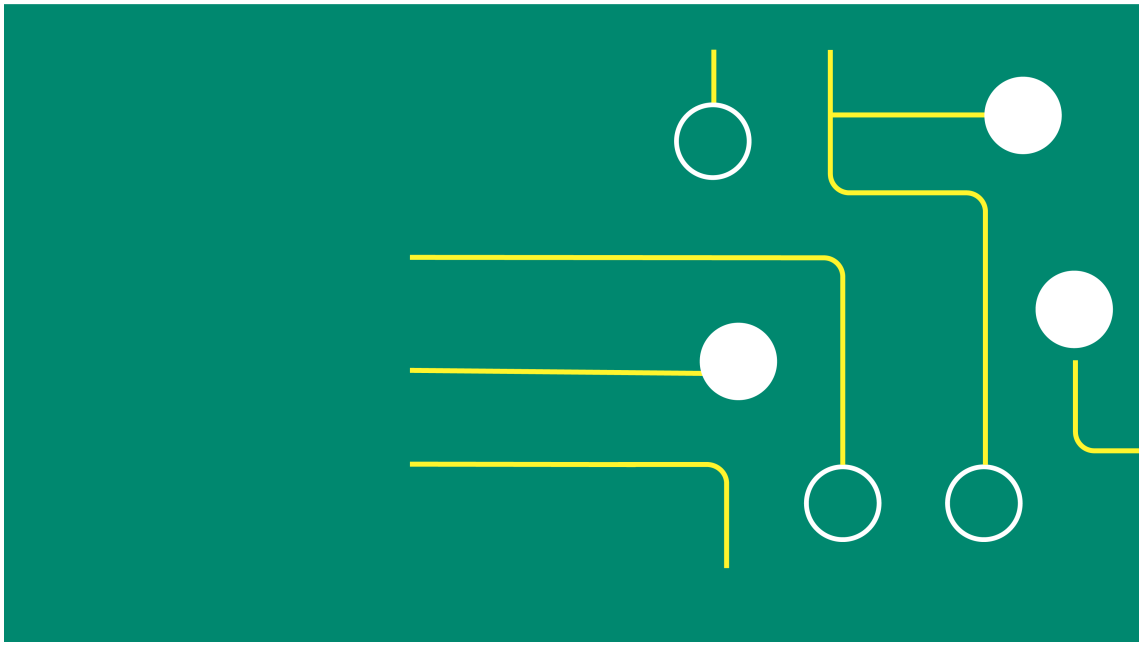
cytiva

UNICORN™ OPC Server 1.3

User Manual



cytiva.com

Table of Contents

1 Introduction 4
1.1 OPC Server 5
1.2 OPC UA standards and versions supported by UNICORN OPC Server 6

2 UNICORN OPC Server general settings 7
2.1 Install UNICORN OPC Server 9
2.2 UNICORN OPC Server license 11
2.3 Installation log files and server configuration files 13
2.4 Configure UNICORN OPC Server 14
2.5 Configure OPC UA Client and connect to UNICORN OPC Server 18
2.6 Trust certificate 25
2.7 Execute method/manual instruction and raise and acknowledge alarm with the OPC UA Client 28
2.7.1 Connect OPC Client with Instrument 29
2.7.2 Execute method 30
2.7.3 Execute manual instruction 32
2.7.4 Raise and acknowledge an alarm 34
2.7.5 Disconnect OPC Client with Instrument 38

3 Deployments 39

4 UNICORN OPC Data Access address space 42
4.1 Run data and Picture data 45
4.2 Trend data 48
4.3 Manual instructions 51
4.4 Method execution 52
4.4.1 Scout Run Index 53
4.4.2 Run Method 55
4.4.3 Result Name 56
4.4.4 Batch ID 58
4.4.5 StartNotes 60
4.4.6 MethodNotes 61
4.4.7 PreCompile 62
4.4.8 Questions 63
4.4.9 Variables 65
4.5 State 67
4.5.1 AssignState 68
4.5.2 RunState 70
4.5.3 Command 72
4.5.4 InstrumentStatus 74
4.5.5 InstrumentStatusConnection 76
4.5.6 UserInControl 78
4.6 Recommendations 79

5 UNICORN OPC Alarms and Events address space 80
5.1 Alarms and errors 82
5.2 Event categories 84

6 UNICORN OPC Historical Data Access address space 85
6.1 AuditTrail 87
6.2 Result file information 88
6.2.1 Curves 89
6.2.2 Documentation 90
6.2.3 Peak tables 92
6.2.4 Systems 93
6.3 UNICORN OPC Historical Data Access XML format definition 94
6.3.1 BufferPro 96
6.3.2 Calibration 97
6.3.3 EvaluationLog 98
6.3.4 Run logbook 99
6.3.5 SignatureList 100
6.3.6 SnapshotList 101
6.3.7 UsedComponents 102
6.3.8 Notes 103
6.3.9 ScoutingList 104
6.3.10 SettingsList 105
6.3.11 TextInstructions 106
6.3.12 VariableList 107
6.3.13 QuestionList 108
6.3.14 PeakTable 109
6.3.15 Unicorn raw 112
6.3.16 Columns 114
6.3.17 EvaluationProcedures 116
6.3.18 FracXY 117
6.3.19 AuditTrail 118
6.3.20 Method/Result Instrument Configuration 119
6.3.21 System 120

1 Introduction

About this chapter

This chapter contains important user information, compatibility information, and a description of the intended use of the UNICORN™ OPC Server.

In this chapter

Section		See page
1.1	OPC Server	5
1.2	OPC UA standards and versions supported by UNICORN OPC Server	6

1.1 OPC Server

The UNICORN OPC Server provides a standardized integration interface to support integration between UNICORN and other software systems, such as Laboratory Information Systems and Manufacturing Execution Systems. Customers can stream the data to a historian. OPC enables open connectivity via open standards that are created in collaboration with leading automation manufacturers worldwide, including Microsoft.

OPC provides inter-operability between system components by creating and maintaining open standard specifications. The benefit of following standard specifications in system implementations is greater independence for hardware and software components. This leads to a higher flexibility, better quality, and an overall lower maintenance costs for the system solution. The first standard developed was the Data Access (DA) specification, which therefore has the broadest client support.

The UNICORN OPC Server supports the following three areas:

- The UNICORN OPC DA that gives access to all process data (e.g., real-time values, valve status, process step information, and commands). UNICORN Alarm and Events (AE) Server informs an OPC UA Client application that a system parameter has exceeded an upper or lower limit value.
- The UNICORN Historical DA allows any OPC UA Client application to access the entire batch results generated by UNICORN.
- The UNICORN OPC security controls client access to the UNICORN OPC DA, Alarms and Events, and Historical DA to protect sensitive information and to guard against unauthorized modification of process parameters. This is an important security feature.

1.2 OPC UA standards and versions supported by UNICORN OPC Server

The UNICORN OPC Server supports the following OPC UA standards:

- Data Access and Alarms and Events Server (DA and AE server)
- Historical Data Access (HDA)
- Security

The UNICORN OPC Server supports `Basic256Sha256` and `Aes128_Sha256_RsaOaep` security policies.

The UNICORN OPC Server is certified by the OPC foundation. The profile *Standard 2017 UA Server Profile* was selected and certified, refer to the profile details in the link below:

https://profiles.opcfoundation.org/profile/1663

This document requires basic knowledge of UNICORN and the OPC UA Client.

2 UNICORN OPC Server general settings

About this chapter

This chapter provides the required system settings for the UNICORN OPC Server.

IMPORTANT
An OPC UA Client does not need to be installed before performing the steps in this chapter. In this document, UaExpert™ is used as an OPC UA Client for performing the steps. The screenshots are also taken using the UaExpert and defined with a particular system. We recommend uninstalling any previous version of the UNICORN OPC Server before installing the latest version of UNICORN OPC Server 1.3.

IMPORTANT
For a new ÄKTA process™ system, the user must be connected and in control of the system before it is possible to configure the UNICORN OPC Server. For all other system types, the user is not required to be connected to the system.

Installation prerequisites

Install Microsoft .Net Windows Desktop Runtime 6.0.27 x86 and Microsoft ASP.NET Core Runtime 6.0.27 x86 before running the UNICORN OPC Server installer. Click the link below to find the .Net installers:

https://dotnet.microsoft.com/en-us/download/dotnet/6.0

UNICORN OPC Server software requirements

Attribute	Specification
Logical processors	Minimum 4 logical processors are recommended
OS version	Windows 10 version 1607 or later (x64) Professional/Enterprise is required
OS language for logged on user	English (U.S.), LCID 1033 is recommended

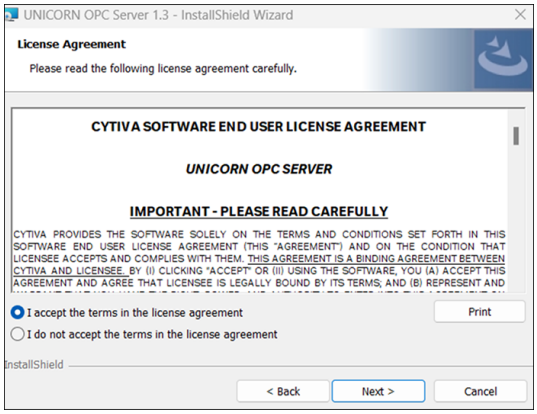
Attribute	Specification
OS language for system user	English (U.S.), LCID 1033 is recommended
File system format	NTFS is required
Minimum free disk space	Minimum 60 GB
Minimum available RAM	Minimum 8 GB
Page file settings	Minimum 4 GB

In this chapter

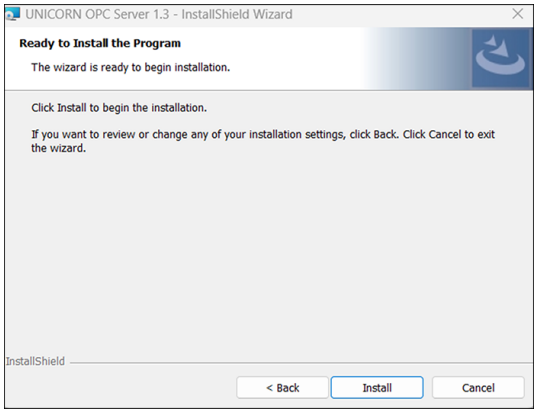
Section		See page
2.1	Install UNICORN OPC Server	9
2.2	UNICORN OPC Server license	11
2.3	Installation log files and server configuration files	13
2.4	Configure UNICORN OPC Server	14
2.5	Configure OPC UA Client and connect to UNICORN OPC Server	18
2.6	Trust certificate	25
2.7	Execute method/manual instruction and raise and acknowledge alarm with the OPC UA Client	28

2.1 Install UNICORN OPC Server

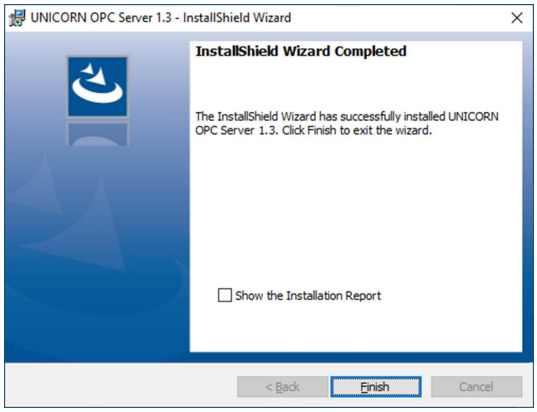
Follow the instructions below to install the UNICORN OPC Server.

Step	Action
1	Run the ***UNICORN OPC Server Setup 1.3.exe*** installation file. ***Note:*** *If a window appears for installing Microsoft .Net Windows Desktop Runtime 6.0.27 x86 or Microsoft ASP.NET Core Runtime 6.0.27 x86, see* *Installation prerequisites, on page 7**.*
2	Click ***Next*** in the ***UNICORN OPC Server 1.3. - InstallShield Wizard***. 
3	Agree to the terms in the license agreement, then click ***Next***. 

Step	Action
4	Click ***Install*** to begin the installation.



5	Click ***Finish*** to close the ***InstallShield Wizard***.



2.2 UNICORN OPC Server license

Introduction

It is assumed that the UNICORN OPC Server is installed, and that the UNICORN OPC service is running.

UNICORN OPC Server license (Concurrent) should be generated, and configured using the ***Configure e-License*** tool.

The license server for UNICORN OPC Server needs to be provided.

IMPORTANT

For the Concurrent license, provide the concurrent license server Hostname or IP address.

The ***UNICORN OPC Server.dll.config*** file is located at `C:\Program Files (x86)\Cytiva\UNICORN OPC Server 1.3\Bin`.

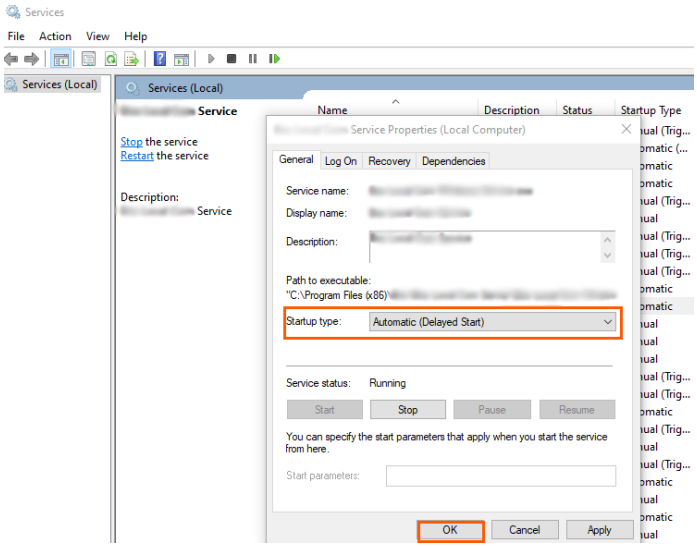
Change service startup type to Automatic (Delay Start)

We recommend changing the ***Startup Type*** of the ***UNICORN OPC Server*** service to ***Automatic (Delay Start)*** in the ***Services*** app in Windows.

Follow the instructions below to perform the change.

Step	Action
1	Open the ***Services*** app in Windows.
2	Right-click the ***UNICORN OPC Server*** service and click ***Properties.***
3	Change the ***Startup type*** to ***Automatic (Delayed Start)***.

Step	Action
4	Click ***OK***.



2.3 Installation log files and server configuration files

The following table describes the locations of the installation log files and the server configuration files:

Files	Location
Installation log files	`C:\ProgramData\Cytiva\UNICORN OPC Server\Logs`
UAHDAServer.Config.xml (for HDA Server) UADAAndAEServer.Config.xml (For DA and AE Server) UNICORN OPC Server.dll	`C:\Program Files (x86)\Cytiva\UNICORN OPC Server 1.3\Bin`

In order to enable more logs, change the value of the tags below in the config file (`UNICORN OPC Server.dll`).

- ***<GatewayLogger DefaultLogLevel="">***
- ***<logger name="GatewayRunLogger">***
- ***<logger name="GatewayCoreLogger">***
- ***<logger name="WatchdogLogger">***
- ***<logger name="DAAndAEAdapterLogger">***
- ***<logger name="HDAAdapterLogger">***

In the example below, more logs can be enabled by changing the level value ***= Trace***, ***Debug***, ***Information***, ***Warning***, ***Error***, or ***Critical***.

<logger name="GatewayRunLogger">
<level value="ERROR" />
<appender-ref ref="GatewayRunLog" />
</logger>

After changing the level value of any of the tags above, save the config file, and restart UNICORN OPC Server to implement the changes.

Note: *When the level value is changed to* ***= Trace***, ***Debug***, ***Information***, ***Warning***, ***Error***, *or* ***Critical***, *the size of the log file increases accordingly.*

2.4 Configure UNICORN OPC Server

Introduction

The ***UNICORN OPC Server Configuration Tool*** is used to update the following:

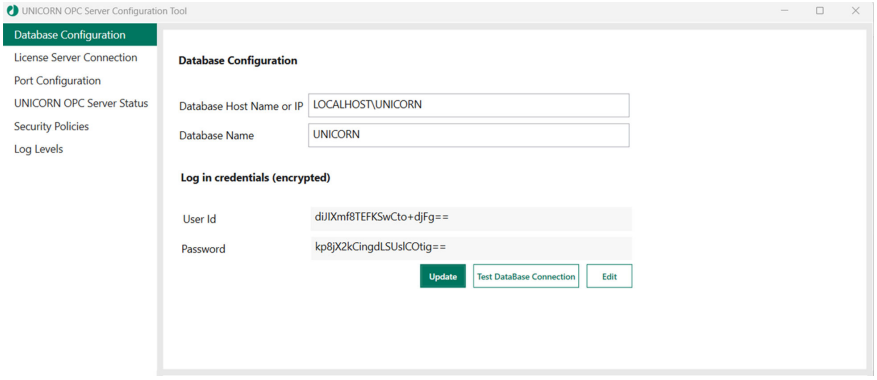
- Database
- License server host name
- Ports
- UNICORN OPC Server status
- Security policies
- Log Levels

Type `UNICORN OPC Server Configuration tool` in ***Windows Search*** and select ***Run as Administrator***.

Use the ***UNICORN OPC Server Configuration Tool*** window to configure the Database, the License Server host name, the Ports, and also to enable or disable some security policies as detailed in the following sections.

Database Configuration

Enter the UNICORN ***Database Host Name*** or ***IP*** address and click ***Update***. Then the host name or the IP address is added to the configuration file.

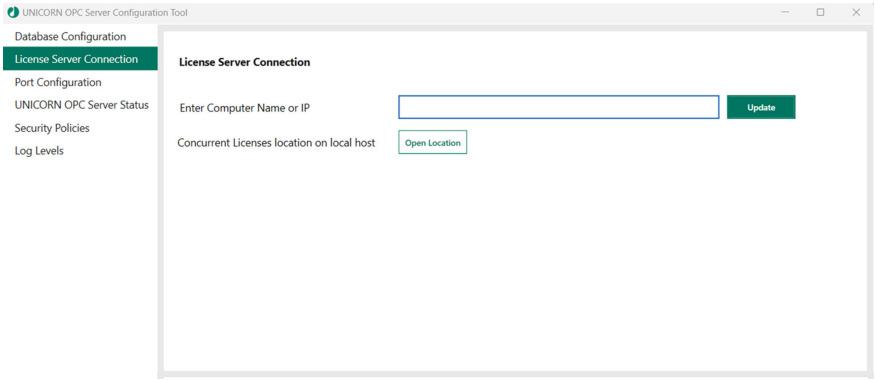


License Server Connection

Enter the OPC License Server ***Computer Name*** or ***IP*** address and click ***Update***. Then the computer name or the IP address is added to the configuration file.

Note: *The License Server for UNICORN OPC Server needs to be provided.*

For a Concurrent license, the license host value is the concurrent License Server IP address or ***Computer name****.*

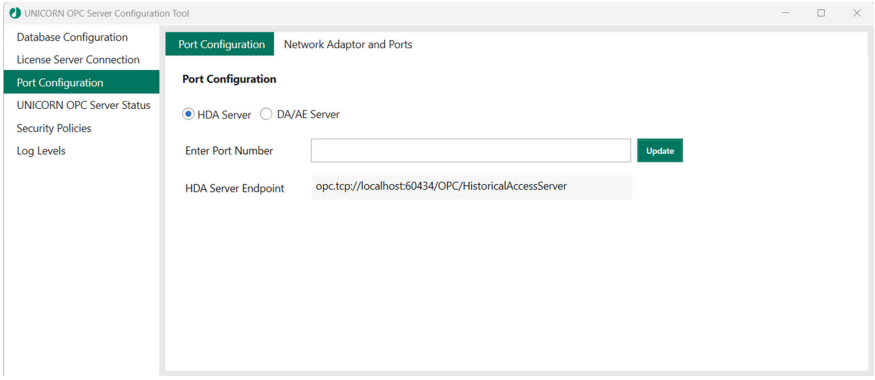


Port Configuration

Port Configuration

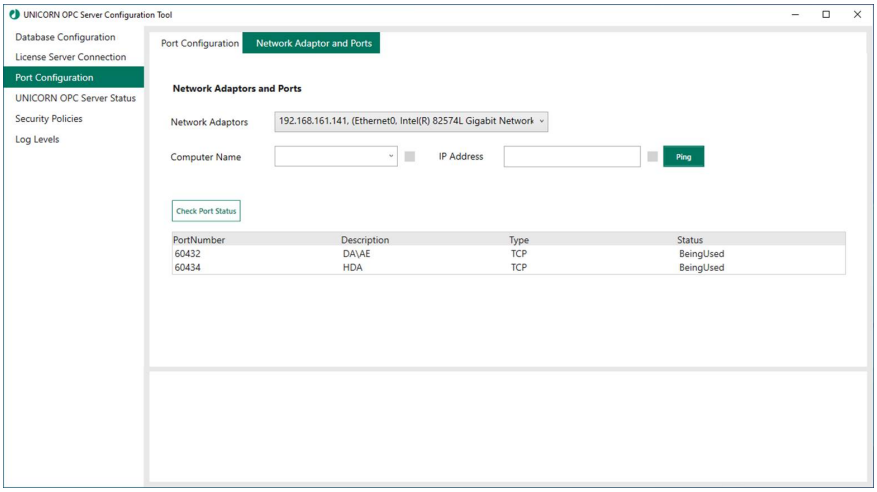
The default Port for the ***DA/AE Server*** is 60432, and the default Port for the ***HDA Server*** is 60434.

To change the Port, select ***HDA Server*** or ***DA/AE Server***, type the Port number in the ***Enter Port Number*** text field, and click ***Update***. Then the Port number is added to the configuration file.



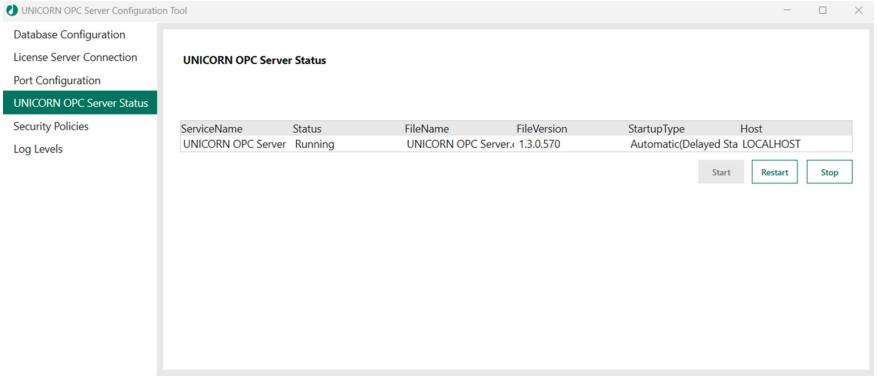
Network Adaptor and Ports

The user can check the ***Status/Type*** of the ***PortNumber*** of DA/AE/HDA server, which are assigned in the ***Port Configuration*** tab on the selected computer name.



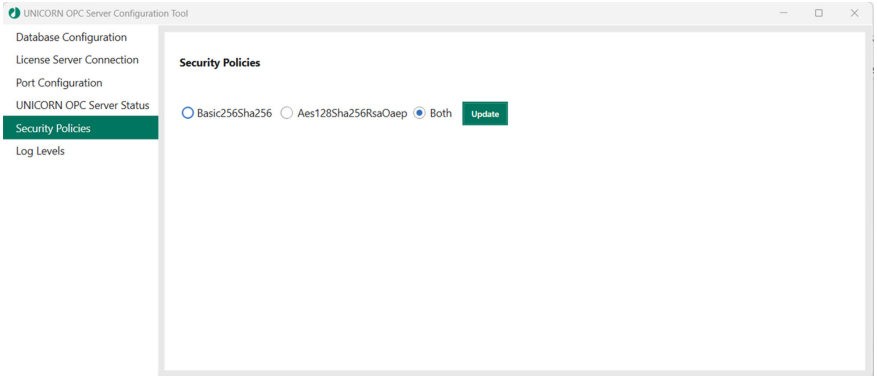
UNICORN OPC Server Status

Use this setting to ***Start***, ***Restart***, or ***Stop*** the ***UNICORN OPC Server***.



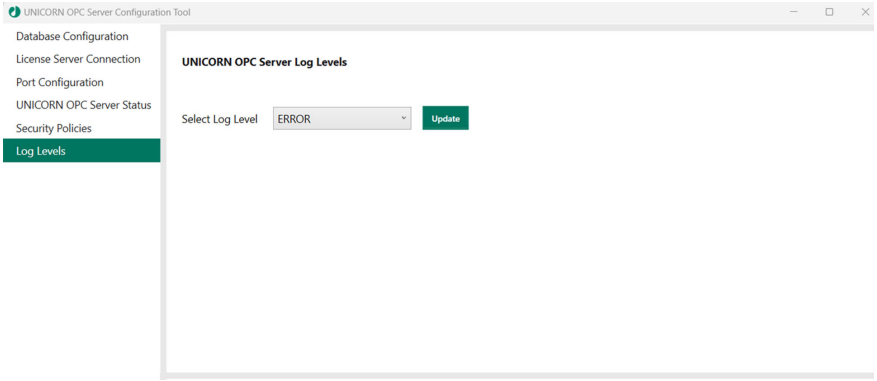
Security Policies

Select any one of the ***Security Policies*** or ***Both*** and click ***Update***.



Log Levels

Click the drop-down box and ***Select Log Level*** (ERROR, INFO, WARNING, DEBUG, ALL) to get detailed UNICORN OPC Server logs, then click ***Update***.

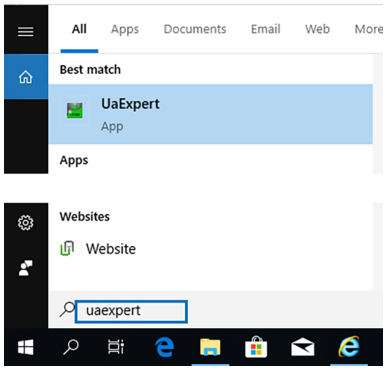


2.5 Configure OPC UA Client and connect to UNICORN OPC Server

Note: *In this section, the UaExpert is used as the OPC UA Client for performing the different steps. The screenshots show the UaExpert interface.*

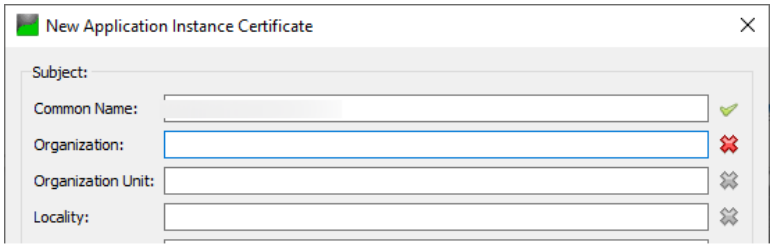
Follow the instructions below to configure and connect the OPC UA Client to the UNICORN OPC Server.

Step	Action
1	Enter `uaexpert` in the Windows search box and press **Enter**.

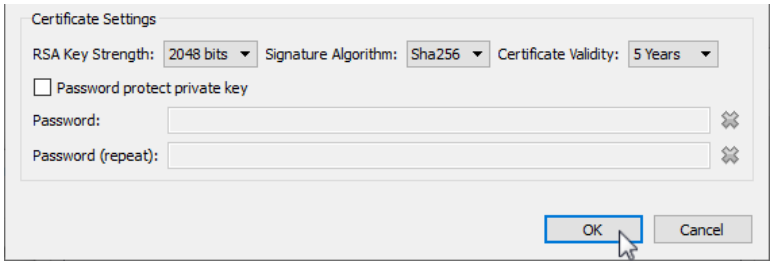


2	If the following dialog box appears, click ***OK***.

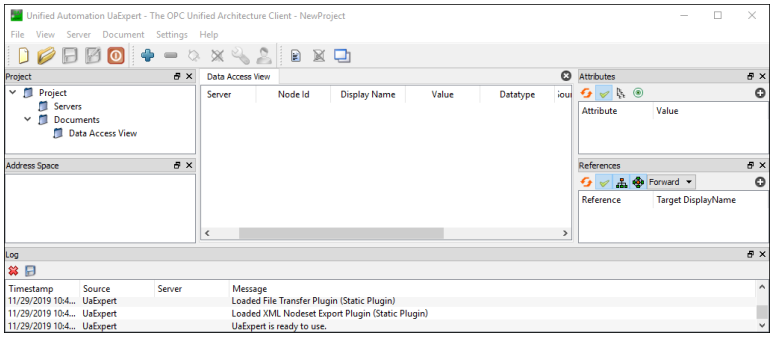
3	Enter the details for the below fields in the ***New Application Instance Certificate*** window.



Step	Action
4	Click ***OK***.



Result:
The following window appears:



Tip:

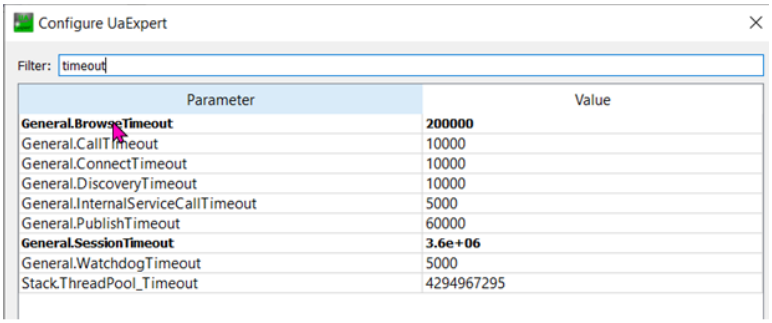
*To avoid the **BadTimeout** error during the connection of the UNICORN OPC Server, or during the browsing and execution of the method call, the user is suggested to increase the **Timeout** value in the OPC UA Client.*

*The user needs to increase the value of the **General.WatchdogTimeout** if the parameter was subscribed to for monitoring, but the client does not receive the monitoring response (i.e., grayed out for a moment) within the specified **WatchdogTimeout** value.*

*For reference, the **Timeout** value can be set in the **OPC UAExpert Client**, as below:*

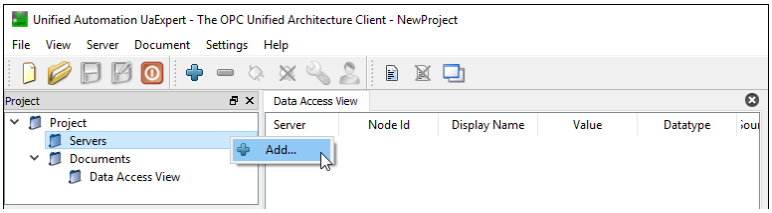
*Go to **Setting** →**Configure UaExpert**, then set a larger value for the **General.Browse.Timeout** parameter, as seen in the below screenshot:*

Step	Action

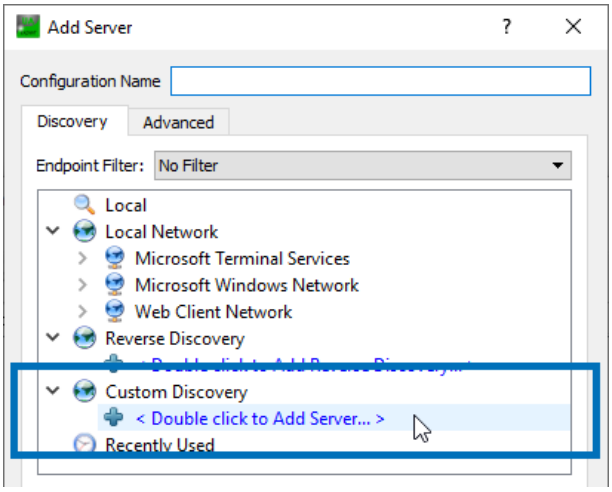


Note:
*The **Timeout** value can vary for different OPC UA Clients, and it also depends on the infrastructure.*

5 Right-click on ***Servers*** and then click ***Add***.

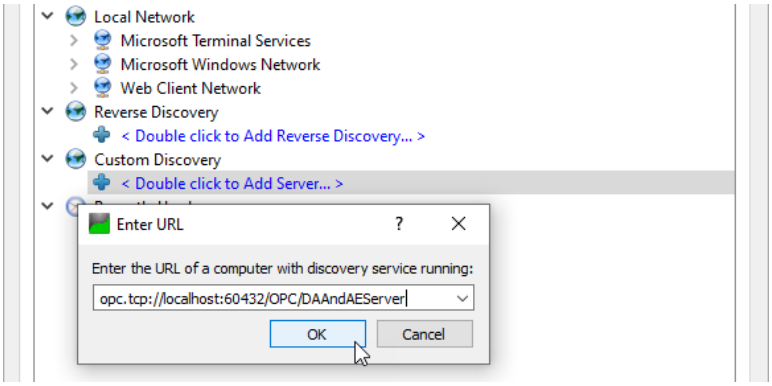


6 Double-click on the item as illustrated below:



Step	Action
7	Enter the IP address/URL of the DA and AE Server/HDA Server and click ***OK***. Below is an example URL of a DA and AE Server.

Note:
The UNICORN OPC Server URL/Address is available in the UNICORN OPC Server configuration tool.



- Definition of the DA and AE server URL:

  opc.tcp://<IP Address/Hostname of the UNICORN OPC Server>:<Port Number of the DAAndAE Server>/OPC/<DAAndAEServer>

- Definition of the HDA server URL:

  opc.tcp://<IP Address/Hostname of the UNICORN OPC Server>:<Port Number of the HDA Server>/OPC/<HistoricalAccessServer>

Example of URL where the OPC Server and the client are on the same machine and default ports are used:

opc.tcp://localhost:60432/OPC/DAAndAEServer

When connecting from a remote client, provide the IP address of the system where the OPC Server is running.

Use an updated port in the URL string if the port has been updated using the OPC Configuration Tool.

Note:

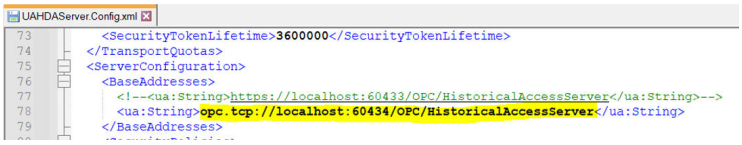
The URL string to add a DA/AE Server or an HDA Server to the OPC UA Clients can be obtained from the configuration files located in `C:\Program Files (x86)\Cytiva\UNICORN OPC Server 1.3\Bin`.

File names:

- *DA/AE Server:* ***UADAAndAEServer.Config***

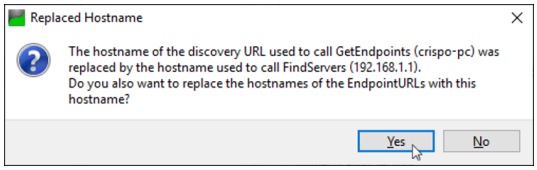
Step	Action

- *HDA Server:* ***UAHDAServer.Config***

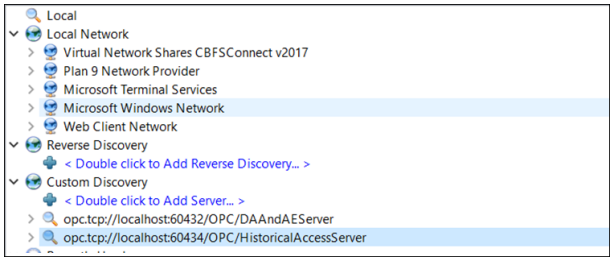


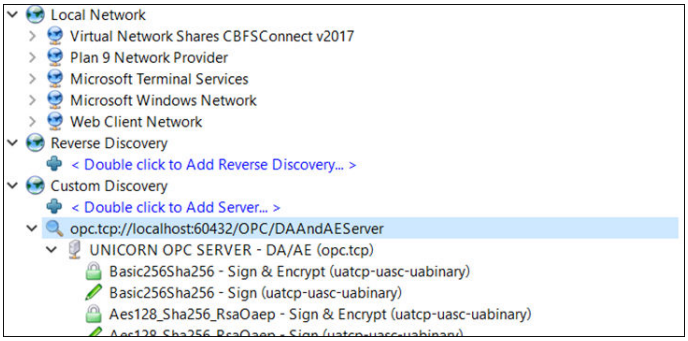
We do not recommend updating the configuration files manually.

8 Click ***Yes***.

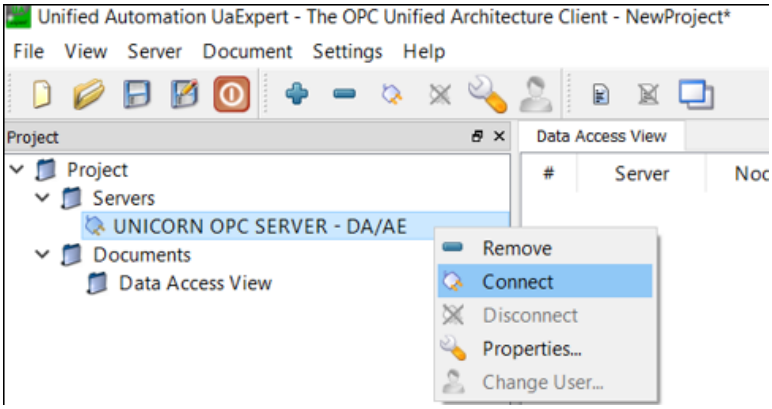


Result:
The IP address/URL is now listed under ***Custom Discovery***.



Step	Action
9	**Connect UA Expert with UNICORN OPC Server** Expand the IP/URL and double-click the highlighted (example) server as illustrated below: 
10	Enter UNICORN ***Username*** and ***Password***. The users' logons from the OPC UA Client should be part of only one access group. ***Note:*** *We do not recommend disabling the password in UNICORN while using UNICORN OPC Server.* *If the user disables the password in UNICORN, the user should provide a password identical to their username when logging on to the OPC UA client.* ***Note:*** *We do not recommend changing the logged on users from OPC UA client without closing the session completely.* *Example: If the default user has logged on to the OPC UA client, do not switch to another user until the session is completely disconnected*

Step	Action
11	Go back to the ***Unified Automation UaExpert*** window and right-click on ***OPC DA and AE server*** and the click ***Connect***.



Note:
The connected OPC UA user can be identified by ***OPC_Hostname*** *displayed in system control, in UNICORN, and system logs.*

2.6 Trust certificate

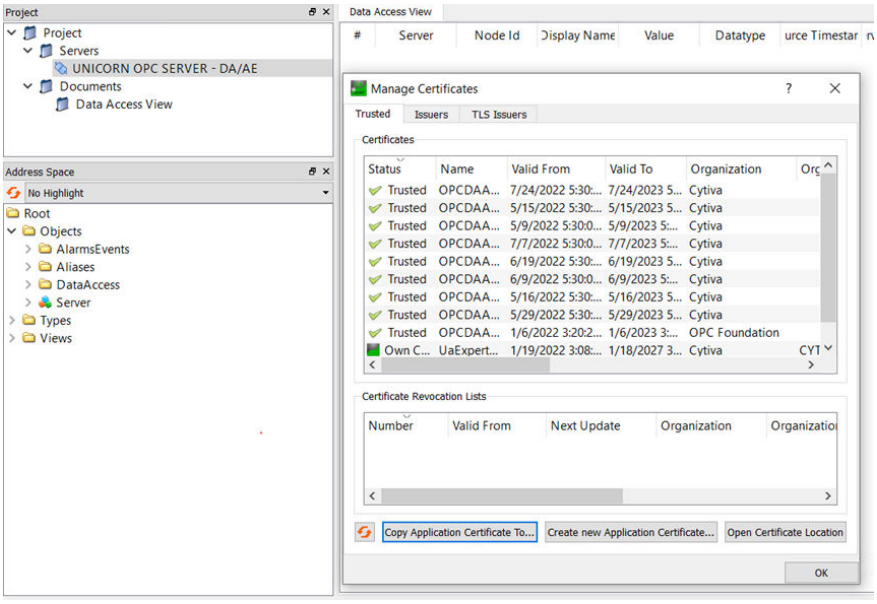
An initial attempt to securely connect to an UNICORN OPC Server from the Workflow server can fail due to a missing client certificate with the following error message:

`Bad Security Check Failed.`

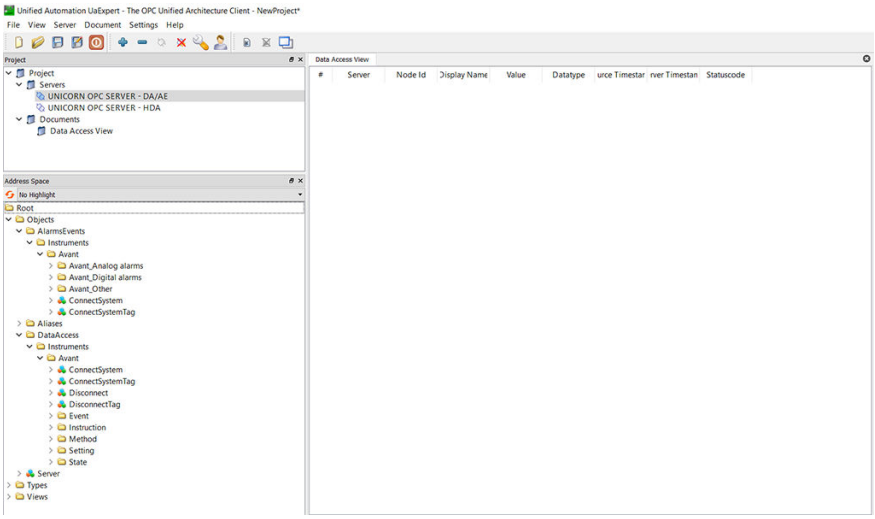
Once a certificate is created, you can establish a connection to the UNICORN OPC Server. The process is described below:

1. The certificate is sent to the server as a part of the connection request.
2. The UNICORN OPC Server returns the server certificate during the connection exchange.
3. The UNICORN OPC Server does not trust the client certificate initially and places the certificate in `C:\ProgramData\Cytiva\UNICORN OPC Server\pki\rejected\certs`.
4. Cut the certificate from the above location and paste it to the location below: `C:\ProgramData\Cytiva\UNICORN OPC Server\pki\trusted\certs`.

To check the details of the certificates, go to ***Settings*** →***Manage Certificates***.



The address space appears after connection to the server.



UNICORN OPC Server creates a self-signed certificate by default.

Note:

- *When UNICORN OPC Server certificate has expired during an ongoing run then there will not be a communication break down from the current session.*

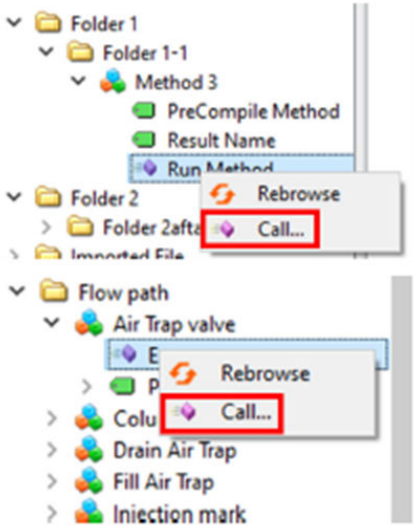
  An ongoing run will not be impacted if UNICORN OPC Server certificate has.
- *Certificate renew should be done when no connection to UNICORN OPC Server exists.*
- *When a user creates a new connection we recommend that the user manually deletes both OPCDAAndAEServer and HDAServer certificates from the path below and restart the UNICORN OPC Server so that OPC stack validates and creates a new certificate:*

  `C:\ProgramData\Cytiva\UNICORN OPC Server\pki\own\certs`

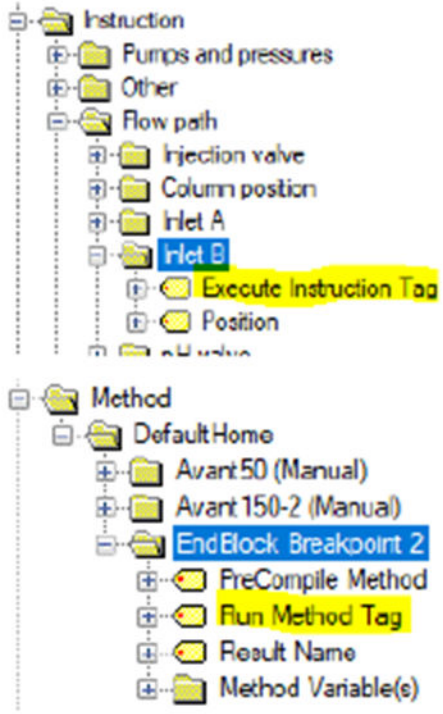
  `C:\ProgramData\Cytiva\UNICORN OPC Server\pki\trusted\certs`

Production Client Settings

For the OPC UA Client that supports ***Method Call / Method services***, the UNICORN method is executed by clicking ***Call...***, as shown in the screenshots below.



For the OPC UA Client that does not support ***Method Call / Method services***, the UNICORN method cannot be executed by clicking ***Call...***. Instead, ***Execute Instruction Tag***, ***Execute Setting Tag***, or ***Run Method Tag*** can be clicked to execute the UNICORN method, as shown in the example screenshots below.



2.7 Execute method/manual instruction and raise and acknowledge alarm with the OPC UA Client

Introduction

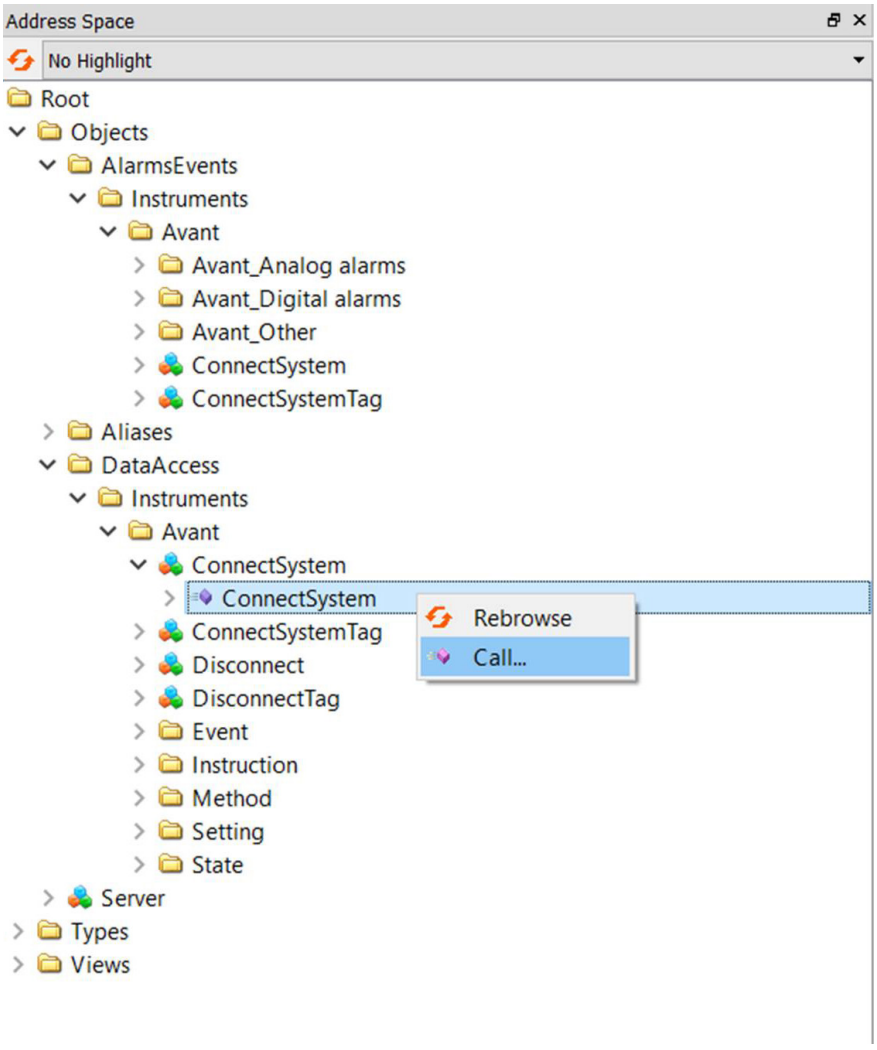
This section provides information regarding how to use the OPC UA Client to execute a method or a manual instruction, and how to raise and acknowledge an alarm.

In this section

Section		See page
2.7.1	Connect OPC Client with Instrument	29
2.7.2	Execute method	30
2.7.3	Execute manual instruction	32
2.7.4	Raise and acknowledge an alarm	34
2.7.5	Disconnect OPC Client with Instrument	38

2.7.1 Connect OPC Client with Instrument

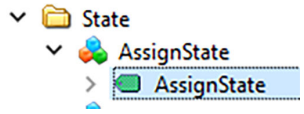
Call the ***ConnectSystem*** method to connect the OPC Client to the Instrument.



2.7.2 Execute method

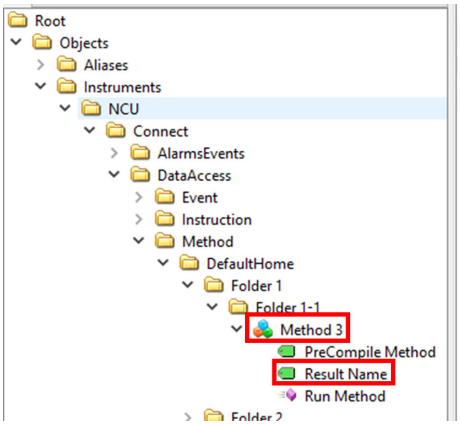
Follow the instructions below to execute a method using the OPC UA Client.

Step	Action
1	Subscribe to the ***AssignState*** tag, and set the value to 1 to connect the OPC in control mode.

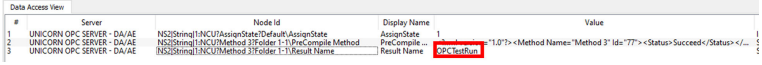


Note:
A value of 0 means the OPC is connected in view mode.

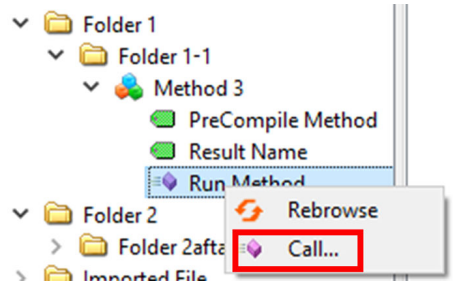
2	Navigate to the ***Method***, then drag and drop ***Method 3***, or the ***Result Name*** tag.

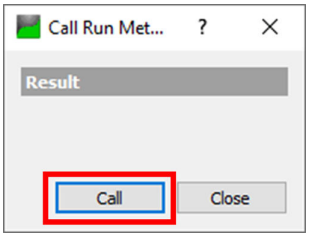
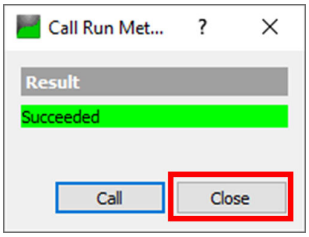


3	Enter the ***Result Name*** for the file, for example, ***OPCTestRun***.



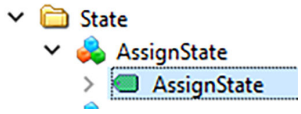
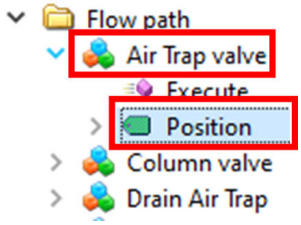
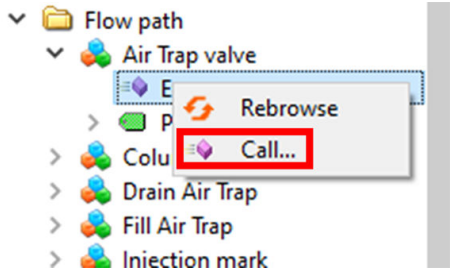
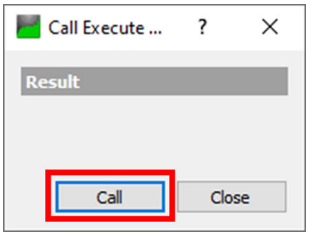
4	Right-click ***Run Method***, then click ***Call...***.

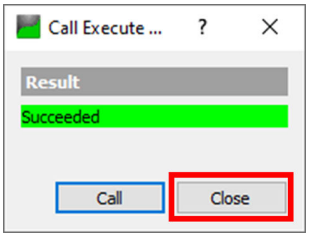


Step	Action
	Result: Click ***Call...*** in the ***Call Run Met...*** window that appears. 
5	If the call is successful when ***Succeeded*** appears, then click ***Close***. 

2.7.3 Execute manual instruction

Follow the instructions below to execute a manual instruction using the OPC UA Client.

Step	Action
1	Subscribe to the ***AssignState*** tag, and set the value to `1` to connect the OPC in control mode.  ***Note:*** *A value of 0 means the OPC is connected in view mode.*
2	Subscribe to a tag, for example, ***AirTrap Valve***, then drag and drop the ***Position*** tag. Set the value to `0` for bypass or `1` for inline. 
3	Right-click ***Execute***, then click ***Call...***.  *Result:* Click ***Call...*** in the ***Call Execute ...*** window that appears. 

Step	Action
4	If the call is successful when ***Succeeded*** appears, then click ***Close***. 

2.7.4 Raise and acknowledge an alarm

Introduction

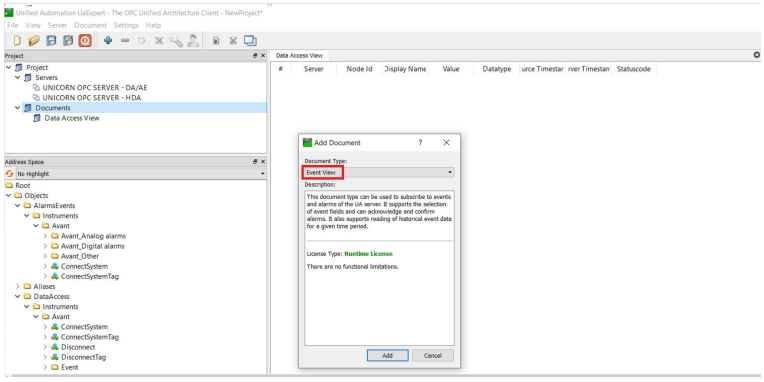
Alarms, events, and warnings only get notified when the user drags the ***Server*** tag, and expands/connects a node for a system under the ***AlarmsEvents*** folder.

This section contains information regarding raising and acknowledging an alarm using the OPC UA Client. Examples will be shown regarding how to generate an analog alarm for the ***Alarm conductivity*** tag, and how to generate a digital alarm for the ***BufferPro pH*** tag.

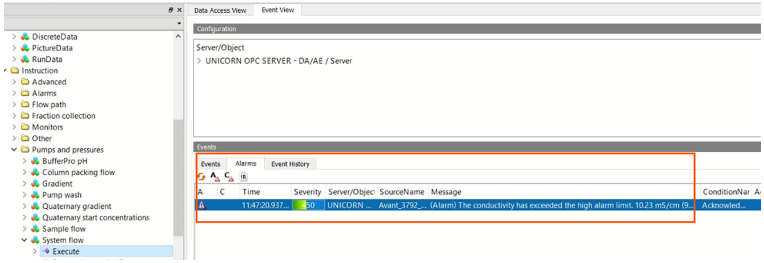
Raise and acknowledge an alarm

Follow the instructions below to raise and acknowledge an alarm using the OPC UA Client.

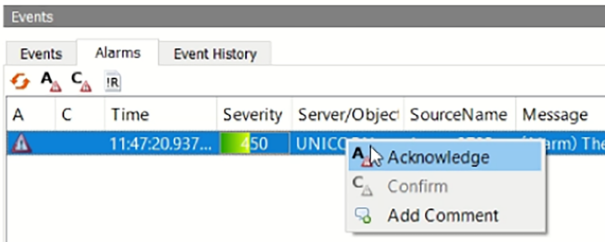
Step	Action
1	To access the Alarms and Events data, click on ***Documents***, then select ***Events View*** under ***Document Type***, and then click ***Add***.



2	Drag ***Server/Object*** to the ***Configuration*** panel, then expand/connect a node for a system under the ***AlarmsEvents*** folder and check for the alarms in the ***Alarms*** tab.



Step	Action
3	Select and right-click the alarm, then click on ***Acknowledge*** to acknowledge the alarm. The acknowledged alarm will now be logged under the ***Events*** tab.

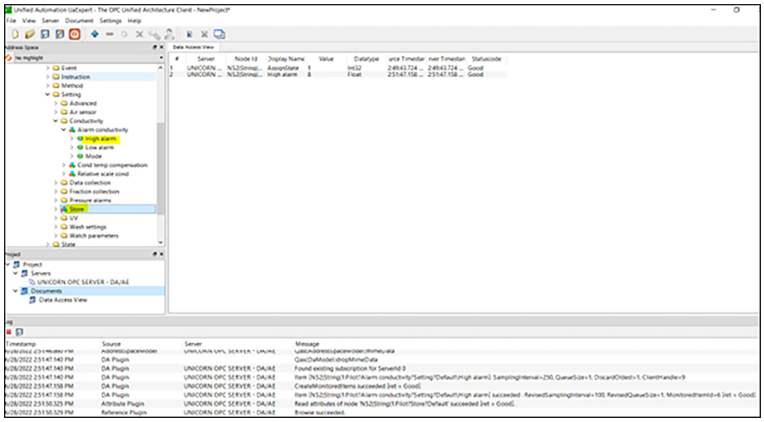


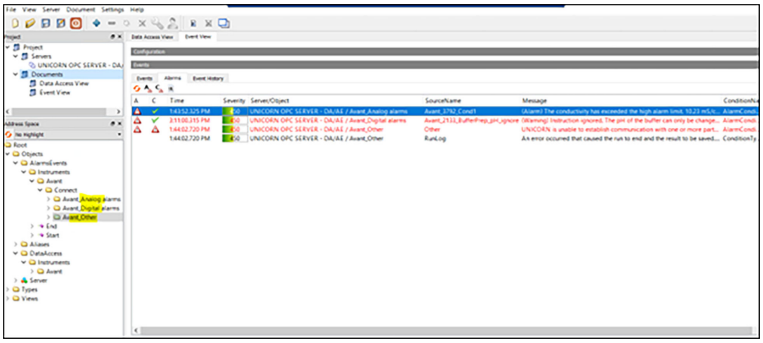
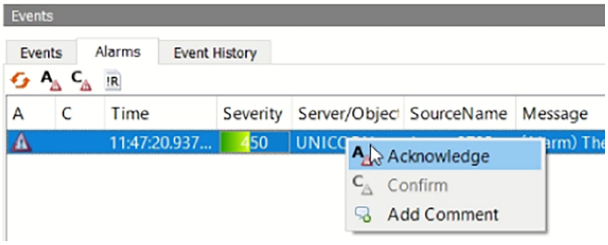
Note:
The alarms disappear from the queue once they are acknowledged.

Raise an analog alarm for *Alarm conductivity* tag

Follow the instructions below to raise an analog alarm for the ***Alarm conductivity*** tag using the OPC UA Client.

Step	Action
1	In the `/State/AssignState` file path under the ***DA*** server, add the ***AssignState*** tag. In the `/Setting/Alarm conductivity` file path, add the ***High alarm*** and ***Execute*** tags.

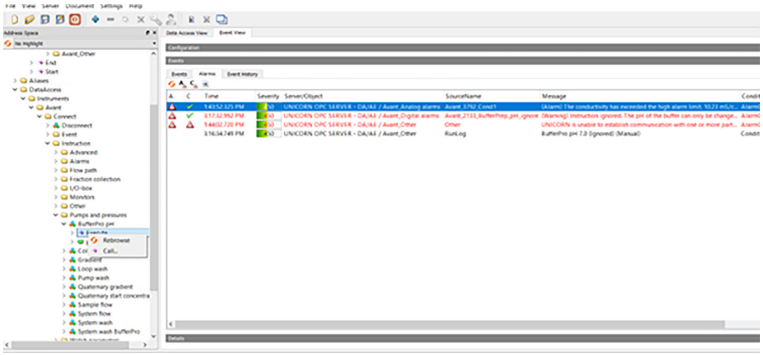
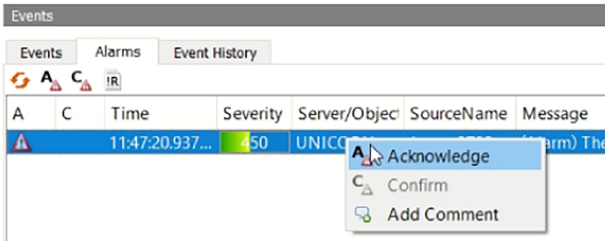


Step	Action
2	Click the ***Alarms*** tab to see the generated analog alarm. 
3	Select and right-click the alarm, then click on ***Acknowledge*** to acknowledge the alarm. The acknowledged alarm will now be logged under the ***Events*** tab.  ***Note:*** *The alarms disappear from the queue once they are acknowledged.*

Raise a digital alarm for BufferPro pH tag

Follow the instructions below to raise an digital alarm for the ***BufferPro pH*** tag using the OPC UA Client.

Step	Action
1	In the `/State/AssignState` file path under the ***DA*** server, add the ***AssignState*** tag. In the `/Instruction/Pumps and pressures` file path, add the ***BufferPro pH*** and ***Execute*** tags.

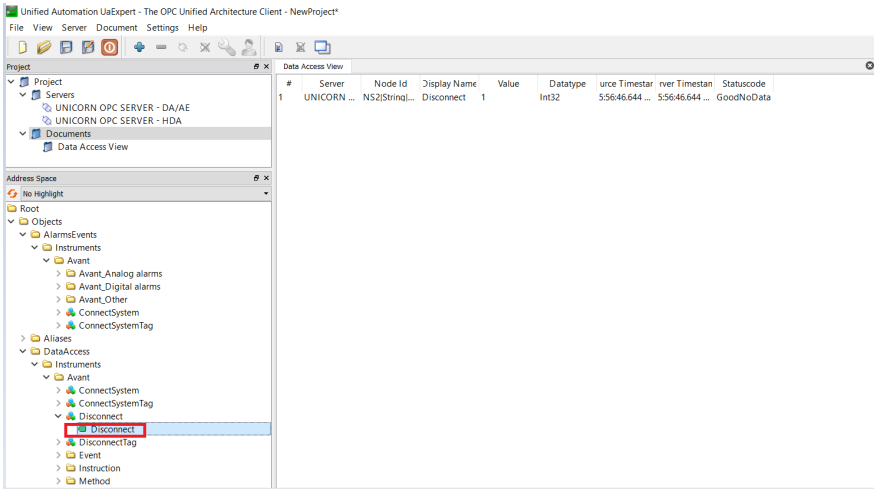
Step	Action
2	Click the ***Alarms*** tab to see the generated digital alarm. 
3	Select and right-click the alarm, then click on ***Acknowledge*** to acknowledge the alarm. The acknowledged alarm will now be logged under the ***Events*** tab. 

Note:
The alarms disappear from the queue once they are acknowledged.

2.7.5 Disconnect OPC Client with Instrument

Subscribe to the ***Disconnect*** tag, and set the value to 1 to disconnect the OPC from the Instrument.

The ***Disconnect*** method call will also disconnect the OPC from the instrument.



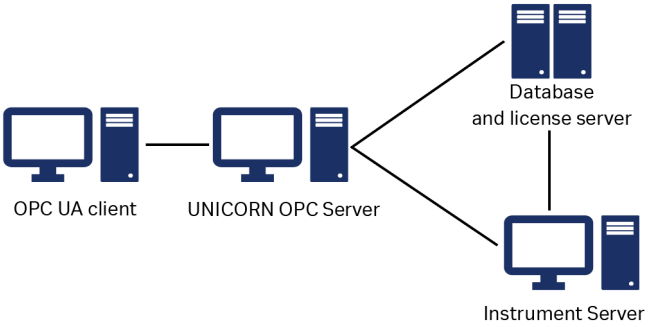
3 Deployments

Introduction

This chapter describes the various deployments by which the UNICORN OPC Server can be connected to the OPC UA Client.

Distributed environment (single system)

The following image shows a single system deployment.



Distributed environment (multisystem)

The UNICORN OPC Server has passed the 72 hour run, required by the OPC foundation. We recommend using a maximum of 5 users or a maximum of 5 systems if used in a distributed/multisystem environment with a monitoring load of 5K tags.

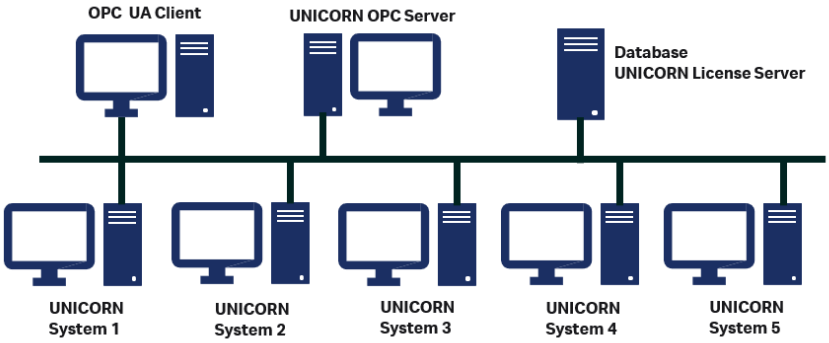
IMPORTANT
We do not recommend installing the UNICORN OPC Server on ReadyToProcess™ 25.

Note: *The user can connect more than five systems with one UNICORN OPC Server, but performance or connectivity issues could be encountered.*

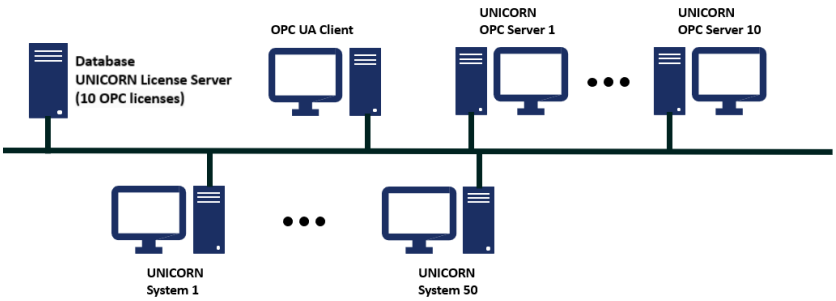
Note: *Do not connect multiple OPC UA Clients (from the same or different computer) to the same system. If multiple users or the same user needs to connect to same system, then log off the current OPC UA user and log on with the user that needs to view the data. Disconnecting one user will disconnect all the other users who are connected to the same system.*

Note: *If the same user wants to connect to different instruments from two different computers using the UNICORN OPC Server, we recommend the following scenarios:*

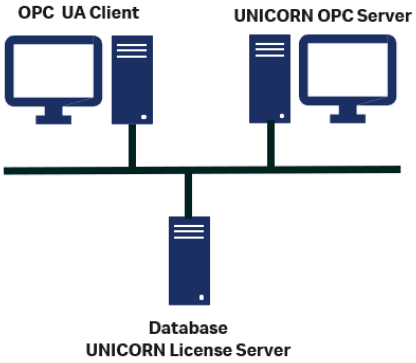
- *The same and unique user id should be used when connecting to a specific instrument, so one dedicated user should be used for each instrument.*
- *The same and unique user id should be used when connecting from a specific client.*
- *Alternatively, make sure all the connections are performed from the same client.*
- *Use one UNICORN OPC Server for each instrument, but this can increase the cost of a license.*



The below image shows the recommended setup for the distributed environment with more than five systems.



The below image shows the Historical Data Access (HDA) Server connection. In this HDA Server connection, the user can access the UNICORN OPC HDA address space, audit trail, HDA XML format definition, and so on, without connecting to the instrument.



Note: *The UNICORN Database and license server can be installed on a different machine.*

4 UNICORN OPC Data Access address space

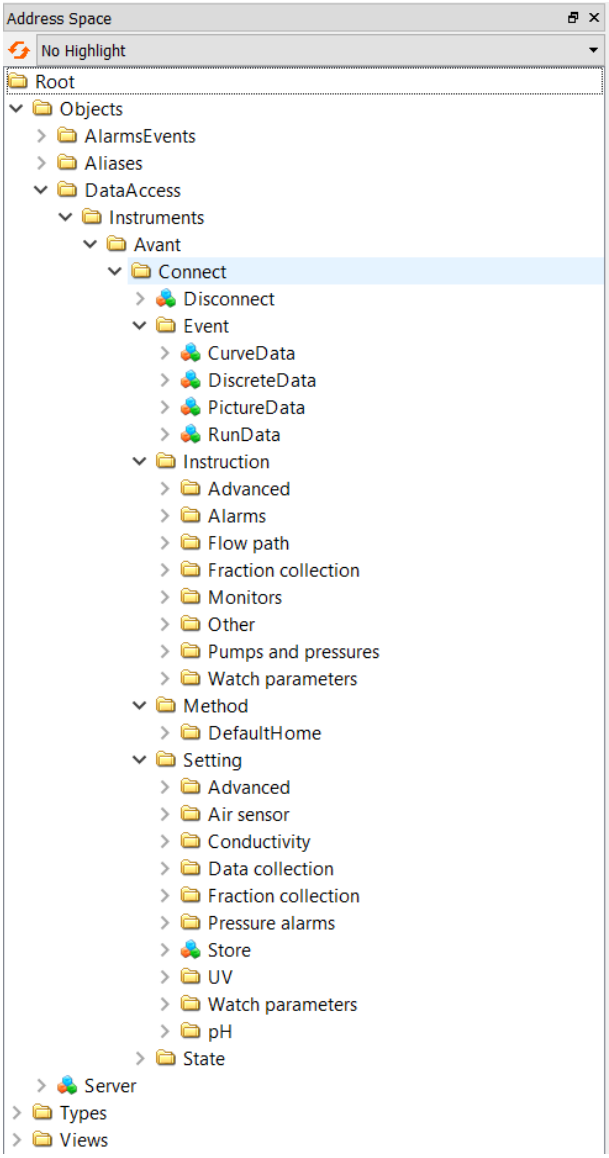
About this chapter

This chapter provides information about the UNICORN OPC Historical Data Access (HDA) address space, method execution, settings, run data, picture data, and so on.

The OPC DA address space is a real-time representation of the process control data for the connected instruments.

Data Access address space view

The following screenshot shows the DA address space view.



Tip: *The address space and the tags depend on the instrument configurations, connected systems, and the user logged on. The OPC server tags contain a username that is included in the Node Id (tag path). The tags generated under the* ***Event***, ***Instruction***, ***Setting***, *and* ***State*** *folders include the user that is logged on. For example, the default user is logged on into the OPC UA*

Client in all the screenshots in the sections of this chapter. See the screenshots in the sections of this chapter to see how the tags are built. The tags are configured by a user during the initialization of the production client. The same user needs to log into the production client to access the configured tags.

In this chapter

Section		See page
4.1	Run data and Picture data	45
4.2	Trend data	48
4.3	Manual instructions	51
4.4	Method execution	52
4.5	State	67
4.6	Recommendations	79

4.1 Run data and Picture data

Introduction

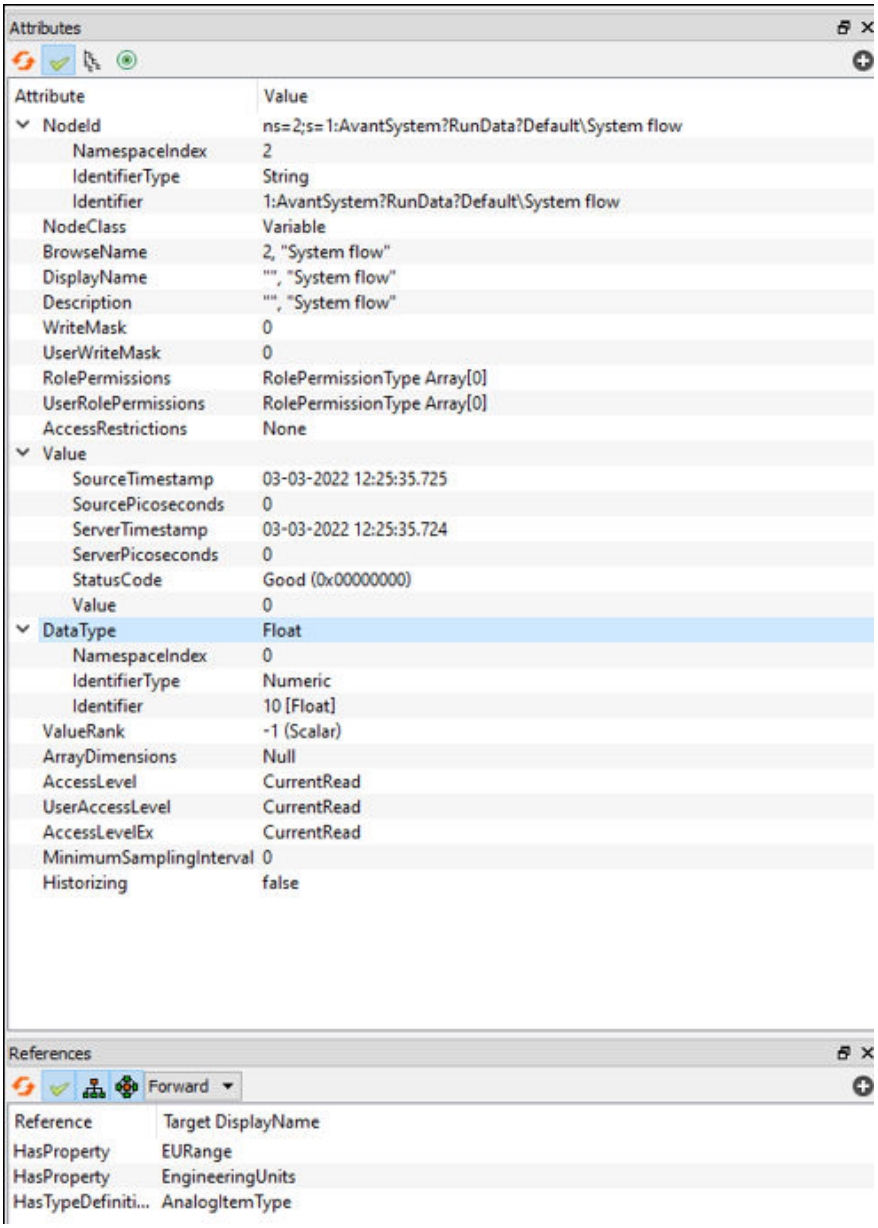
UNICORN run data are available via the `EVENT\Rundata` branch, and can be analog or digital depending on the data type. Picture data are available via `EVENT\PictureData`.

The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

```
Event\Rundata\Accumulated time
Event\Rundata\System flow
Event\Rundata\BufferPro pH
Event\Analog Picture\UV1
Event\Digital Picture\Frac Comp
```

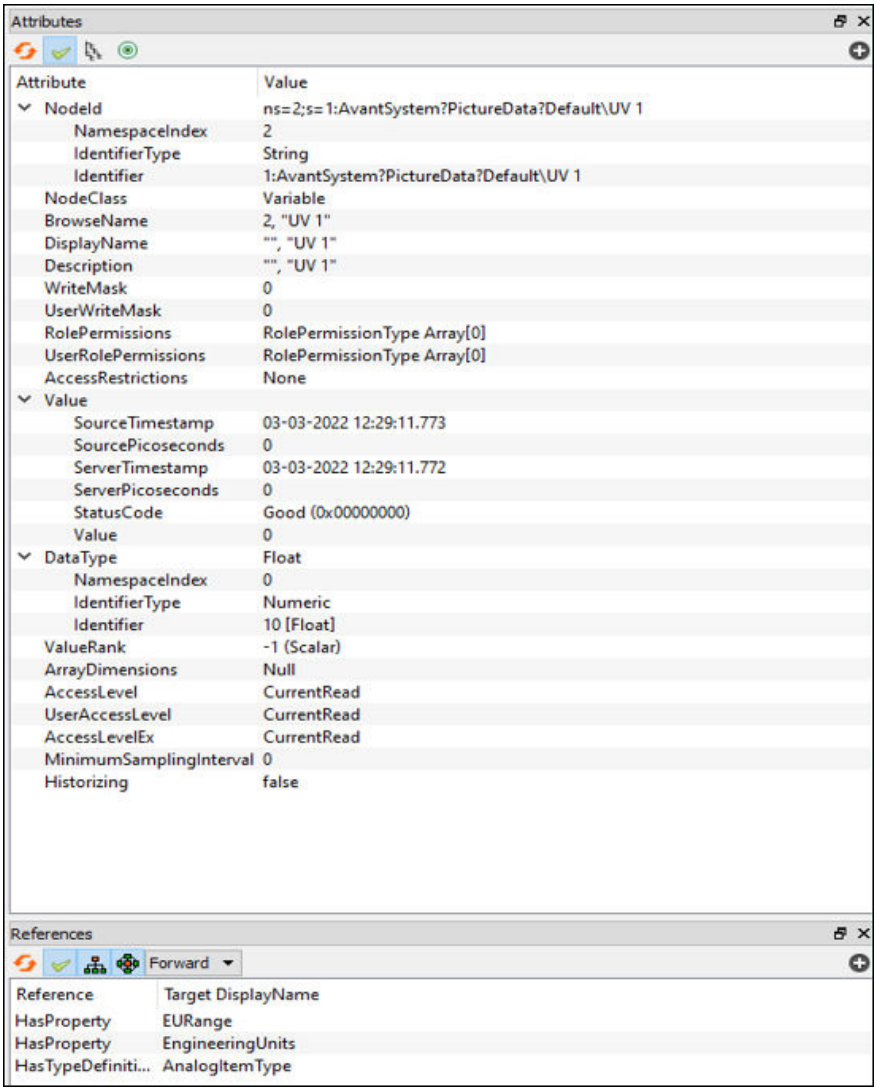
Properties for Run data

Properties for Run data are: (`Event\Rundata\System flow`)



Properties for Picture data

Properties for Picture data are: (`Event\Analog Picture\UV1`)



4.2 Trend data

Introduction

The UNICORN trend data are available via the `EVENT\Curves`. These data have the highest resolution. ***ANALOGTREND*** data are always an array of values.

BINARYTREND data

BINARYTREND contains textual trending data:

Data	Description
CurrentMethod	Name of the current method run.
CurrentResult	Name of the current result to which the method run is saved.
CurrentScouting	Scouting number.
CurrentBlock	Name of the block currently executing.
RunLog	Messages from the current run.
Fractions	Current fraction mark.
Injections	Current injection mark.

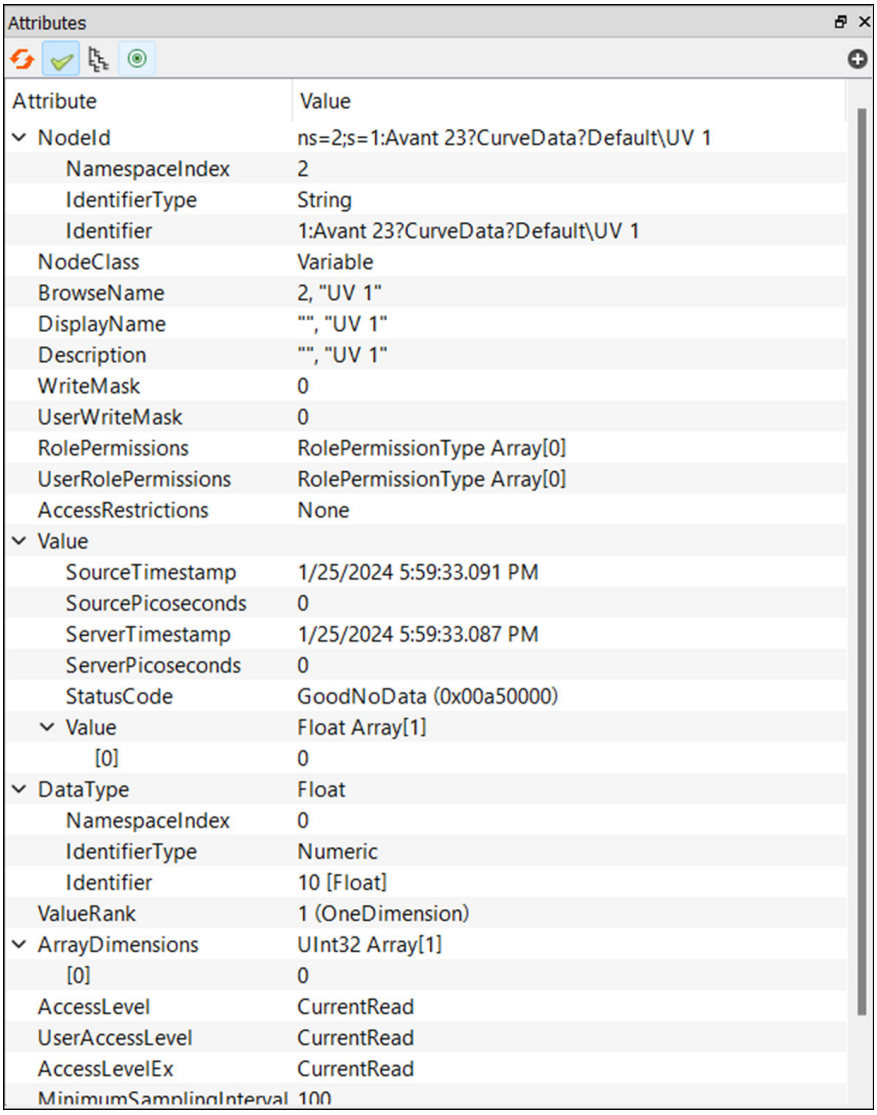
The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`Event\CurveData\UV1`

`Event\DiscreteData\Injections`

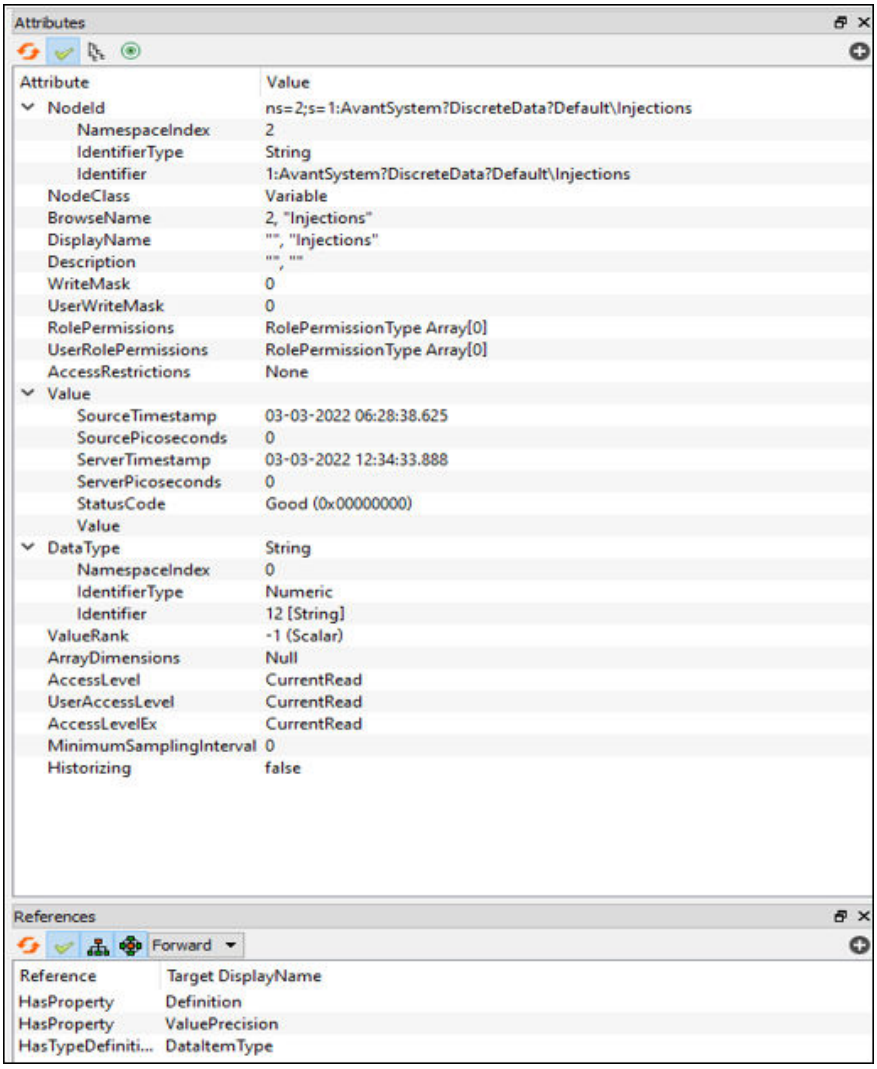
Properties for Analog data

Properties for Analog data are: (`Event\CurveData\UV1`)



Properties for Binary data

Properties for Binary data are: (`Event\DiscreteData\Injections`)



4.3 Manual instructions

Introduction

The UNICORN manual instructions from the ***Manual Instructions*** box in ***System Control*** are available in the ***INSTRUCTION*** branch. To execute a manual instruction, the user can either click on ***Call...*** to execute a method, or the ***Execution Instruction Tag*** can be used by providing the parameter value `1` to execute an item, or providing the parameter value `0` to ignore an instruction.

If the instruction in UNICORN uses text, it is possible to get that text from the EU type.

During method runs, manual instruction parameters will be updated with current values, which means that flow changes will be reflected in the flow parameter.

Execute Parameter

Execute parameter is a Method call without any arguments.

Note: *The instructions can only be executed if the module is in control of UNICORN system.*

Quality is good on successful execution of the instruction, which means that the corresponding manual command is successfully submitted.

The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

```
\Instruction\Pumps and pressures\System flow\Execute
```

4.4 Method execution

Introduction

The methods are visible in the OPC UA Client after they are created in the UNICORN ***Method*** module.

A UNICORN method can be run via the ***METHOD*** branch.

The ***METHOD*** branch is dynamic. Whenever a new method is created or an existing method is modified, the ***METHOD*** branch has to be browsed again to see the changes and the ***METHOD*** branch has to reconfigure the items to use. If a method is deleted, it is removed from the address space.

The tags generated under the ***Method*** folder do not include the user logged on.

Each method file contains several items.

Note: ***Monitoring Method*** *tag values will not update if the user re-browsed the main* ***METHOD*** *folder. To get the updated value reflected in the OPC UA Client, the user has to browse the method that they created.*

In this section

Section		See page
4.4.1	Scout Run Index	53
4.4.2	Run Method	55
4.4.3	Result Name	56
4.4.4	Batch ID	58
4.4.5	StartNotes	60
4.4.6	MethodNotes	61
4.4.7	PreCompile	62
4.4.8	Questions	63
4.4.9	Variables	65

4.4.1 Scout Run Index

Introduction

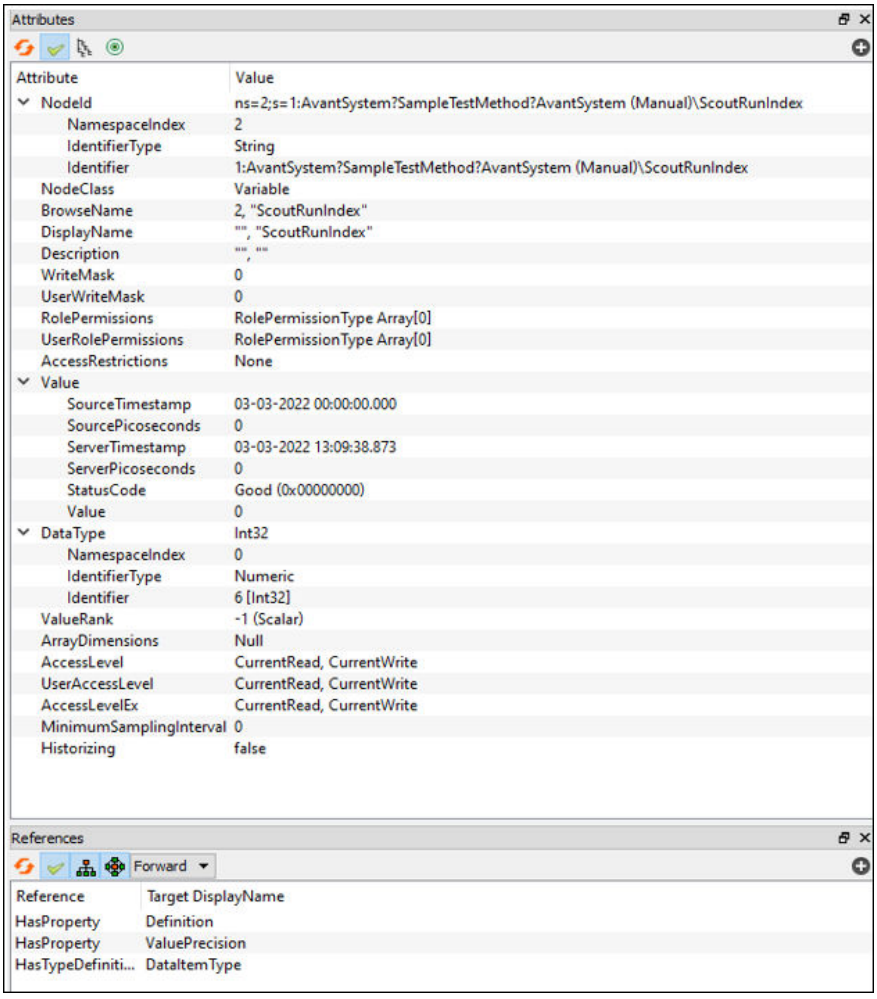
The ***Scout Run Index*** is used to include any scout runs. When a client writes `0`, all scout runs will be included in the method run, unless the excluded flag is set for any particular run. Entering any particular scout run number will include only that run in the method run. The data type is ***Int32***.

The object is always ***CurrentRead*** and ***CurrentWrite***. The object is writable only when in control of the system. Executing the method stores the text in the result.

If the user wants to insert a value for the scout variable, then the user needs to first insert a value for the ***Scout Run Index*** tag to specify the index. When the value is changed for the ***Scout Run Index*** tag, the user needs to re-browse the method to see the scout variable value for the particular ***Scout Run Index*** .

Implemented properties

The implemented scout run index properties are as follows:



Location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\Method\DefaultHome\AvantSystem(Manual)`
`\SampleTestMethod1\Scout Run Index`

4.4.2 Run Method

The available methods are read from the shared directory of the current user, OPC user, or all users, depending on the setup. ***Run Method*** is a method call with no arguments.

When a client writes ***3*** to the state command to end of the method run, the current run state is checked to see if the result is to be saved or not. Other run states imply that it is a manual run being executed, and the result is saved in the `[SystemName](Manual)` folder.

Location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\Method\DefaultHome\AvantSystem(Manual)\SampleTestMethod\Run Method`

4.4.3 Result Name

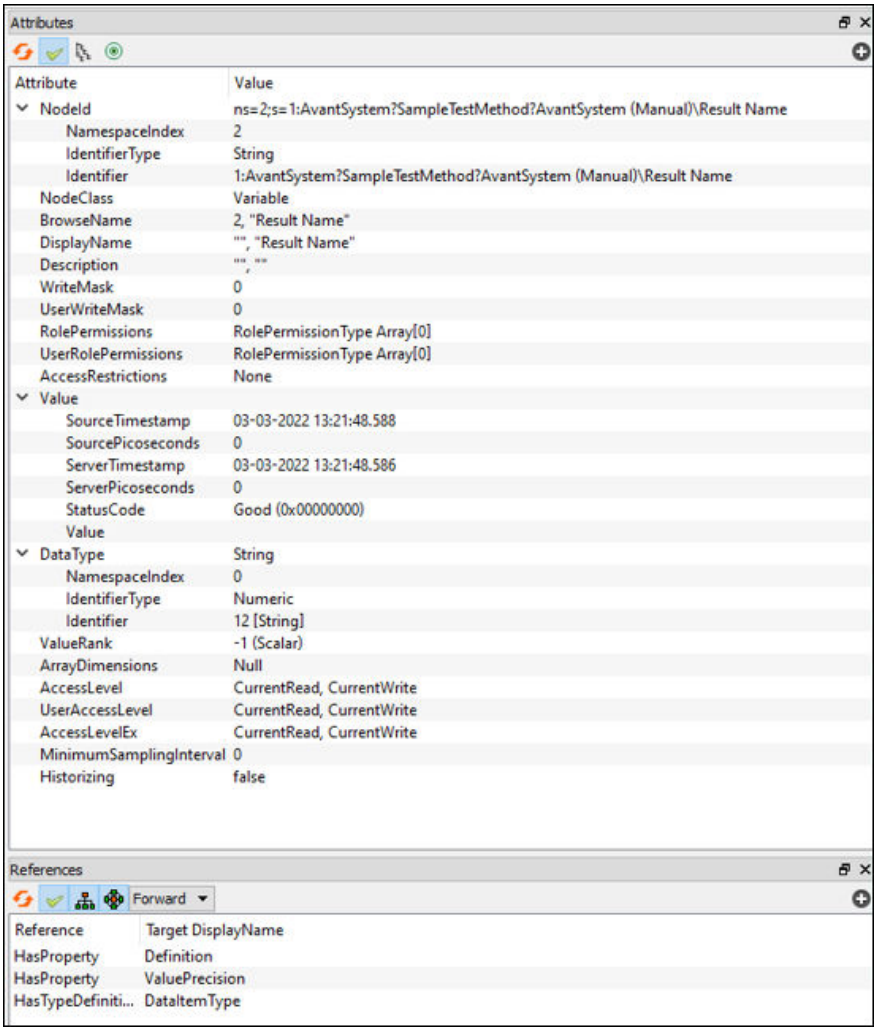
Introduction

The data type is `String`. Result name objects are both ***CurrentRead*** and ***CurrentWrite*** , but only readable from `CACHE`. If a result name is not set in OPC, then UNICORN generates a unique result name when the method starts.

Note: *The methods can only be run if the module is in control of the UNICORN system.*

Implemented properties

The implemented result name properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

```
\Method\DefaultHome\AvantSystem1(Manual)
\SampleTestMethod1\Result Name
```

4.4.4 Batch ID

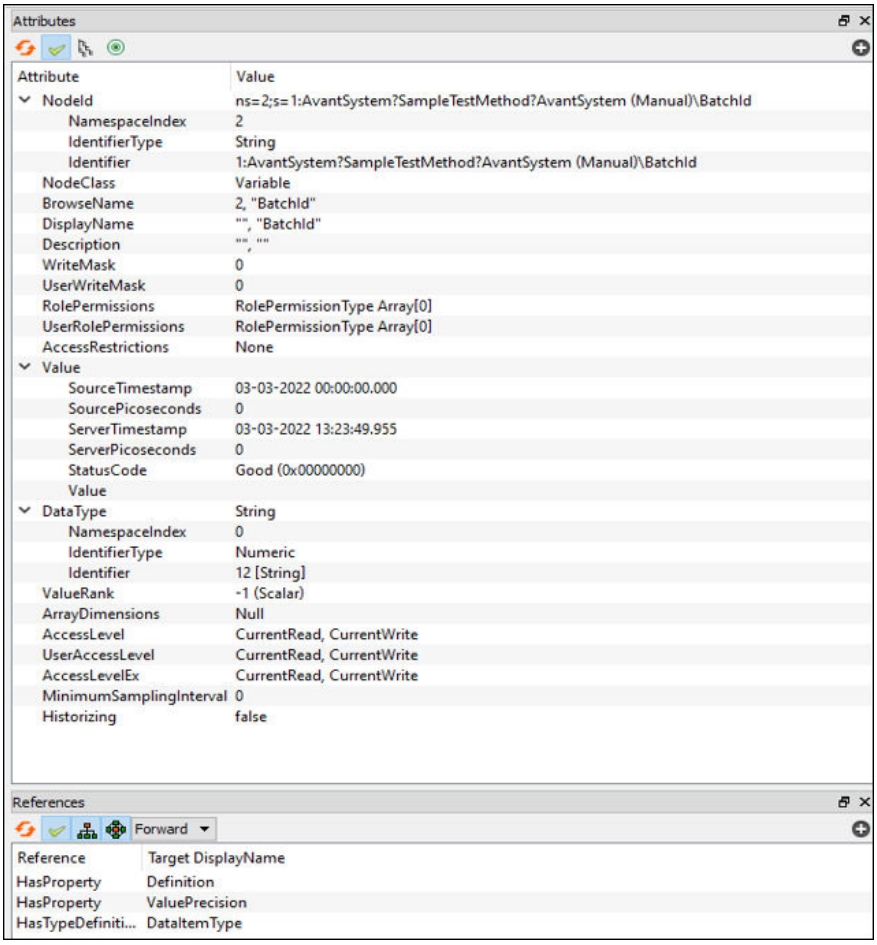
Introduction

The data type is `String`. Batch ID objects are both ***CurrentRead*** and ***CurrentWrite***. If a Batch ID is not set in OPC, then UNICORN generates a unique Batch ID when the method starts.

Note: *The methods can only be run if the module is in control of the UNICORN system.*

Implemented properties

The implemented Batch ID properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

```
\Method\DefaultHome\AvantSystem1(Manual)
\SampleTestMethod1\Batch ID
```

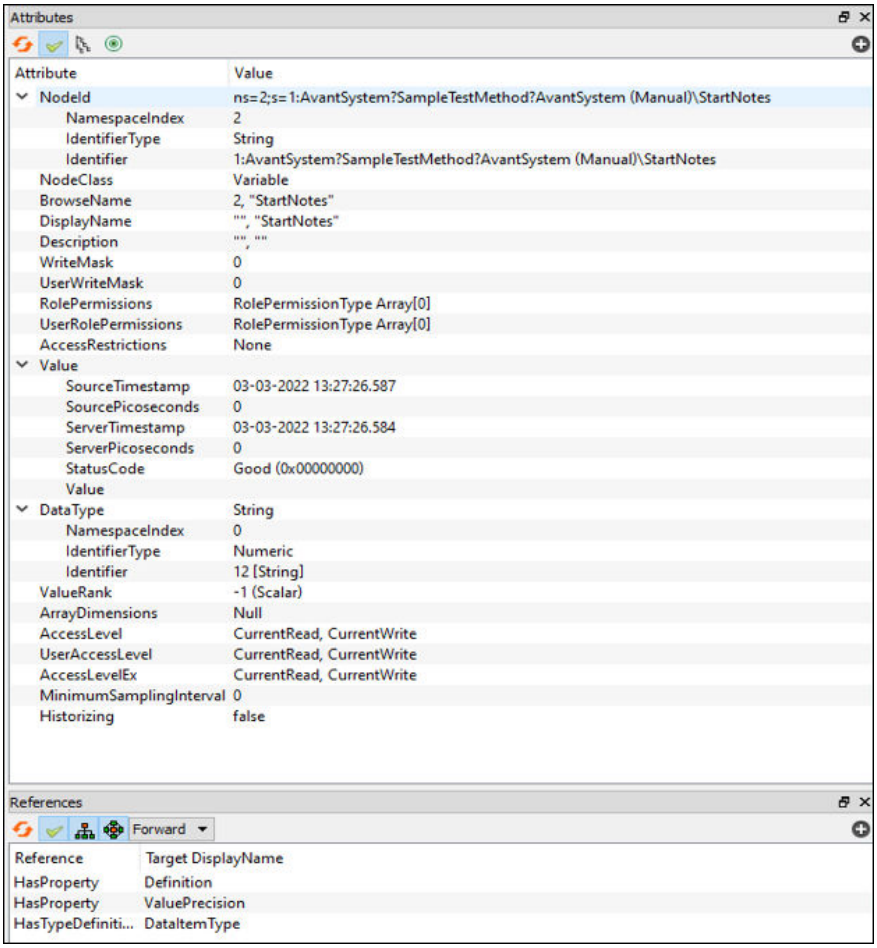
4.4.5 StartNotes

Introduction

The ***StartNotes*** is used to read and write the start notes. The data type is `String`. ***StartNotes*** objects are both ***CurrentRead*** and ***CurrentWrite***. The object is writable only when in control of the system. Executing the method stores the text in the result.

Implemented properties

The implemented start notes properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\Method\DefaultHome\AvantSystem1(Manual)`
`\SampleTestMethod1\Start Notes`

4.4.6 MethodNotes

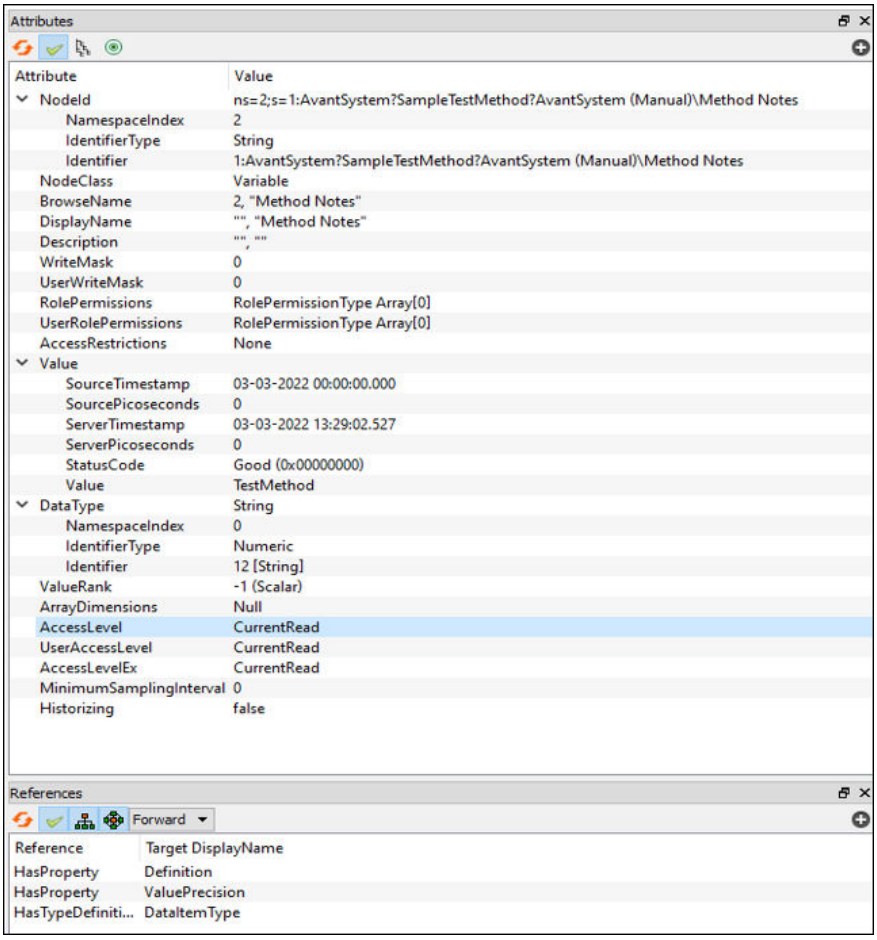
Introduction

The ***MethodNotes*** is used to read the method notes. The data type is `String`.

MethodNotes objects are ***CurrentRead***.

Implemented properties

The implemented method notes properties are as follows:



The following is an example location (<=>***Node/Parameter/Tag path***) in the server configuration tree:

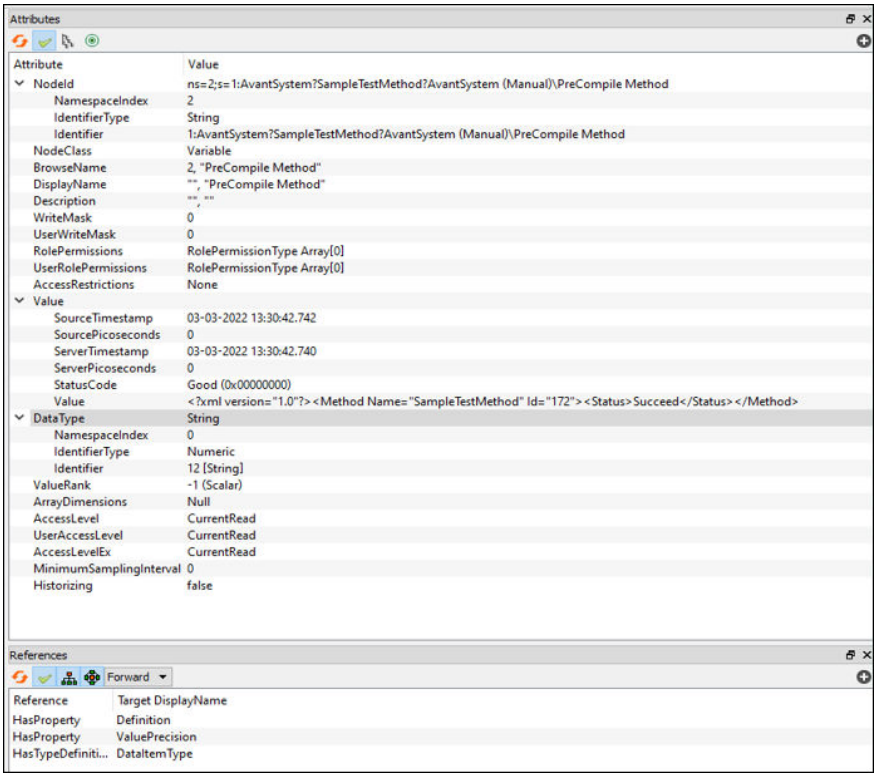
`\Method\DefaultHome\AvantSystem1(Manual)`
`\SampleTestMethod1\Method Notes`

4.4.7 PreCompile

The data type is always `String`. This forces UNICORN OPC to compile the method and check if it is runnable on the system. If it is runnable on the system, the result will be an xml string containing `Succeed` as the status. If it is not runnable on the system, the string will contain an error text.

Implemented properties

The implemented method compile properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\Method\DefaultHome\AvantSystem(Manual)`
`\SampleTestMethod1\PreCompile Method`

4.4.8 Questions

Introduction

The ***QUESTIONS*** branch is used to read and write the start protocol questions. One object is created for each question in a start protocol. Objects are added to the ***QUESTIONS*** branch for every method. The canonical data type is either ***String*** or ***Int32***, depending on the question type.

Table 4.1: Question types and data types

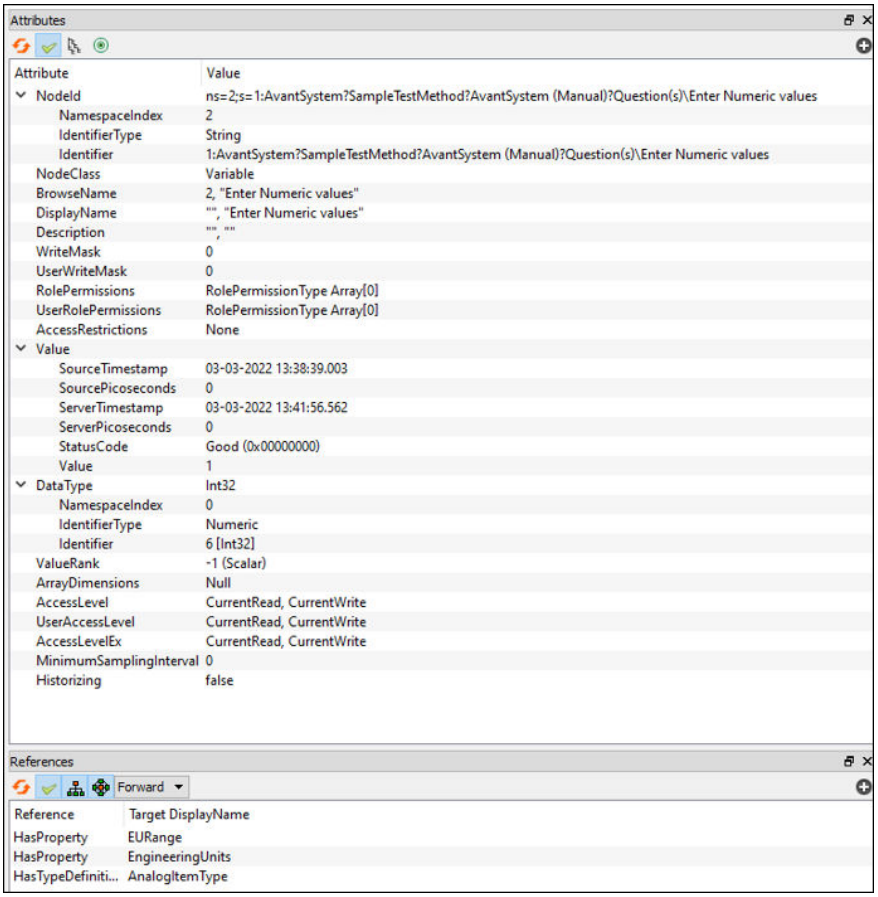
Data type	Question type
NoReply	***Int32***
Entry field	***String***
Multiple choice	***Int32***
Integer	***Int32***
Float	***Float***

Integer and ***Float*** can have minimum and maximum ranges. The writing out of range is not allowed. Empty strings are valid unless the mandatory flag is set. Authorized questions are allowed without supplying any signature. Also, it is possible to run methods without answering all the questions, unless the mandatory flag is set.

Question objects are ***CurrentRead*** and ***CurrentWrite***. The object is writable only when in control of the system. Executing the method stores the answers in the result.

Implemented properties

The implemented properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

```
\Method\DefaultHome\AvantSystem(Manual)
\SampleTestMethod1\Question(s)\Enter Numeric Values
```

4.4.9 Variables

Introduction

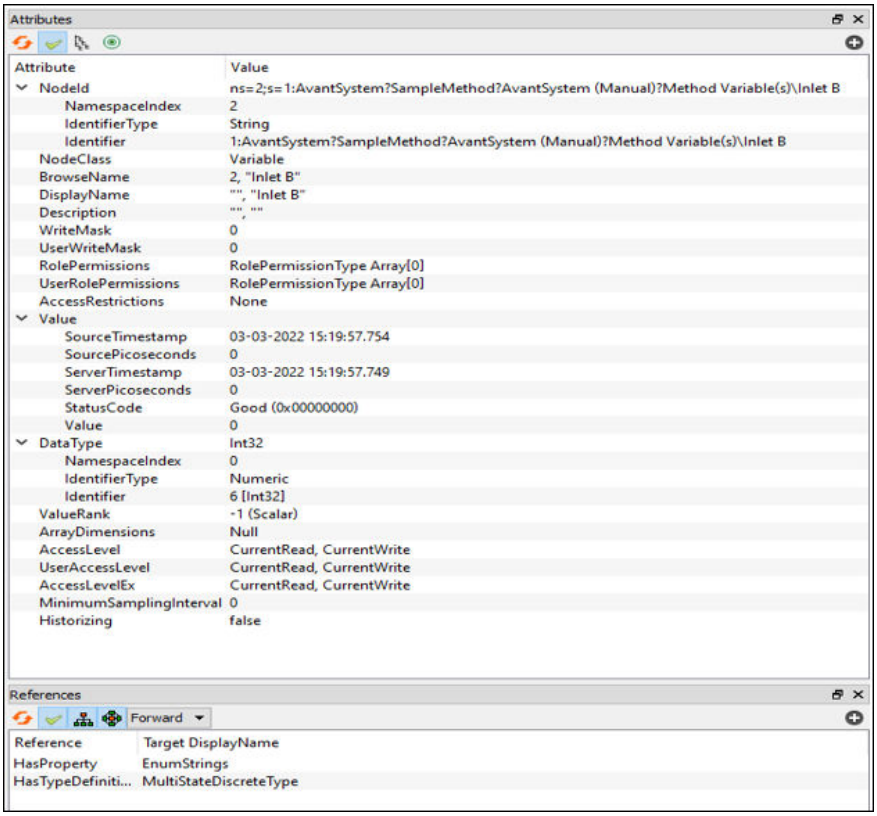
The ***VARIABLES*** branch is used to read and write the method variables. One object is created for each method variable. Objects are added to the ***VARIABLES*** branch for every method, if the ***StartProtocol*** for the method supports the variable value change. The variables on the following are not supported:

- Frac-950 Fractionation instruction parameters
- Frac-950 Peak Fractionation parameters

Variables objects are ***CurrentRead*** and ***CurrentWrite***. The object is writable only when in control of the system. Executing the method uses the variable settings during the method run.

Implemented properties

The implemented start protocol variables general properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

```
\Method\DefaultHome\AvantSystem(Manual)
\SampleTestMethod1\Method Variable(s)\Inlet B
```

4.5 State

Introduction

The ***STATE*** branch contains important state information for the system.

In this section

Section		See page
4.5.1	AssignState	68
4.5.2	RunState	70
4.5.3	Command	72
4.5.4	InstrumentStatus	74
4.5.5	InstrumentStatusConnection	76
4.5.6	UserInControl	78

4.5.1 AssignState

Introduction

AssignState enables the client to change the assign mode to the UNICORN system. The default assign mode is ***View***, which is a mode where the client can monitor the ***EVENT*** and ***STATE*** branch items. By changing the mode to ***Control***, it is also possible to execute instructions, run methods, and save system settings.

Object values

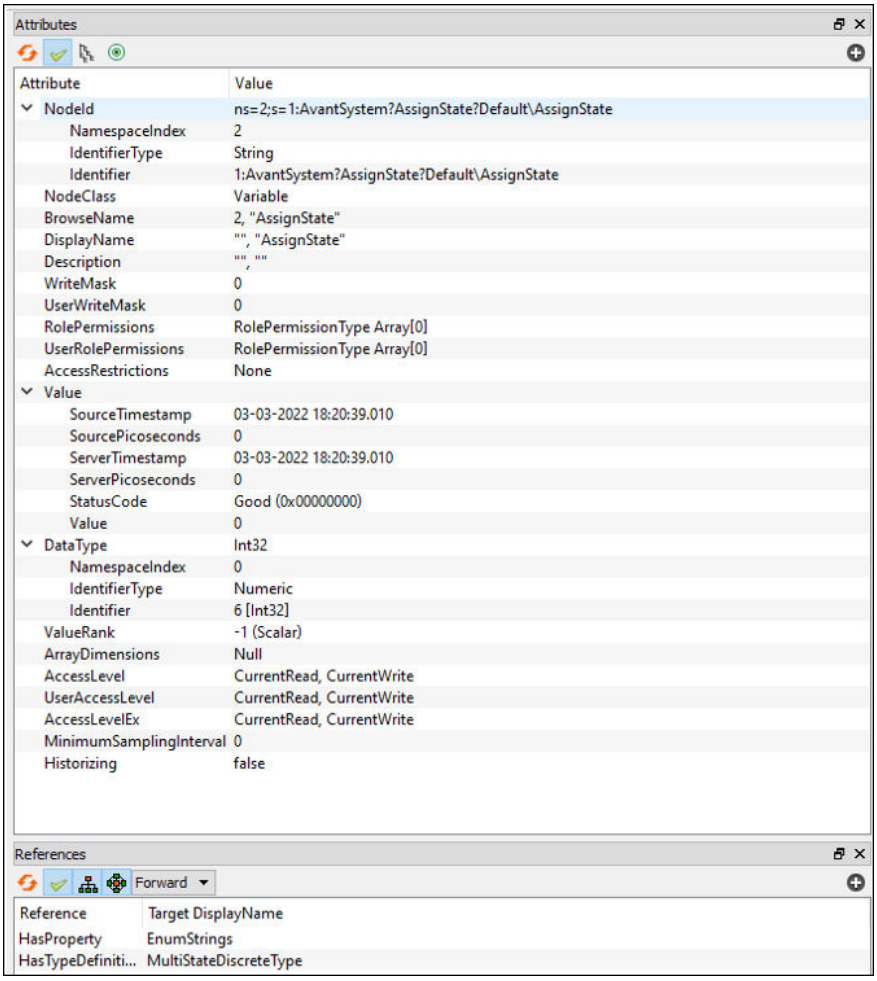
The defined assign state values are as follows:

Value	State
0	***View***
1	***Control***
2	***None***
3	***Disconnected***

The quality is good if a correct ***AssignState*** is written to the object, otherwise it is bad.

Implemented properties

The implemented assign state properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\STATE\AssignState`

4.5.2 RunState

Object values

The data type is ***String***.

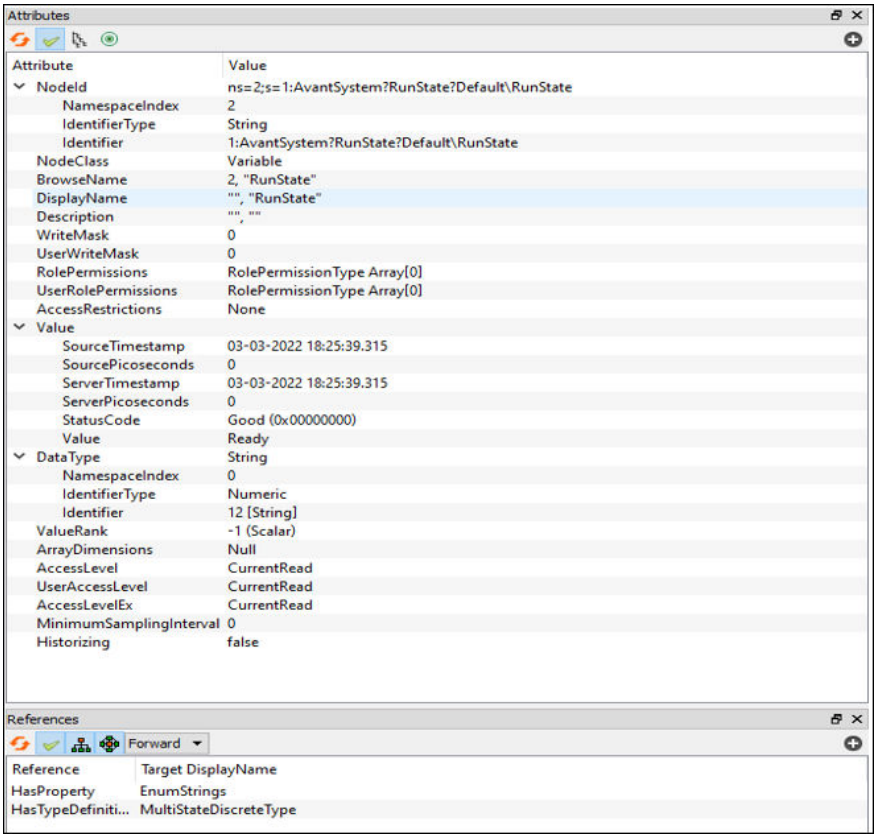
The defined object values (meaning the state) are as follows:

- ***Ready***
- ***Method Run***
- ***Manual Run***
- ***System Pause***
- ***Hold***
- ***Manual Pause***
- ***Wash***
- ***Alarms and errors***
- ***Initializing system***
- ***Resetting***
- ***Starting method run***
- ***Starting manual run***

The quality is good on a successful request of run state data or on a valid update from the online data, otherwise it is bad.

Implemented properties

The implemented run state properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\STATE\RunState`

4.5.3 Command

Object values

The canonical data type is ***Int32***

The defined object values are as follows:

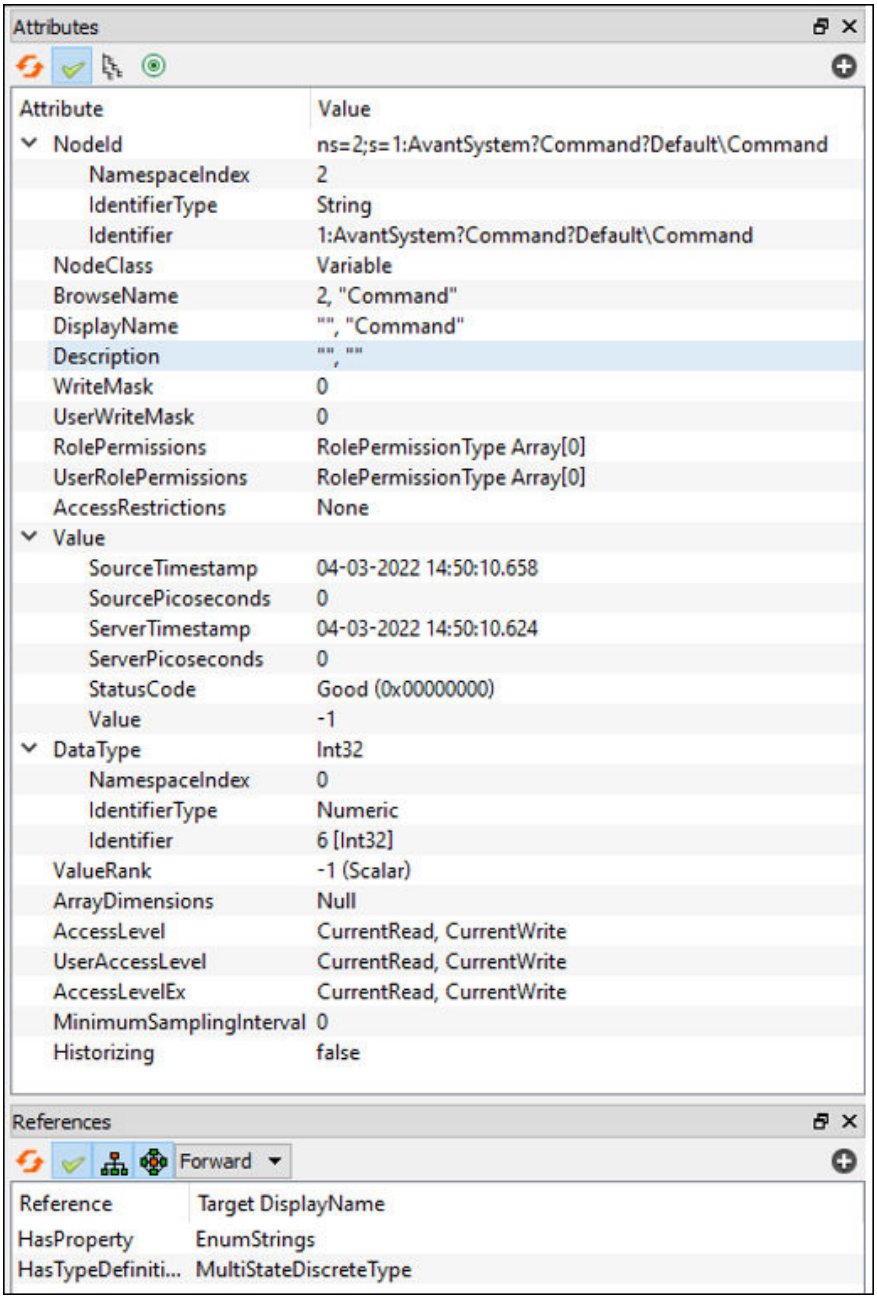
Value	State
0	***Pause***
1	***Hold***
2	***Continue***
3	***End***
4	***Next breakpoint***
5	***None***

The method command objects are ***CurrentRead*** and ***CurrentWrite***. Writing a valid value causes the utility object to submit a manual command.

When a client writes ***END (3)***, the current run state is checked to see if the result should be saved or not. If a method is running, the run state is ***PAUSE (0)*** or ***HOLD (1)***, and the result is saved. Other run state values imply a manual instruction is being executed, and the result is created in the `[SystemName] (Manual)` folder.

Implemented properties

The implemented general command properties are:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\STATE\Command`

4.5.4 InstrumentStatus

Object values

The ***InstrumentStatus*** handles the instrument connection state. The data type is ***Int32***.

The defined object values are as follows:

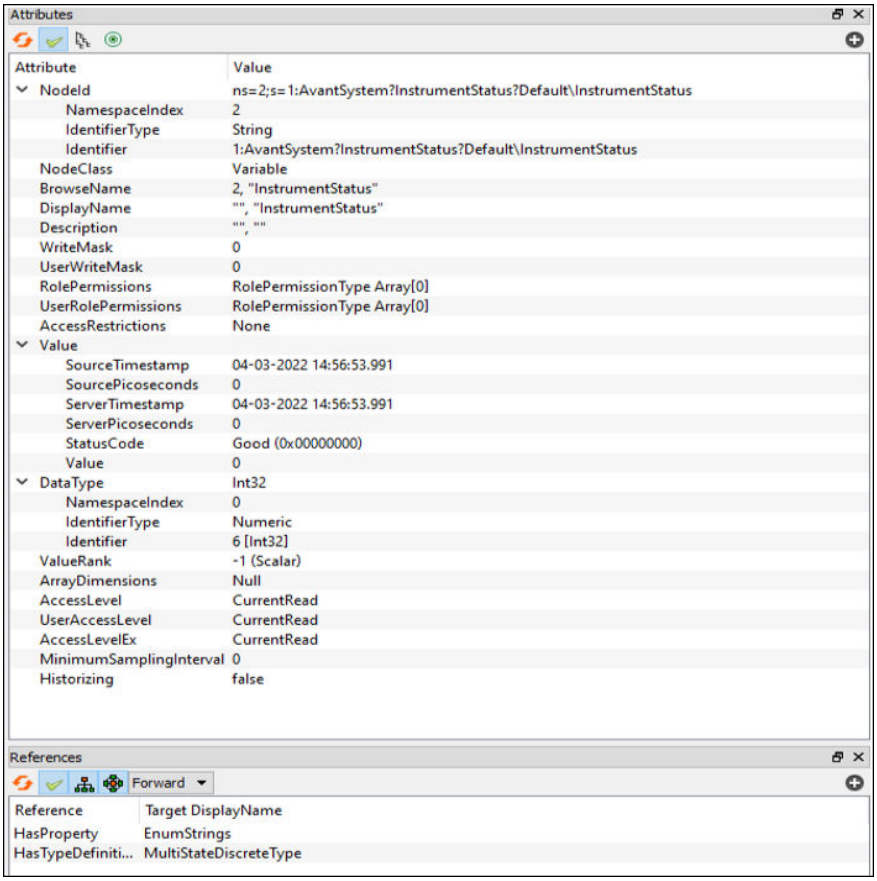
Value	State
0	***Ready***
1	***Scanning***
2	***Unknown***

InstrumentStatus objects are ***CurrentRead***.

Note: *The instrument connection must be valid (connected) for the other items to work as expected.*

Implemented properties

The implemented instrument connection properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\STATE\InstrumentStatus`

4.5.5 InstrumentStatusConnection

Object values

The ***InstrumentStatusConnection*** handles the instrument status state. The data type is ***Int32***.

The defined object values are as follows:

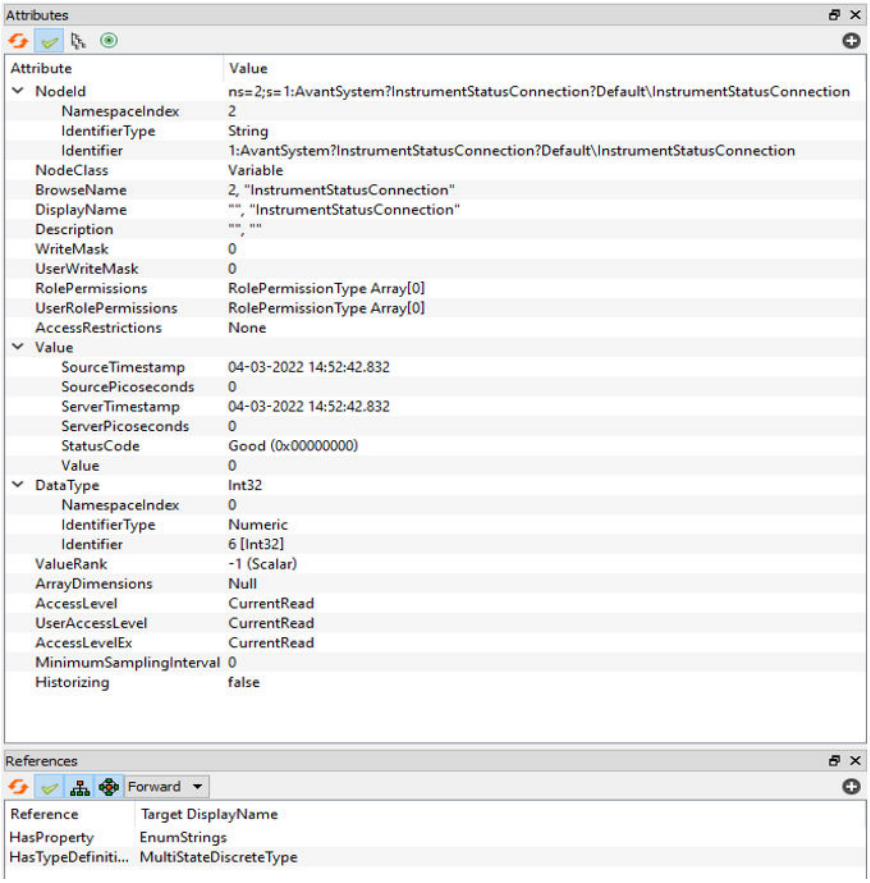
Value	Connection
0	Yes
1	Partially
2	No

InstrumentStatusConnection objects are ***CurrentRead.***.

Note: *The instrument status must be valid (connected) if the other items are to work as expected.*

Implemented properties

The implemented instrument status properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\STATE\InstrumentStatusConnection`

4.5.6 UserInControl

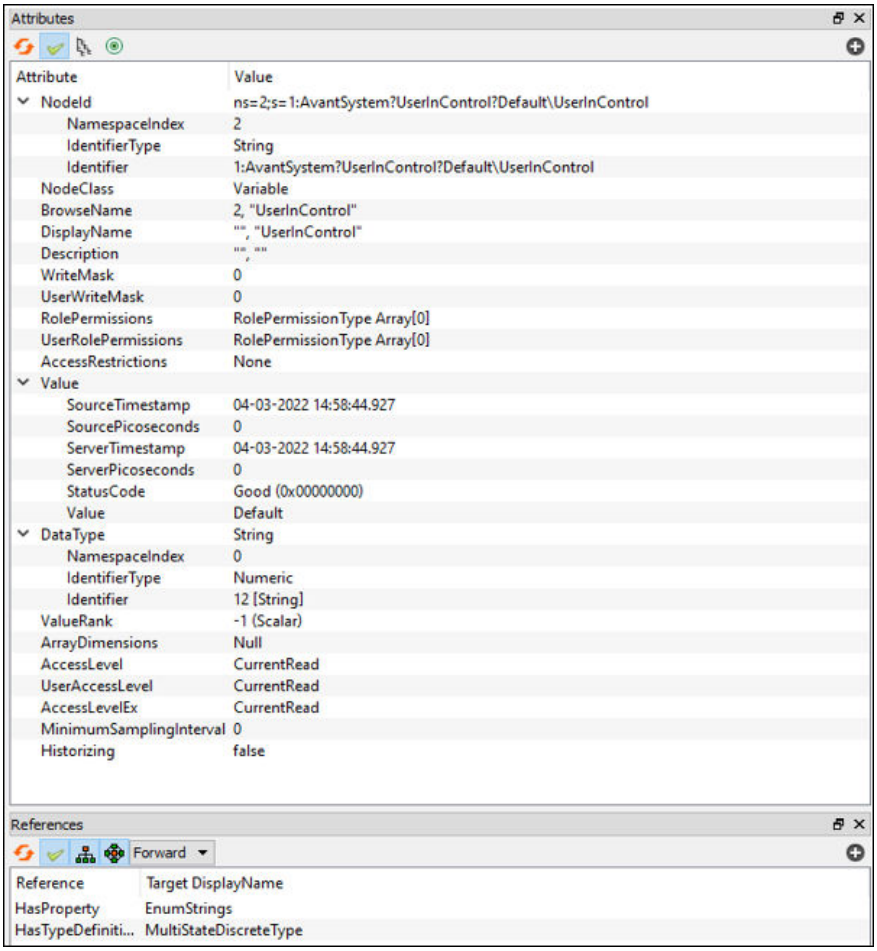
Object values

UserInControl handles the instrument control state. The canonical data type is ***String***. The defined object value is the user name of the current user having control.

UserInControl objects are ***CurrentRead***.

Implemented properties

The implemented instrument status properties are as follows:



The following is an example location (<=> ***Node/Parameter/Tag path***) in the server configuration tree:

`\STATE\UserInControl`

Note: ***Run*** *state and* ***Assign*** *state tag value reflect disconnection of system once user disconnects system from OPC UA Client.*

4.6 Recommendations

We recommend writing to items synchronously. The server handles both synchronous and asynchronous writes, but when making asynchronous writes, it is up to the client to wait until the server is ready with a previous asynchronous write.

Note:

- *The UNICORN system settings are available via the **SETTINGS** branch, which works like the **INSTRUCTION** branch with one exception. There is only one execute instruction, `\Setting\Store`, to store all system settings at once.*
- *System settings can only be saved when the system is in **Ready** state. If UNICORN changes system settings, it updates the settings parameters in OPC.*

5 UNICORN OPC Alarms and Events address space

About this chapter

This chapter describes the UNICORN OPC Alarms and Events address space, various types of alarms, error messages, and event categories.

The Alarms and Events address space shows the internal alarm functionality of the instrument configuration. Each item in the address space might trigger an alarm or event.

Each component's failure/loss triggers an alarm and the Alarms and Events (AE) client connected in control/view of the system is notified. Alarms, events, and warnings are notified only when the user drags a ***Server*** object and expands or connects a node for a system under the ***AlarmsEvents*** folder. If the connection to the instrument is lost, then the AE client is notified through the restart error alarm. Once the restart error occurs and the connection issue has been fixed, then either the UNICORN Instrument Server computer must be restarted, or the restart error must be acknowledged in UNICORN ***System Control***.

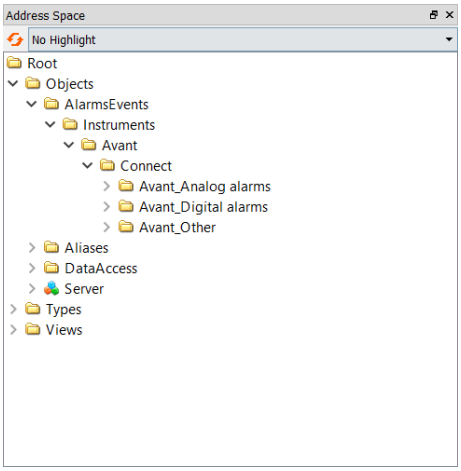
To acknowledge alarms or warnings from the OPC UA Client, the OPC UA Client (UNICORN OPC Server) must be in control of the system. Use the ***Data Access*** address space to take control over the system via the ***AssignState*** item.

The UNICORN OPC and Alarms and Events (AE) server does not support the inactive/acknowledged/enabled state. The UNICORN OPC AE server will not be able to notify that the alarm or warning becomes inactive. The UNICORN OPC AE server only notifies if the signal reaches an alarm or warning state, but it will not notify the opposite when reaching the normal state.

Note: *All alarms raised from the same source will not be available to acknowledge. Only the last alarm, warning, or error generated from the same source will be available to acknowledge. However, all the alarms and warnings will be logged under the events tab. For example, when alarms 1 to 10 are generated from the same source, only alarm 10 will be available to acknowledge. The alarms disappear from the queue once they are acknowledged.*

Alarms and Events address space view

The following screenshot shows the Alarms and Events address space view.



Tip: *The address space depends on the instrument configuration. Different items will appear depending on which instrument configuration is used.*

In this chapter

Section		See page
5.1	Alarms and errors	82
5.2	Event categories	84

5.1 Alarms and errors

Analog alarm

There are always four conditions for analog alarm objects (sources). Corresponding condition names are as follows:

1. ***LO_ALARM***
2. ***LO_WARNING***
3. ***HI_ALARM***
4. ***HI_WARNING***

Both ***Condition Events*** and ***Tracking Events*** can be generated. ***Condition Events*** are generated when a condition becomes active or inactive and when a condition is acknowledged. ***Tracking Events*** are generated when a condition is disabled but still active or inactive and unacknowledged. The event categories are ***Level*** for ***Condition Events*** and ***Enable/Disable*** for ***Tracking Events***. No attributes are supported for analog alarms.

Example location (<=> ***OPC qualified source name***) in server area space:

```
Analog alarms.Cond2
```

Digital alarm

There is only one condition for each digital alarm object (source). The condition names for the two different types are:

1. ***WARNING***
2. ***ALARM***

Both ***Condition Events*** and ***Tracking Events*** can be generated. ***Condition Events*** are generated when a condition becomes active or inactive and when a condition is acknowledged. ***Tracking Events*** are generated when a condition is disabled but still active or inactive and unacknowledged. Event categories are ***DiscreteW*** or ***DiscreteA*** for ***Condition Events*** and ***Enable/Disable*** for ***Tracking Events***. No attributes are supported for digital alarms.

Example location (<=> ***OPC qualified source name***) in server area space:

```
Digital alarms.FlowWarning
```

Errors

This is the error dialog box in UNICORN ***System Control***. The event category of the condition error event is ***Errors***.

No attributes are defined for error events and no acknowledgements are required for this event type.

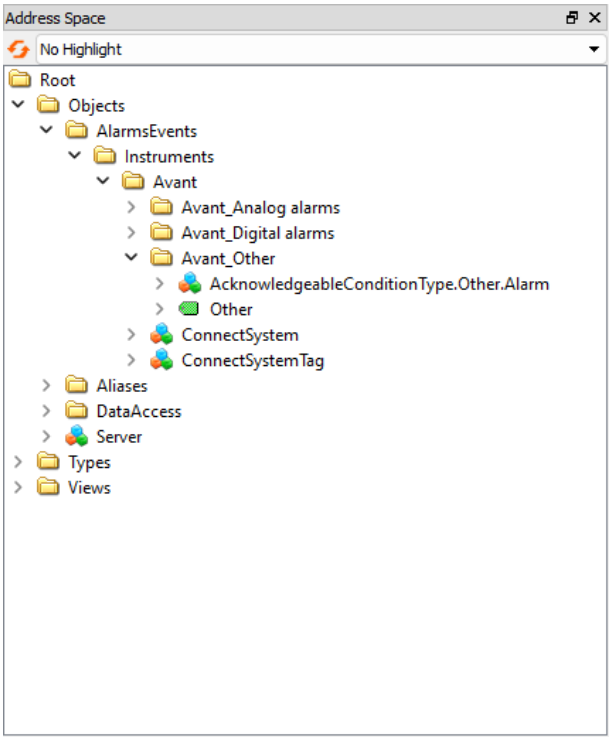
Location (<=> ***OPC qualified source name***) in server area space:

`Other.Errors`

5.2 Event categories

Alarms and Events address space view

The following screenshot shows the Alarms and Events address space view.



6 UNICORN OPC Historical Data Access address space

About this chapter

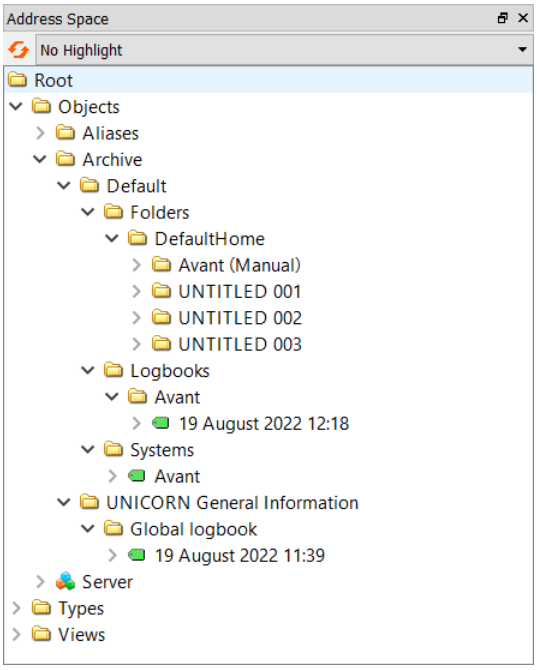
This chapter provides information about the UNICORN OPC Historical Data Access (HDA) address space, audit trail, HDA XML format definition, etc. The HDA server supports Read raw data.

The HDA address space is dynamic. A result file can be browsed whenever a new result is created. If a result is deleted, it is removed from the address space.

Tip: *The address space depends on the result files that have been created based on a specific instrument configuration. Different items will appear in the result file depending on the instrument configuration used when the result file was created.*

Historical Data Access address space view

The following screenshot shows the HDA address space view.



Note: *The address space depends on the result files that have been created based on a specific instrument configuration. Different items will appear in the result file depending on which instrument configuration that was used when the result file was created.*

In this chapter

Section		See page
6.1	AuditTrail	87
6.2	Result file information	88
6.3	UNICORN OPC Historical Data Access XML format definition	94

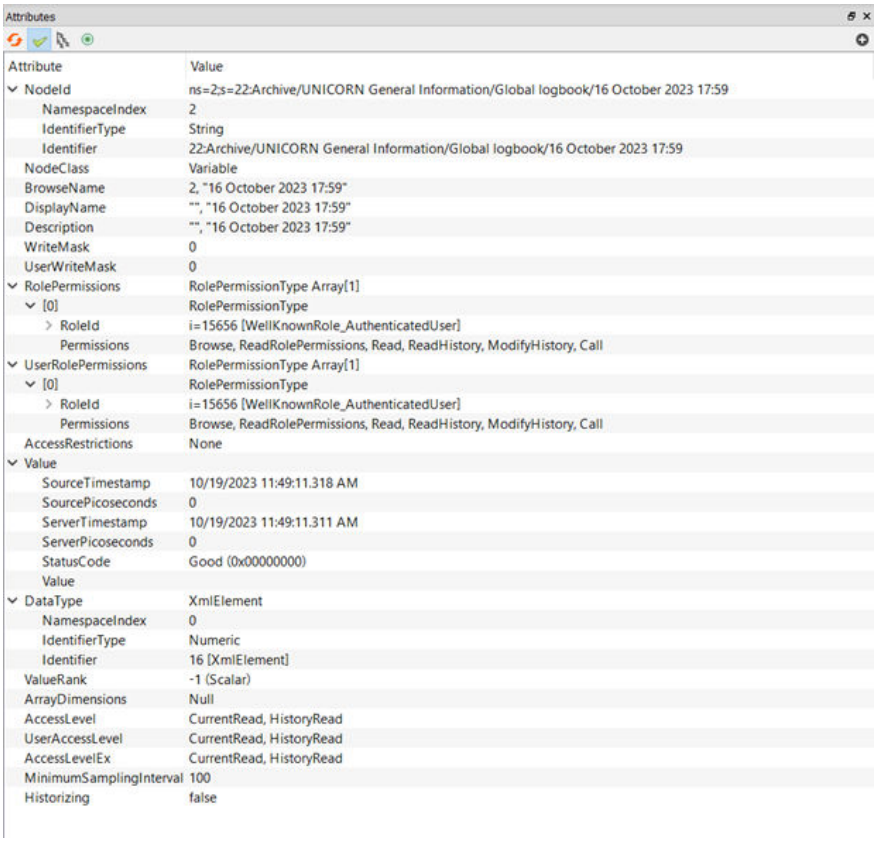
6.1 AuditTrail

Introduction

The ***AuditTrail*** from UNICORN is available under the HDA Server address space. The global logbook is found under the ***UNICORN*** →***GENERAL INFORMATION*** branch and the system specific logs are found under the ***Logbooks*** folder in the user (e.g., ***Default***) folder.

Attributes defined by *AuditTrail*

Each ***AuditTrail*** leaf (item) has the following attributes defined:



The start and end time are equal for a leaf. The time is set to the creation time of the audit trail file, meaning the renew time.

6.2 Result file information

Introduction

Each result file contains ***Curves***, ***Documentation***. and ***Peak tables***.

Information about the system is available under the ***Systems*** folder in the user folder.

In this section

Section		See page
6.2.1	Curves	89
6.2.2	Documentation	90
6.2.3	Peak tables	92
6.2.4	Systems	93

6.2.1 Curves

Introduction

All curves in a result file are listed as items directly under the result file branch in the address space.

Attributes supported by Curves

Each curve has several supported attributes (see below) that allow the kind of curve that the item represents, to be identified.

Attribute	Value
⌄ NodeId	ns=2;s=13:Archive/Default/Folders/DefaultHome/DefaultHome/UNICORN 6.2 Avant 150/pH 3 5/pH 3 5:Chrom.1:% Cond
NamespaceIndex	2
IdentifierType	String
Identifier	13:Archive/Default/Folders/DefaultHome/DefaultHome/UNICORN 6.2 Avant 150/pH 3 5/pH 3 5:Chrom.1:% Cond
NodeClass	Variable
BrowseName	2, "pH 3 5:Chrom.1:% Cond"
DisplayName	"", "pH 3 5:Chrom.1:% Cond"
Description	"", "pH 3 5:Chrom.1:% Cond"
WriteMask	0
UserWriteMask	0
⌄ RolePermissions	RolePermissionType Array[1]
⌄ [0]	RolePermissionType
› RoleId	i=15656 [WellKnownRole_AuthenticatedUser]
Permissions	Browse, ReadRolePermissions, Read, ReadHistory, ModifyHistory, Call
⌄ UserRolePermissions	RolePermissionType Array[1]
⌄ [0]	RolePermissionType
› RoleId	i=15656 [WellKnownRole_AuthenticatedUser]
Permissions	Browse, ReadRolePermissions, Read, ReadHistory, ModifyHistory, Call
AccessRestrictions	None
⌄ Value	
SourceTimestamp	10/19/2023 11:54:27.792 AM
SourcePicoseconds	0
ServerTimestamp	10/19/2023 11:54:27.778 AM
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	
⌄ DataType	Float
NamespaceIndex	0
IdentifierType	Numeric
Identifier	10 [Float]
ValueRank	-1 (Scalar)
ArrayDimensions	Null
AccessLevel	CurrentRead, HistoryRead
UserAccessLevel	CurrentRead, HistoryRead
AccessLevelEx	CurrentRead, HistoryRead
MinimumSamplingInterval	100
Historizing	false

6.2.2 Documentation

Introduction

Evaluation in UNICORN has a documentation dialog for each result file. The ***DOCUMENTATION*** branch for each result file in the HDA contains the same data as ***Evaluation*** in documentation.

Items in *Documentation*

The following items are available:

Item	Data type
BatchNumber	***String***
BufferPro	***XML string***
Calibration	***XML string***
Columns	***XML string***
Evaluation logbook	***XML string***
Evaluation procedures	***XML string***
FracXY	***XML string***
Run logbook	***XML string***
Method name	***String***
Method creator	***String***
Method creation date	***String***
Method created for system	***String***
Method last modifier	***String***
Method last modification date	***String***
Method signatures	***XML string***
Method notes	***XML string***
Start notes	***XML string***
Run notes	***XML string***
Evaluation notes	***XML string***
Method strategy notes	***String***
Result name	***String***

Item	Data type
Result creator	***String***
Result creation date	***String***
Result run system	***String***
Batch number	***String***
Result signatures	***XML string***
Result strategy used components	***XML string***
Scouting	***XML string***
System Settings	***XML string***
Snapshot	***XML string***
Text Instructions	***XML string***
Variables	***XML string***
Method Instrument Configuration	***XML string***
Result Instrument Configuration	***XML string***
Questions	***XML string***

6.2.3 Peak tables

The ***PEAK TABLES*** branch contains all peak tables stored in the result file. When reading from the item, the peak table is returned as an XML-formatted string.

6.2.4 Systems

Introduction

Information about the system is available under the ***Systems*** folder in the user folder.

Items in *Systems*

Each system leaf defines the following attributes:

Attributes

Attribute	Value
NodeId	ns=2;s=21:Archive/Default/Systems/Wave
NamespaceIndex	2
IdentifierType	String
Identifier	21:Archive/Default/Systems/Wave
NodeClass	Variable
BrowseName	2, "Wave"
DisplayName	"", "Wave"
Description	"", "Wave"
WriteMask	0
UserWriteMask	0
RolePermissions	RolePermissionType Array[1]
[0]	RolePermissionType
RoleId	i=15656 [WellKnownRole_AuthenticatedUser]
Permissions	Browse, ReadRolePermissions, Read, ReadHistory, ModifyHistory, Call
UserRolePermissions	RolePermissionType Array[1]
[0]	RolePermissionType
RoleId	i=15656 [WellKnownRole_AuthenticatedUser]
Permissions	Browse, ReadRolePermissions, Read, ReadHistory, ModifyHistory, Call
AccessRestrictions	None
Value	
SourceTimestamp	10/19/2023 12:44:53.330 PM
SourcePicoseconds	0
ServerTimestamp	10/19/2023 12:44:53.302 PM
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	
DataType	XmlElement
NamespaceIndex	0
IdentifierType	Numeric
Identifier	16 [XmlElement]
ValueRank	-1 (Scalar)
ArrayDimensions	Null
AccessLevel	CurrentRead, HistoryRead
UserAccessLevel	CurrentRead, HistoryRead
AccessLevelEx	CurrentRead, HistoryRead
MinimumSamplingInterval	100
Historizing	false

6.3 UNICORN OPC Historical Data Access XML format definition

Introduction

The XML format is defined for each result file item. This section describes the different XML structures.

In this section

Section		See page
6.3.1	BufferPro	96
6.3.2	Calibration	97
6.3.3	EvaluationLog	98
6.3.4	Run logbook	99
6.3.5	SignatureList	100
6.3.6	SnapshotList	101
6.3.7	UsedComponents	102
6.3.8	Notes	103
6.3.9	ScoutingList	104
6.3.10	SettingsList	105
6.3.11	TextInstructions	106
6.3.12	VariableList	107
6.3.13	QuestionList	108
6.3.14	PeakTable	109
6.3.15	Unicorn raw	112
6.3.16	Columns	114
6.3.17	EvaluationProcedures	116
6.3.18	FracXY	117
6.3.19	AuditTrail	118
6.3.20	Method/Result Instrument Configuration	119
6.3.21	System	120

6.3.1 BufferPro

Introduction

This XML structure is exported by reading the ***BufferPro*** leaf.

Note: *For migrated UNICORN 5.x results, the* ***BufferPro*** *OPC item contains* ***BufferPrep*** *data. Both the S function and the S function share the same storage format.*

BufferPro XML structure

Tag name	Description
BufferPro	Root item
RecipeName	Recipe name
pHRange	pH range, separated with "**-**"
B100	100% buffer B
AcidOrBase	Acid or base name
Salt	Salt name and salt stock concentration
Notes	Recipe name
Buffer	Specifications of the different buffers
Name	Buffer name
pKa1	pK_a1
pKa2	pK_a2
pKa3	pK_a3
dpKa1dt	dpk_a1 delta
dpKa2dt	dpk_a2 delta
dpKa3dt	dpk_a3 delta
AcidicProtons	Number of acidic protons
ChargeDeprotonatedIon	Number of charge deprotonated ions
SaltName	Salt name
ChargeAnion	Number of charged anions
ChargeCation	Number of charged cations

6.3.2 Calibration

Introduction

This XML structure is exported by reading the ***Calibration*** leaf.

Calibration XML structure

Tag name	Description
CalibrationList	Root item
Calibration	A calibration
Tagname	Name of item that has been calibrated
Date	Date
Username	User name
Point1	Point 1
Point2	Point 2
Constant	Constant
Offset	Offset
Remote	Calibration local or remote. (YES/NO)
P	Proportional control component
I	Integral control component
D	Derivative control component
ReferenceValue1	Reference value 1
ReferenceValue2	Reference value 2
CalibrationConstantDescription1	Calibration constant description 1
CalibrationConstantValue1	Calibration constant value 1
CalibrationConstantDescription2	Calibration constant description 2
CalibrationConstantValue2	Calibration constant value 2
CalibrationConstantDescription3	Calibration constant description 3
CalibrationConstantValue3	Calibration constant value 3

6.3.3 EvaluationLog

Introduction

This XML structure is exported by reading the ***Evaluation logbook*** leaf.

EvaluationLog XML structure

Tag name	Description
EvaluationLog	Root item.
Item	Item containing evaluation logbook event.

6.3.4 Run logbook

Introduction

This XML structure is exported by reading the ***Run logbook*** leaf.

Logbook XML structure

Tag name	Description
Logbook	Root item
Item	A logbook item
AccumulatedTime	Accumulated time in seconds
AccumulatedVolume	Accumulated volume, including unit
Event	Logbook event
EventType	Event type
EventSubType	Subtype of event

6.3.5 SignatureList

Introduction

This XML structure is exported by reading the ***Method Signatures*** and ***Result Signatures*** leaves.

SignatureList XML structure

Tag name	Description
SignatureList	Root item
Signature	A signature
Username	Name of user
Fullname	Full name of user
Position	Position of user
Date	Date, formatted in server local time
Meaning	Signature meaning
Locked	If present, the file is locked against further change

6.3.6 SnapshotList

Introduction

This XML structure is exported by reading the ***Snapshot*** leaf.

Note: *To get the actual y-axis value, use the time or volume value as a reference into the curves.*

SnapshotList XML structure

Tag name	Description
SnapshotList	Root item
Chromatogram	A signature
Name	Name of chromatogram
Snapshot	Run number
TimeRetention	Time retention
VolumeRetention	Volume retention
Curve	Curve
Name	Curve name
Unit	Curve unit
Time Value	Time stamp of snapshot
Volume Value	Volume stamp of snapshot

6.3.7 UsedComponents

Introduction

This XML structure is exported by reading the ***Result Strategy Used Components*** leaf.

UsedComponents XML structure

Tag name	Description
UsedComponents	Root item.
Component	Component name used during run.

6.3.8 Notes

Introduction

This XML structure is exported by reading the ***Method Notes***, ***Start Notes***, ***Run Notes***, and ***Evaluation Notes*** leaves.

Notes XML structure

Tag name	Description
Notes	Root item, the note follows.

6.3.9 ScoutingList

Introduction

This XML structure is exported by reading the ***Scouting*** leaf.

Note: *To get the actual y-axis value, use the time or volume value as a reference into the curves.*

ScoutingList XML structure

Tag name	Description
ScoutingList	Root item.
Scouting	A scouting run.
TotalNumberOfScoutings	Total number of scouting items available.
RunScouting	If available, this item indicates the scouting run number of this result file.
Run	Run number.
ThisScoutingWasUsedDuringRun	This leaf is available if this scouting run was used when the method was run. The run number is the same as ***RunScouting***.
Variable	Variables defined in this run.
Block	Block name of this variable.
Name	Name of variable.
Unit	Unit, only included if variable has a unit.
Value	Value of variable.

6.3.10 SettingsList

Introduction

This XML structure is exported by reading the ***System Settings*** leaf.

SettingsList XML structure

Tag name	Description
SettingsList	Root item.
Group	A group.
GroupName	The name of the group.
Instruction	Instruction names. There is at least one instruction per group.

6.3.11 TextInstructions

Introduction

This XML structure is exported by reading the ***Text Instructions*** leaf.

TextInstructions XML structure

Tag name	Description
TextMethod	Root item, the complete text method follows. Each row is separated by a carriage return and a new line.

6.3.12 VariableList

Introduction

This XML structure is exported by reading the ***Variables*** leaf.

Start protocol variables XML structure

Tag name	Description
VariableList	Root item.
Variable	A variable.
Block	Block name of the variable.
Name	The variable name.
VisibleInScouting	Indicates if the variable is visible in scouting.
VisibleInDetails	Indicates if the variable is visible in details only.
Value	Value. Can be a number or a string, depending on the variable.
Unit	Unit of the value. Not included if no unit is present.

6.3.13 QuestionList

Introduction

This XML structure is exported by reading the ***Questions*** leaf.

Start protocol questions XML structure

Tag name	Description
QuestionList	Root item
Item	A question item
Question	The question
Answer	The answer

6.3.14 PeakTable

Introduction

This XML structure is exported by reading leaves under the ***PEAK TABLES*** branch.

PeakTable XML structure

Tag name	Description
PeakTable	Root item
Summary	Summary of the peak table
TotalNumberOfDetected-Peaks	Total number of peaks
TotalArea	Total area
Unit	Unit used.
AreaInEvaluatedPeaks	Area in evaluated peaks
RationPeakareaTotalarea	Ratio peak area/total area
TotalPeakDuration	Total peak duration
ColumnHeight	Column height
ColumnV0	Column V_0
ColumnVt	Column V_t
SourceCurve	Source curve name
Baseline	Baseline
Rejection	Rejection
MinHeight	Minimum height
MinWidth	Minimum width
MaxWidth	Maximum width
MinArea	Minimum area
MaxNumberOfPeaks	Maximum number of peaks
Quantitation	Quantitation (molecular size, standard addition, internal standard, external standard, recovery)
Peak	A peak

Tag name	Description
No	Peak number
Name	Peak name
Retention	Retention
Start	Start
End	End
Width	Width
Area	Area
AreaPerTotalArea	Area/total area
AreaPerPeakArea	Area/peak area
Height	Height
WidthAtHalfHeight	Width at half peak height
HeightAtStart	Height at start
HeightAtEnd	Height at end
BaselineAtStart	Baseline at start
BaselineAtMax	Baseline at max
BaselineAtEnd	Baseline at end
ConductivityHeightStart	Conductivity height at start
ConductivityHeightMax	Conductivity height at maximum
ConductivityHeightEnd	Conductivity height at end
FractionTubeAtStart	Fraction tube at start
FractionTubeAtMax	Fraction tube at maximum
FractionTubeAtEnd	Fraction tube at end
PeakStartType	Peak start type (dropline, skim, baseline, skim-drop, unknown)
PeakEndType	Peak end type (dropline, skim, baseline, skim-drop, unknown)
Sigma	Sigma
Resolution	Resolution
CapacityFactor	Capacity factor

Tag name	Description
Kav	K_{av}
PlateHeight	Plate height
PlatesPerMeter	Plates per meter
AsymmetryStart	Asymmetry start
AsymmetryEnd	Asymmetry end
Asymmetry	Asymmetry
AverageConductivity	Average conductivity
ExtinctionCoefficient	Extinction coefficient
ExtCoeffConcentration	Concentration calculated using extinction coefficient.
ExtCoeffAmount	Amount calculated using extinction coefficient.
Concentration	Concentration
Amount	Amount
MolecularSize	Molecular size
Type	Type

6.3.15 Unicorn raw

Introduction

This XML structure is exported by reading the attribute ***HDA_UNICORN_RAW_DATA*** for a ***Curve*** leaf.

UnicornRawData XML structure

Tag name	Description
UnicornRawData	Root item.
Name	Name of curve.
Type	Curve type. Does not contain the bit mask, which the ***HDA_CURVE_TYPE*** does, only the actual type.
Unit	Curve unit (y-axis).
TimeUnit	Unit of time curve (x-axis).
VolumeUnit	Unit of volume curve (x-axis).
CurveMin	Minimum value of y-axis.
CurveMax	Maximum value of y-axis.
ZeroAdjust	Present if zero adjust is enabled.
ZeroAdjustTime	Zero adjust time.
ZeroAdjustVolume	Zero adjust volume.
StartTimeRetention	Retention start time.
InjectionTimeRetention	Injection start time.
NumberOfTimeDataPoints	Number of curve time data points.
CTD	Curve Time Data. This is repeated for all time data points. The values are assigned as attributes to the tag (x and y attributes).
StartVolumeRetention	Retention start volume.
InjectionVolumeRetention	Injection start volume.

Tag name	Description
NumberOfVolumeDataPoints	Number of curve volume data points.
ColumnVolumeDefined	Present if column volume is defined.
ColumnVolume	Column volume value for curve.
ColumnVolumeUnit	Unit of ***ColumnVolume*** curve (X-axis).
CVD	Curve Volume Data. This is repeated for all volume data points. The values are assigned as attributes to the tag (***x***, ***y*** and, if available, ***cv***).
CMD	Curve Markers Data. This is repeated for all marker (text) data points. The values are assigned as attributes to the tag (***t*** = time, ***v*** = volume, ***t1*** = text1 and, if available, ***t2*** = text2).

6.3.16 Columns

Introduction

This XML structure is exported by reading the ***Columns*** leaf.

Columns XML structure

Tag name	Description
Columns	Root item.
Column	A column.
Name	Name of column.
Height	Height of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Diameter	Diameter of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Volume	Volume of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
VolumeUnit	Height of column. Attribute mandatory defines if item is mandatory.
Technique	Technique used by column. Attribute mandatory defines if item is mandatory.
Vt	Total volume of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Vo	Empty volume of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
MaxPressure	Maximum pressure of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
DefaultFlowrate	Default flow rate of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
MaxFlowrate	Maximum flow rate of column. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
TypPeakWidthAtBase	Typical peak width at base. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
pHLongMax	Long-term maximum pH. Attribute mandatory defines if item is mandatory.

Tag name	Description
pHLongMin	Long-term minimum pH. Attribute mandatory defines if item is mandatory.
pHShortMax	Short-term maximum pH. Attribute mandatory defines if item is mandatory.
pHShortMin	Short-term minimum pH. Attribute mandatory defines if item is mandatory.
AverageParticle-Diameter	Average particle diameter. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.
Code	Code of column. Attribute mandatory defines if item is mandatory.
TypicalLoading-Range	Typical loading range. Attribute mandatory defines if item is mandatory.
MolWeigh-tRange	Molecular weight range. Attribute mandatory defines if item is mandatory. Attribute unit defines unit of value.

6.3.17 EvaluationProcedures

Introduction

This XML structure is exported by reading the ***Evaluation procedure*** leaf.

EvaluationProcedures XML structure

Tagname	Description
EvaluationProcedures	Root item.
EvaluationProcedure	An evaluation procedure.
Name	Name of evaluation procedure.
WasRun	Indicates if the evaluation procedure was run during method run.

6.3.18 FracXY

Introduction

This XML structure is exported by reading the ***FracXY*** leaf.

FracXY XML structure

Tagname	Description
FracXY	Root item
FractionOrder	Fraction order (Serpentine row, Row-by-row, Serpentine column or Column-bycolumn)
Name	Name of rack
Group	Rack group
Name	Name of rack group
LastTube	Last tube of group

6.3.19 AuditTrail

Introduction

This XML structure is exported by reading a ***Global Logbook*** or ***System Logbook*** leaf.

AuditTrail XML structure

Tag name	Description
AuditTrail	Root item.
Type	Type of audit trail: ***Global***, ***System*** or ***Backup***. ***Backup*** is used when accessing a direct audit trail file address.
CheckSumError	If this item is available, the file has been modified outside UNICORN or is corrupt.
Audit	An audit item.
Time	Time of audit.
TimeZone	Time zone.
Message	Audit message.
Details	Log entry details.

6.3.20 Method/Result Instrument Configuration

Introduction

This XML structure is exported by reading a ***Method Instrument Configuration*** or ***Result Instrument Configuration*** leaf.

Method/Result *Instrument Configuration* XML structure

Tag name	Description
InstrumentConfiguration	Root item
Name	Name of instrument configuration
Version	Instrument configuration version
Help text	Instrument configuration help text
StrategyName	The instrument configuration name
StrategyNotes	The instrument configuration notes
PhaseConfiguration	Phase configuration name and version
TemplateMethodConfiguration	Protocol configuration name and version
PerformanceTestConfiguration	Performance test configuration name and version
FlowSchemeConfiguration	Flow scheme configuration name and version
VIDConfiguration	VID configuration name and version
FirmwareConfiguration	Firmware configuration name and version
CompatibleUNICORNClientVersion	UNICORN version
CompatibleUNICORNInstrumentServerVersion	UNICORN version

6.3.21 System

Introduction

This XML structure is exported by reading a ***System*** leaf.

System XML structure

Tag name	Description
SystemName	Name of system.
SystemType	Type of system.
InstrumentConfiguration	Instrument configuration name and version.
SystemIsNonActive/SystemIsActive	Active system.
ControlledByUNICORNInstrumentServer	Instrument server computer name. ***Note:*** *Displayed only if the system is Active.*
InstrumentSerialNo	Instrument serial number.
ConnectionBasedOnSerial/ ConnectionBasedOnIP	Type of connection based on serial number or instrument IP address.
InstrumentIPAddress	Instrument IP address.

Page intentionally left blank

Give feedback on this document

Visit cytiva.com/techdocfeedback or scan the QR code.

cytiva.com

For local office contact information, visit cytiva.com/contact

29752660 AA V:1 05/2024