SIEMENS

Introduction 1

Using basic syntax 2

Using expert syntax 3

SIMATIC WinCC

Unified SCADA
WinCC Unified Open Pipe

Operating Manual

A5E48117715-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
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E48117715-AA
℗ 10/2019 Subject to change

Table of contents

1 Introduction 5

2 Using basic syntax 9

2.1 Basics of basic syntax 9

2.2 Commands 10
2.2.1 SubscribeTagValue 10
2.2.2 UnsubscribeTagValue 11
2.2.3 ReadTagValue 12
2.2.4 WriteTagValue 12
2.2.5 SetCharSet 13

2.3 Reference 14

3 Using expert syntax 15

3.1 Basics of expert syntax 15

3.2 Commands 17
3.2.1 SubscribeTag 17
3.2.2 UnsubscribeTag 19
3.2.3 ReadTag 19
3.2.4 WriteTag 20
3.2.5 SubscribeAlarm 21
3.2.6 UnsubscribeAlarm 23
3.2.7 ReadAlarm 23

3.3 Reference 25

3.4 Syntax of the alarm filter 30

Introduction

1

Welcome to WinCC Unified Open Pipe

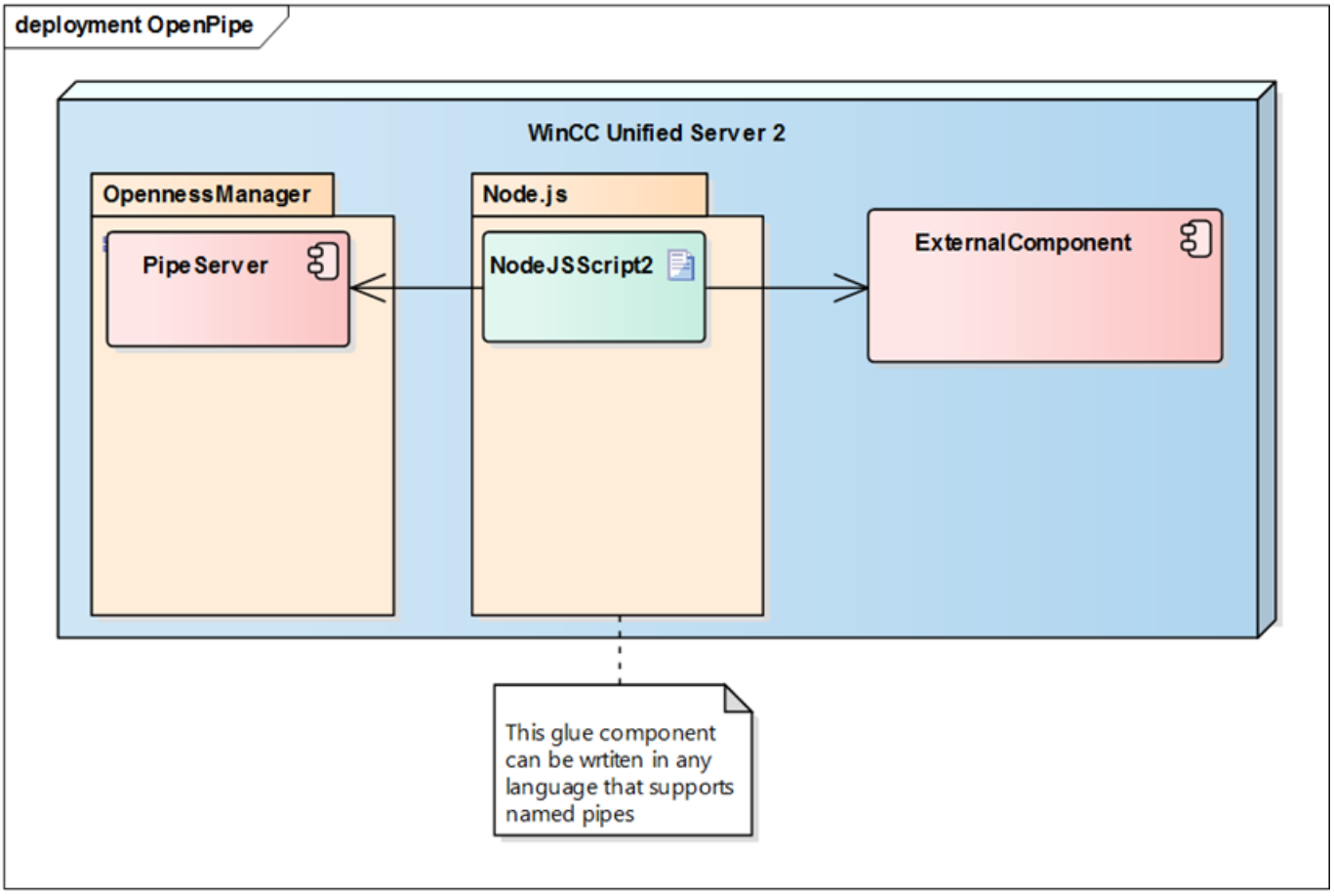
WinCC Unified Open Pipe is an Openness concept based on pipe technology to connect a customer application to WinCC Unified RT.

Compared to Openness RT (ODK), WinCC Unified Open Pipe offers a limited number of functions. However, the connection code can be written in any programming language that supports pipe technology. Even batch access to the pipe is possible.

The main aim of WinCC Unified Open Pipe is to make it possible for you to connect an existing application to WinCC Unified RT with little workload and independent of the programming language. The available commands let you communicate with WinCC Unified RT using tags and alarms.

Pipe technology

The pipe is a data stream with buffer between two processes that works according to the FIFO principle (First In First Out). One process is provided by WinCC Unified RT (OpennessManager). It creates the pipe and processes the requests of the customer application. The second process is the customer application. It connects to the pipe by using its name, sends requests and receives responses.



Name of the pipe:

- Under Windows: "\\.\pipe\HmiRuntime"
- Under Linux: "/tmp/HmiRuntime"

As soon as the pipe is open, single-line commands can be sent; they must end in a line break ("\n" or "\r\n"). Responses are returned using the same pipe instance.

Basic syntax and expert syntax

Two types of syntax are available for WinCC Unified Open Pipe:

- Basic syntax
  You use the basic syntax when you are working with basic batch files (e.g. in a CMD.exe or batch).
- Expert syntax
  You use the expert syntax when you are working with scripts or programming languages that have a JSON parser, e.g. Python, Node.js or Powershell.

Samples

You will find the following file on the installation medium: "Support\Openness \Siemens.Unified.Openness_SDK_<version number>.zip"

Extract the file locally to any directory on your computer. You will then find examples of the use of WinCC Unified Open Pipe in the subfolder "OpenPipe\Samples".

Using basic syntax

2

2.1 Basics of basic syntax

Characteristics

The basic syntax has the following characteristics:

- Simple text-based syntax without JSON parts.
- Only commands for individual objects, e.g. write access to a single tag.
- Object names may not include special characters or space characters.
- No cookies.

Structure of a request

```
<command> <object> [value]
```

- `Command`: Command name, e.g. "WriteTagValue"
- `Object`: Name of the object for which the command is called, e.g. "Tag1"
- `Value`: Input value, e.g. "True"

Structure of a response

OnSuccess:

```
Notify<Command> <Object> [Value1…ValueN]
```

- `Command`: Command name, e.g. "ReadTagValue"
- `Object`: Name of the object for which the command was called, e.g. "Tag1"
- `Value1…ValueN`: For example, value and quality of the read tags

OnError:

```
Error<Command> <Object> <Error text>
```

- `Command`: Command name, e.g. "ReadTagValue"
- `Object`: Name of the object for which the command was called, e.g. "Tag1"
- `Error text`: Detailed error description

Overview of the commands of the simple syntax

Command	OnSuccess	OnError	Description
`SubscribeTagValue <Tag>`	`NotifySubscribeTagValue <Tag> <Quality> <Value>`	`ErrorSubscribeTagValue <Tag> <Error text>` `ErrorNotifyTagValue <Tag> <Error text>`	Subscribe tag for monitoring. `Quality`={good, bad, uncertain}
`UnsubscribeTagValue <Tag>`	`NotifyUnsubscribeTagValue <Tag>`	`ErrorUnsubscribeTagValue <Tag> <Error text>`	Unsubscribe tag from monitoring.
`ReadTagValue <Tag>`	`NotifyReadTagValue <Tag> <Quality> <Value>`	`ErrorReadTagValue <Tag> <Error text>`	Reads the tag value from the device.
`WriteTagValue <Tag> <Value>`	`NotifyWriteTagValue <Tag>`	`ErrorWriteTagValue <Tag> <Error text>`	Writes the value to the tag.
`SetCharSet <Value>`	`NotifySetCharSet <Value>`	`ErrorSetCharSet <Value> <Error text>`	Changes to a different character coding.

Errors and error texts

This help only provides a selection of possible error messages. The error texts can also differ from the texts in your projects.

2.2 Commands

2.2.1 SubscribeTagValue

Description

The command "SubscribeTagValue" subscribes the specified tag for monitoring.

Error handling

- When the tag value contains a line break (\n), the value cannot be signaled. An error is signaled.
- When the same tag is subscribed a second time for monitoring, an error is signaled.
- A global monitoring error is signaled with "ErrorSubscribeTagValue". Because no monitoring was set up, there is no need to unsubscribe the tag from monitoring.
- An error relating to the tag value is signaled with "ErrorNotifyTagValue". Monitoring is set up in this case, but the tag value cannot be signaled for various reasons. Unsubscribe the tag from monitoring when it is no longer needed.

Request

`SubscribeTagValue <Tag>`

For example: `SubscribeTagValue Tag_1`

Response

OnSuccess (or partial success):

```
NotifySubscribeTagValue <Tag> <Quality> <Value>
```

For example:

- `NotifySubscribeTagValue Tag_1 Uncertain 0`
- `NotifySubscribeTagValue Tag_1 Good 10`
- `NotifySubscribeTagValue Tag_1 Bad 12`

OnError:

- Global error:
  `ErrorSubscribeTagValue <Tag> <Error text>`
  For example:
  - `ErrorSubscribeTagValue Tag_1 Tag does not exist`
  - `ErrorSubscribeTagValue Tag_1 Subscription already exists`
- Error for tag value:
  `ErrorNotifyTagValue <Tag> <Error text>`
  For example:
  - `ErrorNotifyTagValue Tag_1 Encoding error`
  - `ErrorNotifyTagValue Tag_1 Value contains newline`

2.2.2 UnsubscribeTagValue

Description

The "UnsubscribeTagValue" command unsubscribes a tag from monitoring.

Request

```
UnsubscribeTagValue <Tag>
```

For example: `UnsubscribeTagValue Tag_1`

Response

OnSuccess:

```
NotifyUnsubscribeTagValue <Tag>
```

For example:

- `NotifyUnsubscribeTagValue Tag_1`
- `NotifyUnsubscribeTagValue Tag_1`
- `NotifyUnsubscribeTagValue Tag_1`

OnError:

```
ErrorUnsubscribeTagValue <Tag> <Error text>
```

For example: `ErrorUnsubscribeTagValue Tag_1 Subscription does not exist`

2.2.3 ReadTagValue

Description

The "ReadTagValue" command reads the value of a tag from the device. Only the tag value and the quality are signaled.

When the tag value contains a line break (\n), the value cannot be signaled. An error is signaled.

Request

`ReadTagValue <Tag>`

For example: `ReadTagValue Tag_1`

Response

OnSuccess (or partial success):

`NotifyReadTagValue <Tag> <Quality> <Value>`

For example:

- `NotifyReadTagValue Tag_1 Uncertain 0`
- `NotifyReadTagValue Tag_1 Good 10`
- `NotifyReadTagValue Tag_1 Bad 12`

OnError:

`ErrorReadTagValue <Tag> <Error text>`

For example:

- `ErrorReadTagValue Tag_1 Tag does not exist`
- `ErrorReadTagValue Tag_1 Encoding error`
- `ErrorReadTagValue Tag_1 Value contains newline`

2.2.4 WriteTagValue

Description

The "WriteTagValue" command writes a value to a single tag.

When the transferred tag value contains a line break (\n), only the partial string in front of the line break is written to the tag.

Request

`WriteTagValue <Tag> <Value>`

For example: `WriteTagValue Motor.Label MC001`

Response

OnSuccess (or partial success):

`NotifyWriteTagValue <Tag>`

For example: `NotifyWriteTagValue Motor.Label`

OnError:

`ErrorWriteTagValue <Tag> <Error text>`

For example: `ErrorWriteTagValue Motor.Label Tag does not exist`

2.2.5 SetCharSet

Description

Sets the character encoding to one of the following specified values: {UTF-8, cp437, cp850}

The default character encoding is UTF-8.

The following character coding values must be supported as a minimum:

- UTF-8
- cp850
  In German Windows systems, this is the default for the SystemLocale.
- cp437
  In US Windows systems, this is the default for the SystemLocale.

Internally, strings are treated as Unicode strings (often as UTF-16 in a CFSTR). For external communication over the pipe, the Unicode characters must be converted into a byte representation.

When a character cannot be converted for a specific character coding (e.g. the Greek character "π" in the character coding cp437), an "encoding error" is triggered.

Request

`SetCharSet <Value>`

For example: `SetCharSet UTF-8`

Response

OnSuccess:

`NotifySetCharSet <Value>`

For example: `NotifySetCharSet UTF-8`

OnError:

`ErrorSetCharSet <Value> <Error text>`

For example: `ErrorSetCharSet UTF-9 unknown character set`

2.3 Reference

The following section contains a reference of the properties of tags that you get with the command ReadTagValue.

The commands transfer the property values as string.

Tag properties

"Name" property

Name of the tag

"Value" property

Value of the tag at the moment of the read operation.

"Quality" property

Quality of the read operation of the tag

Possible values:

- "Good"
- "Bad"
- "Uncertain"

"ErrorDescription" property

Description of the error code of the last read or write operation of the tag

Using expert syntax

3

3.1 Basics of expert syntax

Characteristics

The expert syntax has the following characteristics:

- Complete JSON commands and replies.
- Commands for individual objects and multiple objects, for example, to write multiple tags in one call.
- Character coding is always UTF-8.
- Cookies are available and mandatory.

Structure of a request

```
{
  "Message": "<Command>",
  "Params":
  {
      "<Object name>":
      [
        "<Param1>",
        "<Param2>"
      ]
  },
  "ClientCookie": "<Cookie name>"
}
```

Structure of a response

OnSuccess

```
{
  "Message": "Notify<Command>",
  "Params":
  {
      "<Object name>":
    [
```

```
      {
        "<Value1>"
      },
      {
        "<Value2>"
      }
    ]
  },
  "ClientCookie": "<Cookie name>"
}
```

OnError

```
{
 "Message": "Error<Command>",
 "ErrorCode": "<Error code>",
 "ErrorDescription": "<Error description>",
 "ClientCookie": "<Cookie name>"
}
```

Overview of the commands of the expert syntax

Command	OnSuccess	OnError	Description
`SubscribeTag <Tag> <ClientCookie>`	`NotifySubscribeTag <Tag name> <Value> <TimeStamp> <Quality> <QualityCode> <hasChanged> <Error code> <Error text> <ClientCookie>`	`ErrorSubscribeTag <Error code> <Error text> <ClientCookie>`	Subscribes one or more tags for monitoring.
`UnsubscribeTag <ClientCookie>`	`NotifyUnsubscribeTag <ClientCookie>`	`ErrorUnsubscribeTag <Error code> <Error text> <ClientCookie>`	Unsubscribes the tag or tags from the cookie from monitoring.
`ReadTag <Tags> <ClientCookie>`	`NotifyReadTag <Tag name> <Value> <TimeStamp> <Quality> <QualityCode> <Errorcode> <Error text> <ClientCookie>`	`ErrorReadTag <Error code> <Error text> <ClientCookie>`	Reads the value of one or more tags of the device.

Command	OnSuccess	OnError	Description
`WriteTag <Tags, Values> <ClientCookie>`	`NotifyWriteTag <Tag name> <Error code> <Error text> <ClientCookie>`	`ErrorWriteTag <Error code> <Error text> <ClientCookie>`	Writes the specified values to the specified tags.
`SubscribeAlarm <Filter> <Systems> <Language> <ClientCookie>`	`NotifySubscribeAlarm <ClientCookie> <Alarms>`	`ErrorSubscribeAlarm <Error code> <Error text> <ClientCookie>`	Subscribes the alarms defined over the filter, the system and the language ID for monitoring.
`UnsubscribeAlarm <ClientCookie>`	`NotifyUnsubscribeAlarm <ClientCookie>`	`Error UnsubscribeAlarm <Error code> <Error text> <ClientCookie>`	Unsubscribes the alarms from the cookie from monitoring.
`ReadAlarm <Filter> <Systems> <Language> <ClientCookie>`	`NotifyReadAlarm <ClientCookie> <Alarms>`	`ErrorReadAlarm <Error code> <Error text> <ClientCookie>`	Reads the alarms defined over the filter, the system and the language ID.

Errors and error texts

This help only provides a selection of possible error messages. The error texts can also differ from the texts in your projects.

Attributes of tags and alarms

You can find a description of the attributes of the tags and alarms in the help document Runtime - Open Development Kit (ODK).

3.2 Commands

3.2.1 SubscribeTag

Description

The "SubscribeTag" command subscribes one or more tags for monitoring. The following properties are monitored:

- Tag value
- Quality
- Quality code
- Time stamp

"NotifiySubscribeTag" always returns all monitored tags, even if only the value of one monitored tag changes. The change can be a change to the quality code, time stamp or tag value. The order of tags in the response corresponds to the order in the "SubscribeTag" request.

It is permitted to have the same tag monitored by multiple "SubscribeTag" calls.

Request

```
{"Message":"SubscribeTag","Params":{"Tags":
["<Tag>","<Tag>"]},"ClientCookie":"<Cookie>"}
```

- `Tags`: List of the tags to be monitored
- `ClientCookie`: Is used for "UnsubscribeTag" and to assign the notification to its monitoring.

For example:

```
{"Message":"SubscribeTag","Params":{"Tags":
["Tag_0","Tag_1"]},"ClientCookie":"mySubscription1"}
```

Response

OnSuccess

```
{"Message":"NotifySubscribeTag", "Params":{"Tags":[{"Name":"<Tag>",
"Quality":"<Value>", "QualityCode":"<Value>",
"TimeStamp":"<Value>", "Value":"<Tag value>", "ErrorCode":<Value>,
"ErrorDescription":"<Error text>"}, {"Name":"<Tag>",
"Quality":"<Value>", "QualityCode":"<Value>",
"TimeStamp":"<Value>", "Value":"<Tag value>", "ErrorCode":<Value>,
"ErrorDescription":"<Error text>"}]}, "ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"NotifySubscribeTag", "Params":{"Tags":[{"Name":"Tag_0",
"Quality":"Good", "QualityCode":"192",
"TimeStamp":"2019-01-30T11:25:35Z", "Value":"16", "ErrorCode":0,
"ErrorDescription":""}, {"Name":"Tag_1", "Quality":"Uncertain",
"QualityCode":"76", "TimeStamp":"2019-01-30T11:25:35Z",
"Value":"1", "ErrorCode":-2147483620, "ErrorDescription":"Tag does
not exist"}]}, "ClientCookie":"mySubscription1"}
```

OnError

```
{"Message":"ErrorSubscribeTag", "ErrorCode":<Value>,
"ErrorDescription":"<Error text>", "ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"ErrorSubscribeTag", "ErrorCode":-2147483621,
"ErrorDescription":"Subscription could not be created",
"ClientCookie":"mySubscription1"}
```

3.2.2 UnsubscribeTag

Description

The "UnsubscribeTag" command unsubscribes a tag from monitoring that was started with the cookie transferred in the call of "SubscribeTag".

Request

```
{"Message":"UnsubscribeTag","ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"UnsubscribeTag","ClientCookie":"mySubscription1"}
```

Response

OnSuccess

```
{"Message":"NotifyUnsubscribeTag","ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"NotifyUnsubscribeTag","ClientCookie":"mySubscription1"}
```

OnError

```
{"Message":"ErrorUnsubscribeTag", "ErrorCode":<Value>,
"ErrorDescription":"<Error text>", "ClientCookie":"Cookie"}
```

For example:

```
{"Message":"ErrorUnsubscribeTag", "ErrorCode":-2147483621,
"ErrorDescription":"Subscription could not be closed",
"ClientCookie":"mySubscription1"}
```

3.2.3 ReadTag

Description

The "ReadTag" command reads multiple tags. The tag value, the quality, the quality code and the time stamp are signaled.

The order of tags in the response corresponds to the order in the "ReadTag" request.

Request

```
{"Message":"ReadTag","Params":{"Tags":
["<Tag>","<Tag>"]},"ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"ReadTag","Params":{"Tags":
["Tag_0","Tag_1"]},"ClientCookie":"myRequest1"}
```

Response

OnSuccess

```
{"Message":"NotifyReadTag","Params":{"Tags":
[{"Name":"<Tag>","Quality":"<Value>","QualityCode":"<Value>","TimeSt
amp":"<Value>","Value":"<TagValue>","ErrorCode":<Value>,"ErrorDescri
ption":"<ErrorText>"},
{"Name":"<Tag>","Quality":"<Value>","QualityCode":"<Value>","TimeSta
mp":"<Value>","Value":"<TagValue>","ErrorCode":<Value>,"ErrorDescrip
tion":"<ErrorText>"}]},"ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"NotifyReadTag","Params":{"Tags":
[{"Name":"Tag_0","Quality":"Good","QualityCode":"192","TimeStamp":"2
019-01-30T11:25:35Z","Value":"16","ErrorCode":0,"ErrorDescription":"
"},
{"Name":"Tag_1","Quality":"Uncertain","QualityCode":"76","TimeStamp"
:"2019-01-30T11:25:35Z","Value":"1","ErrorCode":-2147483620,"ErrorDe
scription":"Tag does not exist"}]},"ClientCookie":"myRequest1"}
```

OnError

```
{"Message":"ErrorReadTag","ErrorCode":<Value>,"ErrorDescription":"<E
rror text>", "ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"ErrorReadTag","ErrorCode":-2147483621,"ErrorDescription"
:"Failed to Read", "ClientCookie":"myRequest1"}
```

3.2.4 WriteTag

Description

The "WriteTag" command writes the values of multiple tags.

Request

```
{"Message":"WriteTag","Params":{"Tags":
[{"TagName":"<Tag>","Value":"<TagValue>"},
{"TagName":"<Tag>","Value":"<TagValue>"}]},"ClientCookie":"<Cookie>"
}
```

For example:

```
{"Message":"WriteTag","Params":{"Tags":
[{"TagName":"Tag_0","Value":"50"},
{"TagName":"Tag_1","Value":"40"}]},"ClientCookie":"myRequest2"}
```

Response

OnSuccess

```
{"Message":"NotifyWriteTag","Params":{"Tags":
[{"Name":"<Tag>","ErrorCode":<Value>,"ErrorDescription":"<ErrorText>
"},
{"Name":"<Tag>","ErrorCode":<Value>,"ErrorDescription":"<ErrorText>"
}]},"ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"NotifyWriteTag","Params":{"Tags":
[{"Name":"Tag_0","ErrorCode":0,"ErrorDescription":""},
{"Name":"Tag_1","ErrorCode":-2147483620,"ErrorDescription":"Tag
does not exist"}]},"ClientCookie":"myRequest2"}
```

OnError

```
{"Message":"ErrorWriteTag","ErrorCode":<Value>,"ErrorDescription":"<
Error text>", "ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"ErrorWriteTag","ErrorCode":-2147483621,"ErrorDescription
":"Failed to Write", "ClientCookie":"myRequest2"}
```

3.2.5 SubscribeAlarm

Description

The "SubscribeAlarm" command subscribes systems for monitoring of changes of active alarms.

During the first call of "NotifySubscribeAlarm", all active alarms are queried. Thereafter, "NotifySubscribeAlarm" is only called when the status of an alarm changes.

Request

```
{"Message":"SubscribeAlarm","Params":{"SystemNames":
["<System>","<System>"],"Filter":"<Filter>","LanguageId":<ID>},"Clie
ntCookie":"<Cookie>"}
```

- `SystemNames`: Optional
  When the list is empty or missing, all known systems are subscribed for monitoring.
- `Filter`: Optional
- `LanguageID`: Optional
- `ClientCookie`:
  Is used for "UnsubscribeAlarm" and to assign the notification to its monitoring.

For example:

```
{"Message":"SubscribeAlarm","Params":{"SystemNames":
["System0","System1"],"Filter":"AlarmClassName !=
'Warning'","LanguageId":1033},"ClientCookie":"CookieForSubscribeAlar
ms123"}
```

Response

OnSuccess

```
{"Message":"NotifySubscribeAlarm","ClientCookie":"<Cookie>","params"
:{"Alarms":[{<Key value pairs for the properties of the first
alarm>},{<Key value pairs for the properties of the second alarm>},
{<...>}]}}
```

For example:


```
{"Message":"NotifySubscribeAlarm","ClientCookie":"CookieForSubscribe
Alarms123","params":{"Alarms":[{"AcknowledgmentTime":"1970-01-01
00:00:00.0000000","AlarmClassName":"Alarm","AlarmClassSymbol":"Alarm
","AlarmText1":"","AlarmText2":"","AlarmText3":"","AlarmText4":"","A
larmText5":"","AlarmText6":"","AlarmText7":"","AlarmText8":"","Alarm
Text9":"","Area":"","BackColor":"4294967295","ChangeReason":"3","Cle
arTime":"1970-01-01
00:00:00.0000000","Connection":"1.0.0.0.0.0","DeadBand":"No
deadband
configured.","EventText":"","Flashing":"FALSE","HostName":"md1z5cpc"
,"ID":"0","InfoText":"","InstanceID":"9","LoopInAlarm":"","Modificat
ionTime":"2019-01-30
11:25:39.9780320","Name":"RUNTIME_1::Tag_2:Alarm2","NotificationReas
on":"1","Origin":"","Priority":"1","RaiseTime":"2019-01-30
11:25:39.9780320","ResetTime":"1970-01-01
00:00:00.0000000","SourceID":"","SourceType":"1","State":"1","StateM
achine":"7","StateText":"R","SuppressionState":"0","SystemSeverity":
"0","Tag":"RUNTIME_1::Tag_2","TextColor":"4278190080","UserName":"",
"Value":"7","ValueLimit":"No limit
configured.","ValueQuality":"192"},
{"AcknowledgmentTime":"1970-01-01
00:00:00.0000000","AlarmClassName":"Alarm","AlarmClassSymbol":"Alarm
","AlarmText1":"","AlarmText2":"","AlarmText3":"","AlarmText4":"","A
larmText5":"","AlarmText6":"","AlarmText7":"","AlarmText8":"","Alarm
Text9":"","Area":"","BackColor":"4294967295","ChangeReason":"3","Cle
arTime":"1970-01-01
00:00:00.0000000","Connection":"1.0.0.0.0.0","DeadBand":"No
deadband
configured.","EventText":"","Flashing":"FALSE","HostName":"md1z5cpc"
,"ID":"0","InfoText":"","InstanceID":"9","LoopInAlarm":"","Modificat
ionTime":"2019-01-30
11:25:39.9780320","Name":"RUNTIME_1::Tag_2:Alarm1","NotificationReas
on":"1","Origin":"","Priority":"1","RaiseTime":"2019-01-30
11:25:39.9780320","ResetTime":"1970-01-01
00:00:00.0000000","SourceID":"","SourceType":"1","State":"1","StateM
achine":"7","StateText":"R","SuppressionState":"0","SystemSeverity":
"0","Tag":"RUNTIME_1::Tag_2","TextColor":"4278190080","UserName":"",
"Value":"7","ValueLimit":"No limit
configured.","ValueQuality":"192", "AlarmGroupID": "1"}]}
```


OnError


```
{"Message":"ErrorSubscribeTag", "ErrorCode":<Value>,
"ErrorDescription":"<Error text>", "ClientCookie":"<Cookie>"}
```


For example:


```
{"Message":"ErrorSubscribeAlarm","ErrorCode":"-2147483621","ErrorDes
cription":"Alarm Subscription failed because of invalid
filter","ClientCookie":"CookieForSubscribeAlarms123"}
```

3.2.6 UnsubscribeAlarm

Description

The "UnsubscribeAlarm" command unsubscribes the alarms from monitoring that was started with the cookie transferred in the call of "SubscribeAlarm".

Request

```
{"Message":"UnsubscribeAlarm","ClientCookie":"<Cookie>"}
```

For example:
```
{"Message":"UnsubscribeAlarm","ClientCookie":"CookieForSubscribeAlarms123"}
```

Response

OnSuccess

```
{"Message":"NotifyUnsubscribeAlarm","ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"NotifyUnsubscribeAlarm","ClientCookie":"CookieForSubscribeAlarms123"}
```

OnError

```
{"Message":”ErrorUnsubscribeAlarm","ErrorCode":<Value>,"ErrorDescription":"<Error text>", "ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"ErrorUnsubscribeAlarm","ErrorCode":-2147483621,"ErrorDescription":"Subscription could not be closed",
"ClientCookie":"CookieForSubscribeAlarms123"}
```

3.2.7 ReadAlarm

Description

The "ReadAlarm" command reads all active alarms.

Request

```
{"Message":"ReadAlarm","Params":{"SystemNames":
["<System>","<System>"],"Filter":"<Value>","LanguageId":<Value>},"ClientCookie":"<Cookie>"}
```

- `SystemNames`: Optional
  When the list is empty or missing, all known systems are subscribed for monitoring.
- `Filter`: Optional

- `LanguageID`: Optional
- `ClientCookie`:
  Is used for "UnsubscribeAlarm" and to assign the notification to its monitoring.

For example:

```
{"Message":"ReadAlarm","Params":{"SystemNames":
["System0","System1"],"Filter":"","LanguageId":1033},"ClientCookie":
"CookieForReadAlarmRequest456"}
```

Response

```
{"Message":"NotifyReadAlarm", "ClientCookie":"<Cookie>","params":
{"Alarms":[{<Key value pairs for properties of the first alarm>},
{<Key value pairs for the properties of the second alarm>}, {<...>}]}}
```

OnSuccess

For example:

```
{"Message":"NotifyReadAlarm",
"ClientCookie":"CookieForReadAlarmRequest456", "params":{"Alarms":
[{"AcknowledgmentTime":"1970-01-01 00:00:00.0000000",
"AlarmClassName":"Alarm",
"AlarmClassSymbol":"Alarm","AlarmText1":"","AlarmText2":"","AlarmTex
t3":"","AlarmText4":"","AlarmText5":"","AlarmText6":"","AlarmText7":
"","AlarmText8":"","AlarmText9":"","Area":"","BackColor":"4294967295
","ChangeReason":"3","ClearTime":"1970-01-01
00:00:00.0000000","Connection":"1.0.0.0.0.0","DeadBand":"No
deadband
configured.","EventText":"","Flashing":"FALSE","HostName":"md1z5cpc"
,"ID":"0","InfoText":"","InstanceID":"9","LoopInAlarm":"","Modificat
ionTime":"2019-01-30
11:25:39.9780320","Name":"RUNTIME_1::Tag_2:Alarm2","NotificationReas
on":"1","Origin":"","Priority":"1","RaiseTime":"2019-01-30
11:25:39.9780320","ResetTime":"1970-01-01
00:00:00.0000000","SourceID":"","SourceType":"1","State":"1","StateM
achine":"7","StateText":"R","SuppressionState":"0","SystemSeverity":
"0","Tag":"RUNTIME_1::Tag_2","TextColor":"4278190080","UserName":"",
"Value":"7","ValueLimit":"No limit
configured.","ValueQuality":"192"},
{"AcknowledgmentTime":"1970-01-01
00:00:00.0000000","AlarmClassName":"Alarm","AlarmClassSymbol":"Alarm
","AlarmText1":"","AlarmText2":"","AlarmText3":"","AlarmText4":"","A
larmText5":"","AlarmText6":"","AlarmText7":"","AlarmText8":"","Alarm
Text9":"","Area":"","BackColor":"4294967295","ChangeReason":"3","Cle
arTime":"1970-01-01
00:00:00.0000000","Connection":"1.0.0.0.0.0","DeadBand":"No
deadband
configured.","EventText":"","Flashing":"FALSE","HostName":"md1z5cpc"
,"ID":"0","InfoText":"","InstanceID":"9","LoopInAlarm":"","Modificat
ionTime":"2019-01-30
11:25:39.9780320","Name":"RUNTIME_1::Tag_2:Alarm1","NotificationReas
on":"1","Origin":"","Priority":"1","RaiseTime":"2019-01-30
11:25:39.9780320","ResetTime":"1970-01-01
```

```
00:00:00.0000000","SourceID":"","SourceType":"1","State":"1","StateM
achine":"7","StateText":"R","SuppressionState":"0","SystemSeverity":
"0","Tag":"RUNTIME_1::Tag_2","TextColor":"4278190080","UserName":"",
"Value":"7","ValueLimit":"No limit
configured.","ValueQuality":"192"", "AlarmGroupID": "1"}]}
```

OnError

```
{"Message":"ErrorReadAlarm","ErrorCode":<Value>,
"ErrorDescription":"<Error text>", "ClientCookie":"<Cookie>"}
```

For example:

```
{"Message":"ErrorReadAlarm","ErrorCode":-2147483621,
"ErrorDescription":"Alarm Subscription failed because of invalid
filter", "ClientCookie":"CookieForReadAlarmRequest456"}
```

3.3 Reference

The following section contains a reference of the properties of alarms and tags that you get with the commands ReadAlarm and ReadTag.

The commands transfer the property values as string.

Tag properties

"Name" property

Name of the tag

"Value" property

Value of the tag at the moment of the read operation.

"Quality" property

Quality of the read operation of the tag

Possible values:

- "Good"
- "Bad"
- "Uncertain"

"QualityCode" property

Quality code of the read operation of the tag

"TimeStamp" property

Time stamp of the last successful read operation of the tag

"Error" property

Error code of the last read or write operation of the tag

"ErrorDescription" property

Description of the error code of the last read or write operation of the tag

Alarms properties

"InstanceID" property

InstanceID for an alarm with multiple instances

"SourceID" property

Source at which the alarm was triggered.

"Name" property

Name of the alarm

"AlarmClassName" property

Name of the alarm class

"AlarmClassSymbol" property

Symbol of the alarm class

"AlarmParameterValues" property

Parameter values of the alarm

"AlarmText1" … "AlarmText9" properties

Additional texts 1-9 of the alarm

"ChangeReason" property

Trigger event for modification of the alarm state

"Connection" property

Connection by which the alarm was triggered

"State" property

Current alarm state

The property can contain the following values:

- "0": Normal
- "1": Raised
- "2": RaisedCleared
- "5": RaisedAcknowledged
- "6": RaisedAcknowledgedCleared
- "7": RaisedClearedAcknowledged
- "8": Removed

"StateText" property

Current alarm state as text, for example "Incoming" or "Outgoing"

"EventText" property

Text that describes the alarm event.

"InfoText" property

Text that describes an operator instruction for the alarm.

"TextColor" property

Number with the text color of the alarm state

"BackColor" property

Number with the background color of the alarm state

"Flashing" property

Indicates whether the alarm is flashing.

Values: "TRUE" or "FALSE"

"ModificationTime" property

Time of the last modification to the alarm state

"RaiseTime" property

Trigger time of the alarm

"AcknowledgementTime" property

Time of alarm acknowledgment

"ClearTime" property

Time of alarm reset

"ResetTime" property

Time of alarm reset

"SuppressionState" property

Status of alarm visibility

"SystemSeverity" property

Severity of the system error

"Priority" property

Relevance for display and sorting of the alarm

"Origin" property

Origin for display and sorting of the alarm

"Area" property

Origin area for display and sorting of the alarm

"Value" property

Current process value of the alarm

"ValueQuality" property

Quality of the process value of the alarm

"ValueLimit" property

Limit of the process value of the alarm

"UserName" property

User name of the operator input alarm

"HostName" property

Name of the host that triggered the alarm.

"ID" property

ID of the alarm that is also used in the display.

"AlarmGroupID" property

ID of the alarm group to which the alarm belongs.

"SourceType" property

Source from which the alarm was generated, e.g. tag-based, controller-based or system-based alarm.

```
HmiAlarmSourceType SourceType { get; }
```

The enumeration "HmiAlarmSourceType" can contain the following values:

- `Undefined` (0)
- `Tag` (1)
- `Controller` (2)
- `System` (3)
- `Alarm` (4)

"DeadBand" property

Range of the triggering tag in which no alarms are generated.

```
object DeadBand { get; }
```

"LoopInAlarm" property

Function that navigates from the alarm view to its origin.

```
string LoopInAlarm { get; }
```

"NotificationReason" property

Reason for the notification

The property can contain the following values:

- `"0": Unknown`
- `"1": Add`
  The alarm was added to the filtered result list. The alarm meets the filter criteria that apply to the monitoring.
- `"2": Modify`
  Properties of the alarm were changed, but the alarm is still part of the filtered result list.
- `"3": Remove`
  The alarm was part of the result list, but it no longer meets the filter criteria due to changes to its properties.

Note

Changes to the alarm only lead to notifications again when the alarm meets the filter criteria again. In this case, "NotificationReason" is set to `Add`.

Note

Removing alarm from business logic

The use case of the client determines whether the client ignores notifications via alarms with the "NotificationReason" `Modify` or `Remove`.

For example:

- State-based monitoring: The client wants to show a list of incoming alarms. All notification reasons are relevant. The client removes an alarm from the list as soon as the notification reason is `Remove`.
- Event-based monitoring: The client wants to send an email when an alarm comes in. Only the notification reason `Add` is relevant.

Example:

A customer application begins monitoring with the filter criterion "State" = 1. An alarm is triggered. Runtime notifies the customer application of the "NotificationReason" as follows:

NotificationReason	Description
Add	• The "State" property is 1. The alarm has come in.
Modify	• The "State" property has not changed. • Another property that is not part of the filter criterion has changed, e.g. "Priority".
Remove	The "State" property has changed, e.g. alarm has gone out.

3.4 Syntax of the alarm filter

With an AlarmSubscription, a filter can be transferred so that not all active alarms of the alarm system are notified, but only those which match the filter. The filter syntax is based on SQL syntax. However, only the WHERE instruction is relevant. The keyword "WHERE" must be omitted.

Operators

The following operators can be used in the filter string of the alarm filter:

Operator	Description	Example
`=`	equal to	`Name = 'Recipe246'`
`<>`	not equal	`Value <> 0.0`
`>`	greater than	`Value > 25.0`
`<`	less than	`Value < 75.0`
`>=`	greater than or equal to	`Value >= 25.0`
`<=`	less than or equal to	`Value <= 75.0`
`OR, ||`	logical OR	`State = 1 OR State = 3`
`AND, &&`	logical AND	`Value >= 25.0 AND Value <= 75.0`
`BETWEEN`	within a range	`Value BETWEEN 25.0 AND 75.0`
`NOT BETWEEN`	outside a range	`Value NOT BETWEEN 25.0 AND 75.0`
`LIKE string`	corresponds to the string *string*	`Name LIKE 'Motor*'`
`NOT LIKE string`	does not correspond to the string *string*	`Name NOT LIKE 'Valve*'`
`IN (v1, v2, …)`	corresponds to one or more values	`State IN (1, 4, 7)`
`NOT IN (v1, v2, …)`	does not correspond to one or more values	`State NOT IN (0, 2, 3, 5, 6)`
`(…)`	brackets expressions	`Value <= 75.0 AND (State = 1 OR State = 3)`

Precedence of the operators:

Rank	Operators
1	• Relational operators: =, <>, >, <, >=, <= • LIKE • IN • BETWEEN
2	NOT
3	AND, &&
4	OR, ||
5	

Permitted wildcards:

Wildcard	Description	Example
*	Replaces 0 to more characters	`Name LIKE 'Motor*'` `Reference = <1.*.15>1`
?	Replaces 1 character	`Name = 'Recipe?'`