
ValueRank	Scalar (-1)				
DataType	MessageDefinitionDataType				
References	Node Class	Browse Name	DataType	TypeDefinition	Description

MessageDefinitionVariableType			**Definition**		
Subtype of:	BaseDataVariableType defined in [UA_Part5]				
HasProperty	Variable	"Clientid"	Byte-String	PropertyType	Unique Id
HasProperty	Variable	"Created"	DateTime	PropertyType	Date/Time of item's creation (in PH)
HasProperty	Variable	"CustomData"	String	PropertyType	application specific information
HasProperty	Variable	"MessageClassNodeId"	NodeId	PropertyType	NodeId of related message class
HasProperty	Variable	"MessageTypeNodeId"	NodeId	PropertyType	NodeId of related message type
HasProperty	Variable	"MessageId"	Int32	PropertyType	WinCC: message number
HasProperty	Variable	"AGNr"	Int16	PropertyType	WinCC: AG (i.e. PLC) number
HasProperty	Variable	"CPUNr"	Int16	PropertyType	WinCC: CPU number
HasProperty	Variable	"MsgVar"	Int32	PropertyType	WinCC: specifies the tag triggering the message
HasProperty	Variable	"QuitBit"	Int32	PropertyType	WinCC: acknowledgment bit
HasProperty	Variable	"Params"	Int32	PropertyType	WinCC: message configuration parameters
HasProperty	Variable	"Priority"	Int32	PropertyType	WinCC: message priority

2.2.3 Message Class

Message Classes are mapped to OPC UA variables of type **MessageClassesVariableType** defined below: a MessageClass represents a class of WinCC messages (e.g. errors) as defined in the SIMATIC WinCC documentation.

MessageClassVariableType		**Definition**	
Attributes	Value		
BrowseName	<MessageClass Name>		
IsAbstact	False		
ValueRank	Scalar (-1)		
DataType	MessageClassDataType		
References	Browse Name	NodeClass / DataType	Description
Subtype of BaseDataVariableType defined in [UA_Part5]			
HasProperty	"Created"	Variable / DateTime	Date/Time of item's creation (in PH)
HasProperty	"TextCame"	Variable / LocalizedText	Display text when the message came
HasProperty	"TextCameAndWent"	Variable / LocalizedText	Display text when the message came & went
HasProperty	"TextConfirmed"	Variable / LocalizedText	Display text when the message was acknowledged
HasProperty	"TextWent"	Variable / LocalizedText	Display text when the message went
HasProperty	"ClassId"	Variable / Int32	Message class id

2.2.4 Message Type

Message Types are mapped to OPC UA variables of type **MessageTypeVariableType** defined below: a MessageType represents a type of WinCC messages (e.g. warnings, alarms) as defined in the SIMATIC WinCC documentation.

MessageTypeVariableType		**Definition**	
Attributes	Value		
BrowseName	<MessageType Name>		
IsAbstact	False		
ValueRank	Scalar (-1)		
DataType	MessageTypeDataType		
References	Browse Name	NodeClass / DataType	Description
Subtype of BaseDataVariableType defined in [UA_Part5]			
HasProperty	"Created"	Variable / DateTime	Date/Time of item's creation (in PH)
HasProperty	"ColorTextCame"	Variable / Int32	Font color for message status "came in"
HasProperty	"ColorTextCameAndWent"	Variable / Int32	Font color for message status "came and went"
HasProperty	"ColorTextConfirmed"	Variable / Int32	Font color for message status "Acknowledged"
HasProperty	"ColorTextWent"	Variable / Int32	Font color for message status "went"
HasProperty	"BGColorTextCame"	Variable / Int32	Background color for message status "came in"
HasProperty	"BGColorTextCame AndWent"	Variable / Int32	Background color for message status "came and went"
HasProperty	"BGColorTextConfirmed"	Variable / Int32	Background color for message status "Acknowledged"
HasProperty	"BGColorTextWent"	Variable / Int32	Background color for message status "went"
HasProperty	"TypeId"	Variable / Int16	Message Type Id

2.2.5 PH Runtime Messages

As shown in Fig. 6, runtime messages archived in PH are mapped to OPC UA events of following types:

PHEventType	Base event type for “simple” messages archived in PH by any application (except WinCC)
WinCCEventType	Event type for “simple” WinCC messages archived in PH, i.e. messages with following acknowledgment philosophy: • “Acknowledgment ‘came in’” is not checked (not acknowledgeable) • “without status ‘went out’” is checked (message may only come in) e.g. “Motor was started” or “Motor was stopped” are simple messages.
WinCCAlarmCondition-Type	Event type for conditional WinCC messages (i.e. messages that come & go or must be acknowledged), e.g. WinCC alarms or warnings
HmiEventType	Event type for "simple" hmi logged alarm states archived in PH by PCS neo
HmiAlarmType	Event type for conditional PCS neo hmi logged alarm states

Note: There is no configuration data for Hmi Events and Hmi Alarms from PCS neo.

Not only WinCC or PCS neo messages may be stored in Process Historian. Other applications may also archive messages that are mapped to PHEventType.

Simple WinCC messages are mapped to WinCCEventType that extends PHEventType with specific WinCC event properties. WinCC condition messages are mapped to WinCCAlarmConditionType.

Each PH event is related to its message definition via the “MessageDefinitionId” property.

Simple PCS neo hmi logged alarm states are mapped to HmiEventType. And condition hmi logged alarm states are mapped to HmiAlarmType.

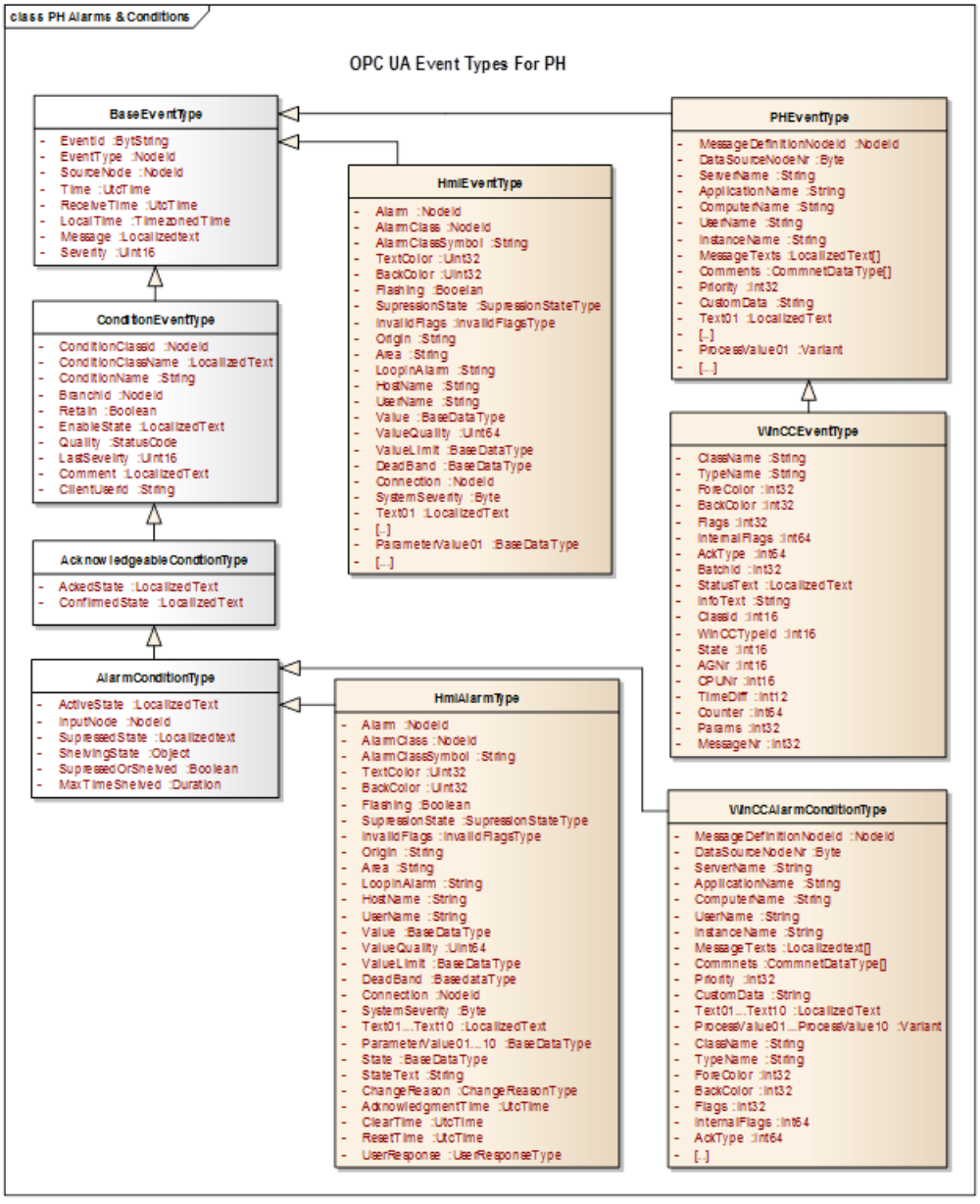


Figure 2-6 Fig. 6 OPC UA Information Model for PH Events

2.2.6 PHEventType

PHEventType			**Definition**		
Attributes	Value				
Browse-Name	PHEventType				
IsAbstact	False				
References	Node Class	Browse Name	DataType	TypeDefinition	Description
Subtype of:	BaseEventType defined in [UA_Part5]				
HasProperty	Variable	"Message DefinitionNodeId"	NodeId	PropertyType	Date/Time of item's creation (in PH)
HasProperty	Variable	"DataSource NodeNr"	Byte	PropertyType	Number of related Data Source Node
HasProperty	Variable	"ServerName"	String	PropertyType	Name of the server the event originated from
HasProperty	Variable	"ApplicationName"	String	PropertyType	Name of the application generating the event
HasProperty	Variable	"ComputerName"	String	PropertyType	Name of the computer the event originated from
HasProperty	Variable	"UserName"	String	PropertyType	Name of the user
HasProperty	Variable	"InstanceName"	String	PropertyType	Name of the instance
HasProperty	Variable	"MessageTexts"	LocalizedText [ ]	PropertyType	Array of message texts
HasProperty	Variable	"Comments"	Comment DataType[ ]	PropertyType	Array of comments
HasProperty	Variable	"ProcessValues"	Variant[ ]	PropertyType	Array of process values
HasProperty	Variable	"Priority"	Int32	PropertyType	Message priority
HasProperty	Variable	"CustomData"	String	PropertyType	Application specific information
HasProperty	Variable	"Text01"	LocalizedText	PropertyType	Message text #1
HasProperty	Variable	"Text02"	LocalizedText	PropertyType	Message text #2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"Text10"	LocalizedText	PropertyType	Message text #10
HasProperty	Variable	"ProcessValue01"	Variant	PropertyType	Process value#1
HasProperty	Variable	"ProcessValue02"	Variant	PropertyType	Process value#2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"ProcessValue10"	Variant	PropertyType	Process value#10

2.2.7 WinCCEventType

WinCCEventType		**Definition**
Attributes	Value	
Browse-Name	WinCCEventType	

WinCCEventType			**Definition**		
IsAbstact	False				
References	Node Class	Browse Name	DataType	TypeDefinition	Description
Subtype of:	PHEventType				
HasProperty	Variable	"ClassName"	String	PropertyType	WinCC message class name
HasProperty	Variable	"TypeName"	String	PropertyType	WinCC message type name
HasProperty	Variable	"ForeColor"	Int32	PropertyType	Text color for the message display
HasProperty	Variable	"BackColor"	Int32	PropertyType	Background color for the message display
HasProperty	Variable	"Flags"	Int64	PropertyType	Indicates whether the message requires acknowledgment
HasProperty	Variable	"InternalFlags"	Int64	PropertyType	
HasProperty	Variable	"AckType"	Int64	PropertyType	WinCC acknowledgement philosophy
HasProperty	Variable	"BatchId"	Int32	PropertyType	
HasProperty	Variable	"StatusText"	LocalizedText	PropertyType	Message status text
HasProperty	Variable	"InfoText"	String	PropertyType	Message info text
HasProperty	Variable	"ClassId"	Int16	PropertyType	WinCC message class id
HasProperty	Variable	"TypeId"	Int16	PropertyType	WinCC message type id
HasProperty	Variable	"State"	Int16	PropertyType	WinCC message state
HasProperty	Variable	"AGNr"	Int16	PropertyType	WinCC message AG (i.e. PLC) number
HasProperty	Variable	"CPUNr"	Int16	PropertyType	WinCC message CPU number
HasProperty	Variable	"TimeDiff"	Int32	PropertyType	Specifies the period between incoming, outgoing, and acknowledgment of a message
HasProperty	Variable	"Counter"	Int64	PropertyType	WinCC message counter
HasProperty	Variable	"Params"	Int32	PropertyType	WinCC message configuration parameters
HasProperty	Variable	"MessageNr"	Int32	PropertyType	WinC message number

2.2.8 WinCCAlarmConditionType

WinCCAlarmConditionType			**Definition**		
Attributes	Value				
Browse Name	WinCCAlarmConditionType				
IsAbstact	False				
References	Node Class	Browse Name	DataType	Type Definition	Description
Subtype of:	BaseEventType				
HasProperty	Variable	"MessageDefinition NodeId"	NodeId	PropertyType	Date/Time of item's creation (in PH)
HasProperty	Variable	"DataSourceNodeNr"	Byte	PropertyType	Number of related Data Source Node
HasProperty	Variable	"ServerName"	String	PropertyType	Name of the server the event originated from
HasProperty	Variable	"ApplicationName"	String	PropertyType	Name of the application generating the event
HasProperty	Variable	"ComputerName"	String	PropertyType	Name of the computer the event originated from
HasProperty	Variable	"UserName"	String	PropertyType	Name of the user
HasProperty	Variable	"InstanceName"	String	PropertyType	Name of the instance
HasProperty	Variable	"MessageTexts"	LocalizedText [ ]	PropertyType	Array of message texts
HasProperty	Variable	"Comments"	Comment DataType[ ]	PropertyType	Array of comments
HasProperty	Variable	"Priority"	Int32	PropertyType	WinCC: Message priority
HasProperty	Variable	"CustomData"	String	PropertyType	Application specific information
HasProperty	Variable	"Text01"	LocalizedText	PropertyType	Message text #1
HasProperty	Variable	"Text02"	LocalizedText	PropertyType	Message text #2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"Text10"	LocalizedText	PropertyType	Message text #10
HasProperty	Variable	"ProcessValue01"	Variant	PropertyType	Process value#1
HasProperty	Variable	"ProcessValue02"	Variant	PropertyType	Process value#2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"ProcessValue10"	Variant	PropertyType	Process value#10
HasProperty	Variable	"ClassName"	String	PropertyType	WinCC message class name
HasProperty	Variable	"TypeName"	String	PropertyType	WinCC message type name
HasProperty	Variable	"ForeColor"	Int32	PropertyType	Text color for the message display
HasProperty	Variable	"BackColor"	Int32	PropertyType	Background color for the message display
HasProperty	Variable	"Flags"	Int64	PropertyType	Indicates whether the message requires acknowledgment
HasProperty	Variable	"InternalFlags"	Int64	PropertyType	
HasProperty	Variable	"AckType"	Int64	PropertyType	WinCC acknowledgement philosophy
HasProperty	Variable	"BatchId"	Int32	PropertyType	

WinCCAlarmConditionType			**Definition**		
HasProperty	Variable	"StatusText"	LocalizedText	PropertyType	Message status text
HasProperty	Variable	"InfoText"	String	PropertyType	Message info text
HasProperty	Variable	"ClassId"	Int16	PropertyType	WinCC message class id
HasProperty	Variable	"TypeId"	Int16	PropertyType	WinCC message type id
HasProperty	Variable	"State"	Int16	PropertyType	WinCC message state
HasProperty	Variable	"AGNr"	Int16	PropertyType	WinCC message AG (i.e. PLC) number
HasProperty	Variable	"CPUNr"	Int16	PropertyType	WinCC message CPU number
HasProperty	Variable	"TimeDiff"	Int32	PropertyType	Specifies the period between incoming, outgoing, and acknowledgment of a message
HasProperty	Variable	"Counter"	Int64	PropertyType	WinCC message counter
HasProperty	Variable	"Params"	Int32	PropertyType	WinCC message configuration parameters
HasProperty	Variable	"MessageNr"	Int32	PropertyType	WinCC message number

2.2.9 HmiEventType

HmiEventType

HmiEventType			**Definition**		
Attributes	Value				
Browse Name	HmiEventType				
IsAbstact	False				
References	Node Class	Browse Name	DataType	Type Definition	Description
Subtype of:	BaseEventType defined in [UA_Part5]				
HasProperty	Variable	"InstanceId"	UInt32	PropertyType	Instance index used for referencing an active, multi-instance alarm in the scope of the (configured) HmiAlarm
HasProperty	Variable	"Alarm"	NodeId	PropertyType	Reference to the corresponding HmiAlarm
HasProperty	Variable	"AlarmClass"	NodeId	PropertyType	Reference to the alarm class, may differ from the alarm class reference as defined by the associated HmiAlarm
HasProperty	Variable	"AlarmClassSymbol"	String	PropertyType	Symbol (abbreviation) of the referenced AlarmClass
HasProperty	Variable	"TextColor"	UInt32	PropertyType	Text color
HasProperty	Variable	"BackColor"	UInt32	PropertyType	Back color
HasProperty	Variable	"Flashing"	Boolean	PropertyType	Flashing

HmiEventType			**Definition**		
HasProperty	Variable	"SupressionState"	SupressionStateType	PropertyType	States whether this alarm is unsuppressed, suppressed, or shelved
HasProperty	Variable	"InvalidFlags"	InvalidFlagsType	PropertyType	Indicator of invalid attribute values, bitwise interpretation
HasProperty	Variable	"Origin"	String	PropertyType	Dynamic alarm instance specific name of the alarm generating object
HasProperty	Variable	"Area"	String	PropertyType	Dynamic alarm instance specific name of the area to which the alarm generating object is related to
HasProperty	Variable	"LoopInAlarm"	String	PropertyType	Function that is called when navigating from the alarm window to the e.g. a screen that reflects the origin (source) of the alarm, or an application that provides more information
HasProperty	Variable	"HostName"	String	PropertyType	Name of the machine that hosts the originator of the alarm
HasProperty	Variable	"UserName"	String	PropertyType	Name of the User related with the event (operator-input alarms only)
HasProperty	Variable	"Value"	BaseDataType	PropertyType	Current value when the alarm was raised; updated value when the alarm cleared
HasProperty	Variable	"ValueQuality"	UInt64	PropertyType	Current quality of the value when the alarm was raised
HasProperty	Variable	"ValueLimit"	BaseDataType	PropertyType	Current limit of the value when the alarm was raised; updated value limit when the alarm cleared in case of dynamic limits
HasProperty	Variable	"DeadBand"	BaseDataType	PropertyType	Deadband value of the analog alarm condition when the alarm was raised
HasProperty	Variable	"Connection"	NodeId	PropertyType	Reference to the corresponding HMI connection
HasProperty	Variable	"SystemSeverity"	Byte	PropertyType	Severity for system voting (redundancy), which is based on alarms
HasProperty	Variable	"Text01"	LocalizedText	PropertyType	Additional multi-lingual text #1
HasProperty	Variable	"Text02"	LocalizedText	PropertyType	Additional multi-lingual text #2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"Text10"	LocalizedText	PropertyType	Additional multi-lingual text #10
HasProperty	Variable	"ParameterValue01"	BaseDataType	PropertyType	Alarm Parameter value#1
HasProperty	Variable	"ParameterValue02"	BaseDataType	PropertyType	Alarm Parameter value#2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"ParameterValue16"	BaseDataType	PropertyType	Alarm Parameter value#16

2.2.10 HmiAlarmType

HmiAlarmType

HmiAlarmType			Definition		
Attributes	Value				
Browse Name	HmiAlarmType				
IsAbstact	False				
References	Node Class	Browse Name	DataType	Type Definition	Description
Subtype of:	BaseEventType defined in [UA_Part5]				
HasProperty	Variable	"InstanceId"	UInt32	PropertyType	Instance index used for referencing an active, multi-instance alarm in the scope of the (configured) HmiAlarm
HasProperty	Variable	"Alarm"	NodeId	PropertyType	Reference to the corresponding HmiAlarm
HasProperty	Variable	"AlarmClass"	NodeId	PropertyType	Reference to the alarm class, may differ from the alarm class reference as defined by the associated HmiAlarm
HasProperty	Variable	"AlarmClassSymbol"	String	PropertyType	Symbol (abbreviation) of the referenced AlarmClass
HasProperty	Variable	"TextColor"	UInt32	PropertyType	Text color
HasProperty	Variable	"BackColor"	UInt32	PropertyType	Back color
HasProperty	Variable	"Flashing"	Boolean	PropertyType	Flashing
HasProperty	Variable	"SupressionState"	SupressionStateType	PropertyType	States whether this alarm is unsuppressed, suppressed, or shelved
HasProperty	Variable	"InvalidFlags"	InvalidFlagsType	PropertyType	Indicator of invalid attribute values, bitwise interpretation
HasProperty	Variable	"Origin"	String	PropertyType	Dynamic alarm instance specific name of the alarm generating object
HasProperty	Variable	"Area"	String	PropertyType	Dynamic alarm instance specific name of the area to which the alarm generating object is related to
HasProperty	Variable	"LoopInAlarm"	String	PropertyType	Function that is called when navigating from the alarm window to the e.g. a screen that reflects the origin (source) of the alarm, or an application that provides more information
HasProperty	Variable	"HostName"	String	PropertyType	Name of the machine that hosts the originator of the alarm

HmiAlarmType			**Definition**		
HasProperty	Variable	"UserName"	String	PropertyType	Name of the User related with the event (operator-input alarms only)
HasProperty	Variable	"Value"	BaseDataType	PropertyType	Current value when the alarm was raised; updated value when the alarm cleared
HasProperty	Variable	"ValueQuality"	UInt64	PropertyType	Current quality of the value when the alarm was raised
HasProperty	Variable	"ValueLimit"	BaseDataType	PropertyType	Current limit of the value when the alarm was raised; updated value limit when the alarm cleared in case of dynamic limits
HasProperty	Variable	"DeadBand"	BaseDataType	PropertyType	Deadband value of the analog alarm condition when the alarm was raised
HasProperty	Variable	"Connection"	NodeId	PropertyType	Reference to the corresponding HMI connection
HasProperty	Variable	"State"	Byte	PropertyType	Textual representation of the alarm state
HasProperty	Variable	"StateText"	String	PropertyType	Textual representation of the alarm state
HasProperty	Variable	"ChangeReason"	ChangeReasonType	PropertyType	Reason for Modification time, see enumeration definition
HasProperty	Variable	"AcknowledgementTime"	UtcTime	PropertyType	Timestamp when the alarm has been ack'ed at the source (or service in case that the alarm source doesn't provide Acknowledgment)
HasProperty	Variable	"ClearTime"	UtcTime	PropertyType	Timestamp when the alarm has been cleared at the source (or service in case that the alarm source doesn't provide date and time information)
HasProperty	Variable	"ResetTime"	UtcTime	PropertyType	Timestamp when the alarm has been reset (or service in case that the alarm source doesn't provide date and time information)
HasProperty	Variable	"UserResponse"	UserResponseType	PropertyType	The type for variable that represents a property of another node
HasProperty	Variable	"Text01"	LocalizedText	PropertyType	Additional multi-lingual text #1
HasProperty	Variable	"Text02"	LocalizedText	PropertyType	Additional multi-lingual text #2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"Text10"	LocalizedText	PropertyType	Additional multi-lingual text #10
HasProperty	Variable	"ParameterValue01"	BaseDataType	PropertyType	Alarm Parameter value#1

HmiAlarmType			Definition		
HasProperty	Variable	"ParameterValue02"	BaseDataType	PropertyType	Alarm Parameter value#2
[...]	[...]	[...]	[...]	[...]	[...]
HasProperty	Variable	"ParameterValue16"	BaseDataType	PropertyType	Alarm Parameter value#16

2.2.11 WinCC Message Attributes

WinCC Message Attributes

The following table lists the WinCC attributes of the WinCC message system. The attributes are configured in the WinCC message system.

NOTE: This table only contains the attributes that WinCC archives to PH Server.

WinCC Attribute	Data Type	Comment
ClassName	String	WinCC message class name
TypeName	String	WinCC message type name
ForeColor	Int32	Text color for the message display
BackColor	Int32	Background color for the message display
FlashColor	Int32	Flashing color for the message display
Flags	Int32	Indicates whether the message requires acknowledgment
Text01	String	Content of the user text block #1
Text02	String	Content of the user text block #2
...		...
Text10	String	Content of the user text block #10
ProcessValue01		Message process value #1
ProcessValue02		Message process value #2
...		...
ProcessValue10		Message process value #10
StateText	String	status of the message
InfoText	String	Information text for the message
LoopInAlarm	Int32	Indicates whether loop-in alarm is configured
ClassId	Int32	WinCC message class id
TypeId	Int32	WinCC message type id
ModifyState	Int32	Value of the state variable of the message
AGNr	Int16	Indicates the number of the automation system on which the message occurred
CPUNr	Int16	Specifies the CPU number on which the message occurred
Duration	Int32	Specifies the period between incoming, outgoing, and acknowledgment of a message
Counter	Int32	Specifies the number of messages after the start of Runtime
QuickStateText	String	Specifies whether the message has been acknowledged
QuitCount	Int32	Specifies the number of open, unacknowledged messages
Parameter	Int32	Specifies the configuration parameters for the message
BlockInfo	String	Current content of the message block

AlarmCount	Int32	Indicates the number of pending messages
LockCount	Int32	Number of blocked messages
Priority	Int32	Specifies the configured priority of the message
Application	String	Specifies the application that triggered the message
Computer	String	Specifies the computer name that edited the message
User	String	Specifies the user who edited the message
Comment	String	Comment of the message

NOTE: PH is providing some WinCC Attributes like ClassName, TypeName, ForeColor, BackColor, Flags etc in *"CustomDataXml"*.

2.2.12 Mapping of WinCC Message Attributes

OPC UA Event Types

WinCC messages are mapped to following specific OPC UA event types:

WinCCEventType

This type is derived from BaseEventType and corresponds to "simple" events, i.e. WinCC messages with following acknowledgment philosophy:

- "Acknowledgment Came In" is not activated
- "Message without Status Went Out" is activated.

Example: events like "motor was started", "motor was stopped" are simple events.

WinCCAlarmConditionType

This type is derived from AlarmConditionType and corresponds to WinCC messages that are conditional, e.g. acknowledgeable messages or messages that may come and go.

Example: events like "pressure is too low", "temperature exceeds 80°C"

Both event types contain the WinCC specific message attributes as OPC UA properties.

(e.g. "CUPNr", "AGNr").

Message class and type

The WinCC message system informs the user of disturbances and operating conditions in the process. A WinCC message always belongs to a specific message class and message type specified via the ClassId, TypeId, ClassName, and TypeName properties.

Priority

WinCC messages may have a priority between "0" and "16".

The OPC UA event's severity is a value between "1" and "1000", "1" being the lowest and "1000" the highest severity.

By default, the WinCC message priority "0" will be mapped to severity "1" and priority "16" to severity "1000". Other priority values are linearly interpolated to severities between "1" and "1000".

Mapping of WinCC message attributes

Following table shows the mapping of WinCC message attributes with OPC UA event properties/ attributes:

OPC UA – Attribute / Property	**Mapping to WinCC Message Attribute**
for all Event Types:	
EventId	Unique event identifier (Message value Id from PH)
EventType	NodeId of the PHEventType, WinCCEventType or WinCCAlarmConditionType node
SourceName	Server name
Message	MessageTexts[1]
Time	Timestamp (UTC)
Severity	Priority (see above)
SourceNode	Not applicable or Not archived or Unable to retrieve from PH
ReceiveTime	Not applicable or Not archived or Unable to retrieve from PH
LocalTime	Not applicable or Not archived or Unable to retrieve from PH
for PH Event Types:	
MessageDefinitionNodeId	MessageDefinition NodeId (From PH System Namespace)
ServerName	ServerName
InstanceName	InstanceName
MessageTexts	MessageTexts
Comments	Comments
Priority	Priority
CustomData	CustomDataXml (Not there in WinCC, PH is providing)
Text01...Text10	Text01...Text10
ProcessValue01...ProcessValue10	ProcesValue01... ProcesValue10
DataSourceNodeNr	Not applicable or Not archived or Unable to retrieve from PH
ApplicationName	Not applicable or Not archived or Unable to retrieve from PH
ComputerName	Not applicable or Not archived or Unable to retrieve from PH
UserName	Not applicable or Not archived or Unable to retrieve from PH
for WinCC Event Types:	
Flags	Flags
ClassName	ClassName (from CustomDataXml)
TypeName	TypeName (from CustomDataXml)
ForeColor	ForeColor (from CustomDataXml)
BackColor	BackColor (from CustomDataXml)
AckType	AckType (from CustomDataXml)
StatusText	StatusText (from CustomDataXml)
InfoText	InfoText (from CustomDataXml)
State	State (from CustomDataXml)
AGNr	AGNr (from CustomDataXml)
CPUNr	CPUNr (from CustomDataXml)

Counter	Counter (from CustomDataXml)
Params	Params (from CustomDataXml)
InternalFlags	Flags1 (from CustomDataXml)
BatchId	Not applicable or Not archived or Unable to retrieve from PH
ClassID	Not applicable or Not archived or Unable to retrieve from PH
WinCCTypeId	Not applicable or Not archived or Unable to retrieve from PH
TimeDiff	Not applicable or Not archived or Unable to retrieve from PH
MessageNr	Not applicable or Not archived or Unable to retrieve from PH
for WinCCAlarmConditionType:	
ConditionClassId	NodeId of the "ProcessConditionClassType" node
ConditionClassName	"ProcessConditionClassType"
ConditionName	WinCC message class name
Comment	Comments[1]
Retain	True for a pending event
Quality	Message quality
EnabledState/Id	True if the message is enabled
ActiveState/Id	True if the message came in
AckedState/Id	True if the message was acknowledged
ClientUserId	Not applicable or Not archived or Unable to retrieve from PH
BranchId	Not applicable or Not archived or Unable to retrieve from PH
LastSeverity	Not applicable or Not archived or Unable to retrieve from PH
ConfirmedState	Not applicable or Not archived or Unable to retrieve from PH
InputNode	Not applicable or Not archived or Unable to retrieve from PH
SupressedState	Not applicable or Not archived or Unable to retrieve from PH
ShelvingState	Not applicable or Not archived or Unable to retrieve from PH
SupressedOrShelved	Not applicable or Not archived or Unable to retrieve from PH
MaxTimeShelved	Not applicable or Not archived or Unable to retrieve from PH

2.2.13 PCS neo Message Attributes

PCS neo Message Attributes

The following table lists the PCS neo attributes of the PCS neo message system. The attributes are configured in the PCS neo message system.

NOTE: This table only contains the attributes that PCS neo archives to PH Server.

PCS neo Attribute	Data type	Comment
InstanceID	UDInt	Active alarm counter to group all alarm state objects of the same logged active alarm object that belong together.
AlarmClass	String	HMI logged alarm class name
AlarmClassSymbol	String	Symbol (abbreviation) of the referenced Alarm Class

ParameterValue01... ParameterValue16	Variant	Alarm parameter values
Text01...Text09	String	Multilingual alarm texts
Area	String	Dynamic alarm instance specific name of the area to which the alarm generating object is related to
BackColor	UDInt	Back color for the message display
ChangeReason	UInt	Reason for LoggedAlarmState entry
Commnets	String	Alarm comments
Enabled	Bool	Alarm active state
Group	String	Name of the associated alarm group
InfoText	String	Multilingual information text; typically used for operator instructions
InvalidFlags	UInt16	Indicator of invalid attribute values, bitwise interpretation
Name	String	Unique name of the LoggedAlarm inside the HmiAlarmLog
Origin	String	Dynamic alarm instance specific name of the alarm generating object
Priority	UInt32	Priority of the alarm at time of alarm state change
Quality	USInt	Quality information of the alarm state
Source	Ref	Source Hmi Alarm
SourceType	USInt	Defines the alarm generation method
State	USInt	Alarm state machine definition of Hmi Alarm
StateText	String	Multilingual alarm state text
SuppressionState	USInt	Alarm suppression state
TextColor	UDInt	Text color for the message display
TimeStamp	DateTime	Alarm logged time
UserName	String	Name of the User related with the event
Value	Variant	Value that caused the alarm
Limit	Variant	Limit exceeded when the alarm was raised
ValueQuality	UInt	Quality of the value that caused the alarm

2.2.14 Mapping of PCS neo Message Attributes

OPC UA Event Types

PCS neo messages are mapped to following specific OPC UA event types:

HmiEventType:

This type is derived from BaseEventType and corresponds to “simple” events.

HmiAlarmType:

This type is derived from AlarmConditionType and corresponds to PCS neo messages that are conditional, e.g. acknowledgeable messages.

Priority

See WinCC Priority mapping.

Following table shows the mapping of PCS neo message attributes with OPC UA event properties/attributes:

OPC UA – Attribute / Property	**Mapping to** PCS neo **Message Attribute**
for all Hmi Event Types:	
EventId	Unique event identifier(LoggedAlarmId of PH + TimeStamp)
EventType	NodeId of the HmiEventType or HmiAlarmType
SourceName	Name
Message	Text01
Time	TimeStamp
Severity	Priority(see above)
SourceNode	Not applicable or Not archived or Unable to retrieve from PH
ReceiveTime	Not applicable or Not archived or Unable to retrieve from PH
LocalTime	Not applicable or Not archived or Unable to retrieve from PH
AlarmClassSymbol	AlarmClassSymbol
TextColor	TextColor
BackColor	BackColor
SupressionState	SupressionState
Text01...Text09	Text01... Text09
ParameterValue01...ParameterValue16	ParameterValue01... ParameterValue16
InvalidFlags	InvalidFlag
Origin	Origin
Area	Area
Value	Value
ValueQuality	ValueQuality
ValueLimit	Limit
UserName	UserName
Connection	Not applicable or Not archived or Unable to retrieve from PH
Flashing	Not applicable or Not archived or Unable to retrieve from PH
InstanceId	Not applicable or Not archived or Unable to retrieve from PH
Alarm	Not applicable or Not archived or Unable to retrieve from PH
AlarmClass	Not applicable or Not archived or Unable to retrieve from PH
LoopInAlarm	Not applicable or Not archived or Unable to retrieve from PH
HostName	Not applicable or Not archived or Unable to retrieve from PH
DeadBand	Not applicable or Not archived or Unable to retrieve from PH
SystemSeverity	Not applicable or Not archived or Unable to retrieve from PH
Only for HmiAlarmType:	
ConditionClassId	NodeId of the “ProcessConditionClassType“ node
ConditionClassName	“ProcessConditionClassType“
ConditionName	AlarmClass name
Comment	Comment[1]

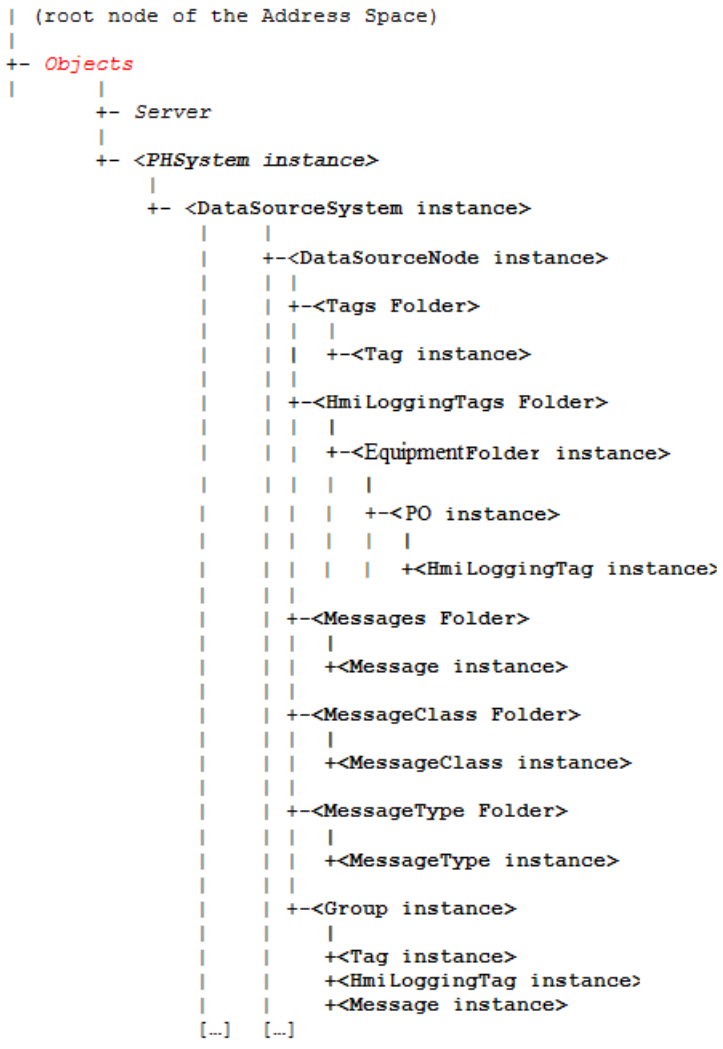
Retain	True for a pending event
Quality	Quality
ClientUserId	User
EnabledState/Id	True if the Alarm is enabled
ActiveState/Id	True if the Alarm came in
AckedState/Id	True if the Alarm was acknowledged
SupressedState	SupressionState
ShelvingState	SupressionState
SupressedOrShelved	SupressionState
BranchId	Not applicable or Not archived or Unable to retrieve from PH
LastSeverity	Not applicable or Not archived or Unable to retrieve from PH
ConfirmedState	Not applicable or Not archived or Unable to retrieve from PH
InputNode	Not applicable or Not archived or Unable to retrieve from PH
MaxTimeShelved	Not applicable or Not archived or Unable to retrieve from PH
State	State
StateText	StateText
ChangeReason	ChangeReason
AcknowledgementTime	Not applicable or Not archived or Unable to retrieve from PH
ClearTime	Not applicable or Not archived or Unable to retrieve from PH
ResetTime	Not applicable or Not archived or Unable to retrieve from PH
UserResponse	Not applicable or Not archived or Unable to retrieve from PH

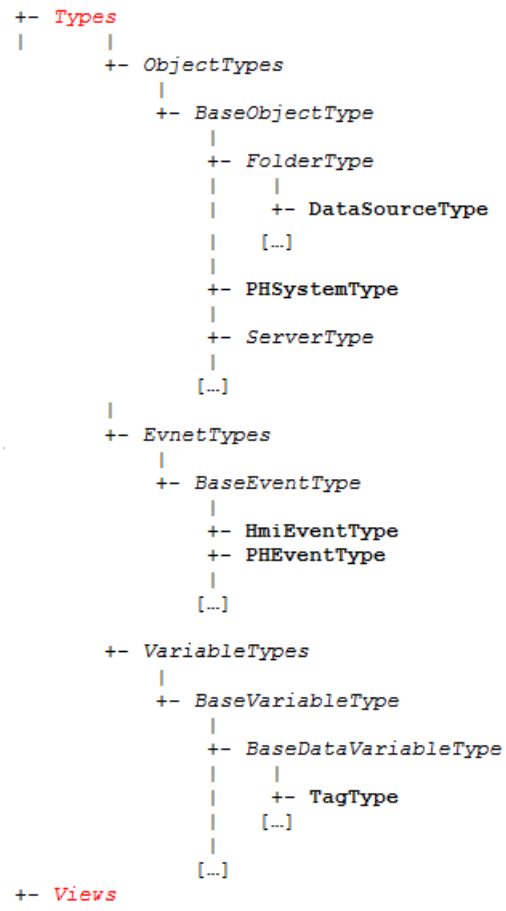
NOTE: PH server does not support condition instances, so the *ConditionId* is not supported for historical events.

Address space

3

The OPC UA PH server's Address Space is organized as follows:





The top level nodes of the address space are specified by the OPC UA norm:

- The "Types" folder contains all standard UA types and also the specific UA types defined in this document
- The "Objects" folder contains the standard "Server" node (with general information like server state or capabilities) and all specific node instances the server exposes to clients (e.g. PH System node and all of its subcomponents)
- The "Views" folder is reserved for OPC UA Views, if any. This OPC UA PH version does not support views.

A PH server is represented by an UA object of type PHSystemType in the "Objects" folder.

NOTE: In current version, there is only one PHSystem node because the OPC UA PH server only connects to the local PH server.

New Properties with Type Definition

A

A.1 New Properties with Type Definition

ClientId

```
typedef struct { uint8_t id[40]; } PhClientId;
```

ArchiveValueType

```
typedef enum : uint16_t
  {
    /// <summary>
    /// Unknown value type (invalid)
    /// </summary>
    PhArchiveValueType_Unknown = 0x0000,
    /// <summary>
    /// Tag value of type <c>bool</c>.
    /// </summary>
    PhArchiveValueType_Bool = 0x0001,
    /// <summary>
    /// Tag value of type <c>int8_t</c>.
    /// </summary>
    PhArchiveValueType_Int8 = 0x0002,
    /// <summary>
    /// Tag value of type <c>int16_t</c>.
    /// </summary>
    PhArchiveValueType_Int16 = 0x0003,
    /// <summary>
    /// Tag value of type <c>int32_t</c>.
    /// </summary>
    PhArchiveValueType_Int32 = 0x0004,
    /// <summary>
    /// Tag value of type <c>int64_t</c>.
    /// </summary>
    PhArchiveValueType_Int64 = 0x0005,
    /// <summary>
    /// Tag value of type <c>uint8_t</c>.
    /// </summary>
    PhArchiveValueType_Uint8 = 0x0006,
    /// <summary>
    /// Tag value of type <c>uint16_t</c>.
    /// </summary>
    PhArchiveValueType_Uint16 = 0x0007,
    /// <summary>
    /// Tag value of type <c>uint32_t</c>.
    /// </summary>
    PhArchiveValueType_Uint32 = 0x0008,
    /// <summary>
    /// Tag value of type <c>uint64_t</c>.
    /// </summary>
    PhArchiveValueType_Uint64 = 0x0009,
    /// <summary>
    /// Tag value of type <c>float</c>.
    /// </summary>
    PhArchiveValueType_Float = 0x000A,
    /// <summary>
    /// Tag value of type <c>double</c>.
    /// </summary>
    PhArchiveValueType_Double = 0x000B,
    /// <summary>
    /// Tag value of type <c>int64_t</c> containing a tick count.
    /// </summary>
    PhArchiveValueType_Ticks = 0x000C,
    /// <summary>
    /// Tag value of type <c>FILETIME</c>.
    /// </summary>
    PhArchiveValueType_FileTime = 0x000D,
```

UnitsofMeasurement

```
class IPhUnitOfMeasurement : public IPhBusinessObject
 {
```

```
    public:
};
```

UnitOfMeasurementUid

```
struct PhNullableGuid
 {
    PhGuid Value;
    PhBool HasValue;
 };
```

Note: Some properties of HmiLoggingTag has Data Types like "PhNullableInt32", "IPhString", "PhNullableDouble".

```
struct PhNullableInt32
 {
    int32_t Value;
    PhBool HasValue;
 };
```

```
struct PhNullableDouble
 {
    int64_t Value;
    PhBool HasValue;
 };
```

```
class IPhString : public IPhUnknown
 {
    public:
    virtual wchar_t const * PHCALLTYPE Str() const = 0;
    virtual int32_t PHCALLTYPE Length() const = 0;
 };
```

These are directly mapped to basic Data Types ignoring the properties like "HasValue" as shown below:

- PhNullableInt64 = Int64
- PhNullableUInt8 = Uint8
- IPhString = String