Prosys OPC UA Client – User Manual

Version: 2.3.2

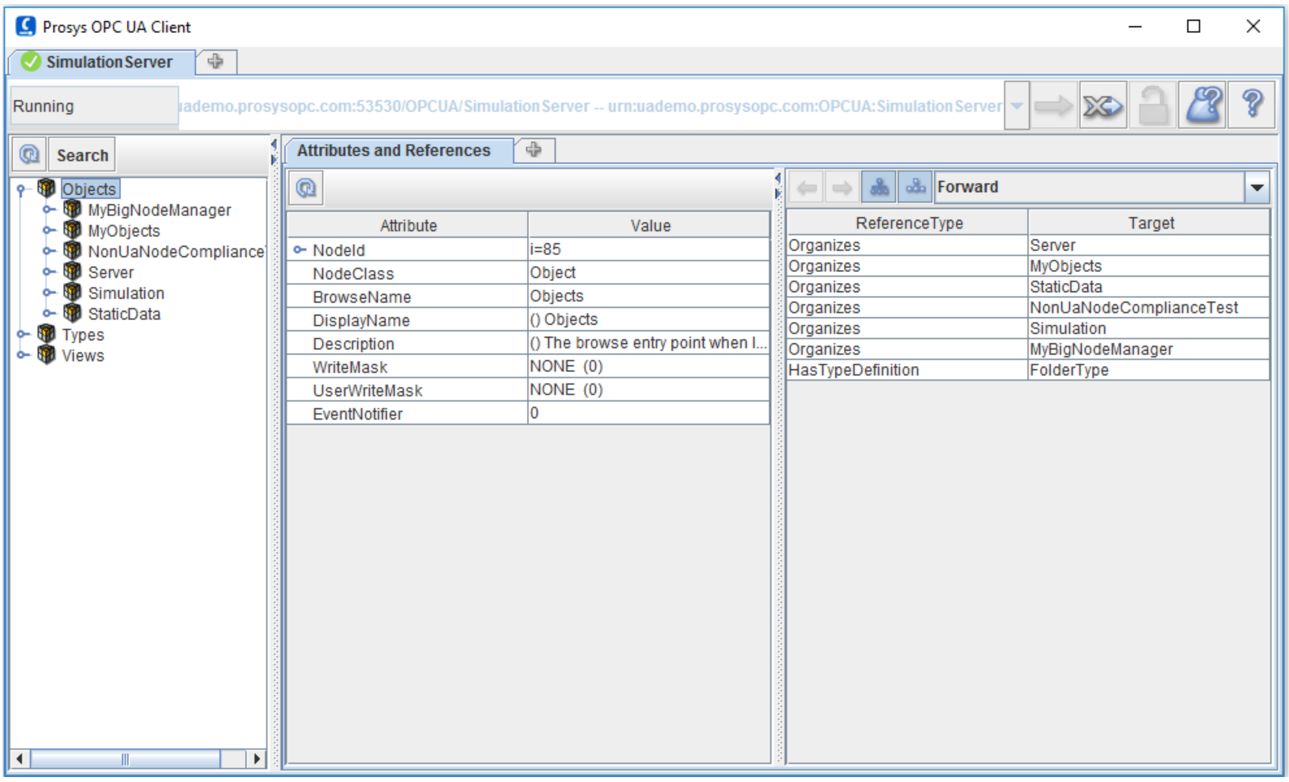
Contents

Prosys OPC UA Client – User Manual

Prosys OPC UA Client is a generic purpose OPC UA client. It implements the three OPC UA information models:

1. Data Access
2. Historical Access
3. Alarms & Conditions

With the OPC UA Client you can have multiple secure server connections and manage them using tabbed pages. Data of each server is represented using a browser tree on the left side of the GUI. The GUI includes tabbed pages called Attributes and References to explore node attributes and references, Data View tabbed page for monitoring nodes and writing values to them, History View to explore historical data of nodes and create charts of it, and Events View to monitor events and alarms sent by servers and Event History View for requesting history of events from the servers. Figure 1 illustrates the overview of the OPC UA Client.



Attribute	Value
NodeId	i=85
NodeClass	Object
BrowseName	Objects
DisplayName	() Objects
Description	() The browse entry point when l...
WriteMask	NONE (0)
UserWriteMask	NONE (0)
EventNotifier	0

ReferenceType	Target
Organizes	Server
Organizes	MyObjects
Organizes	StaticData
Organizes	NonUaNodeComplianceTest
Organizes	Simulation
Organizes	MyBigNodeManager
HasTypeDefinition	FolderType

Figure 1: OPC UA Client Overview

1 OPC UA Servers

There are a couple of different test server options provided by Prosys OPC. You can download Prosys OPC UA Simulation Server in www.prosysopc.com and run it locally, or connect to a remote server running in our premises. If you run the OPC UA Java Server locally, you can connect to it using this URL:

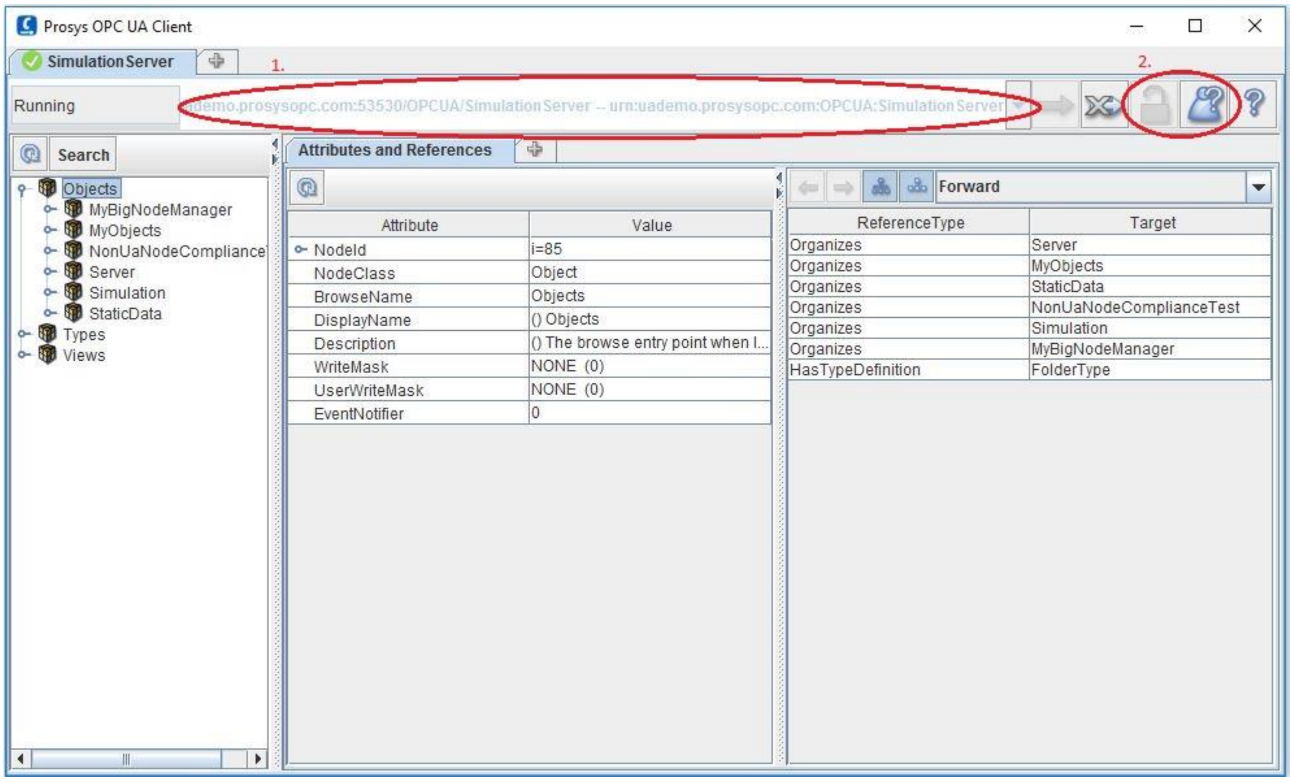
opc.tcp://localhost:53530/OPCUA/SimulationServer

If you would like to connect to a remote server, use this URL:

opc.tcp://uademo.prosysopc.com:53530/OPCUA/SimulationServer

2 Connecting to Server

Connecting to an OPC UA server is based on the server URL that can be typed to the address bar, which is located in the toolbar below the server tabs (see Figure 2). The address bar remembers previous URL:s and it also acts as a dropdown menu so that it is possible to select the remembered URL:s. On the right side of the address bar there are push buttons for connecting and disconnecting. If the client cannot connect to the server, the GUI launches an error dialog or proposes discovery by opening a discovery dialog box. If discovery is available, the desired server can then be selected in the dialog, clicking Select and then clicking Connect.



Running

uademo.prosysopc.com:53530/OPCUA/SimulationServer -- urn:uademo.prosysopc.com:OPCUA:SimulationServer

Attribute	Value
NodeId	i=85
NodeClass	Object
BrowseName	Objects
DisplayName	() Objects
Description	() The browse entry point when l...
WriteMask	NONE (0)
UserWriteMask	NONE (0)
EventNotifier	0

ReferenceType	Target
Organizes	Server
Organizes	MyObjects
Organizes	StaticData
Organizes	NonUaNodeComplianceTest
Organizes	Simulation
Organizes	MyBigNodeManager
HasTypeDefinition	FolderType

Figure 2: Server Address Bar (1.) and Security Settings & UserID Settings (2.)

2.1 Security Options

Security options can be set before connecting to a server by clicking the padlock icon below server tabs. When connecting, the client application sends an OpenSecureChannel request secured in accordance to Security Mode and Security Policy to the selected Session Endpoint of the server. There are three different Security Modes available:

1. None
2. Sign
3. Sign & Encrypt

If the selected Security Mode is "None", the OpenSecureChannel request is not secured at all. If "Sign" is selected, the message is signed with the associated Private Key of the Application Instance Certificate of the client application. Signing the message allows detecting if it has been manipulated by an untrusted third party. If the selected Security Mode is "Sign & Encrypt", the message is also encrypted with the Public Key of the server's Application Instance Certificate. Encryption makes it hard for third parties to read the content of the messages.

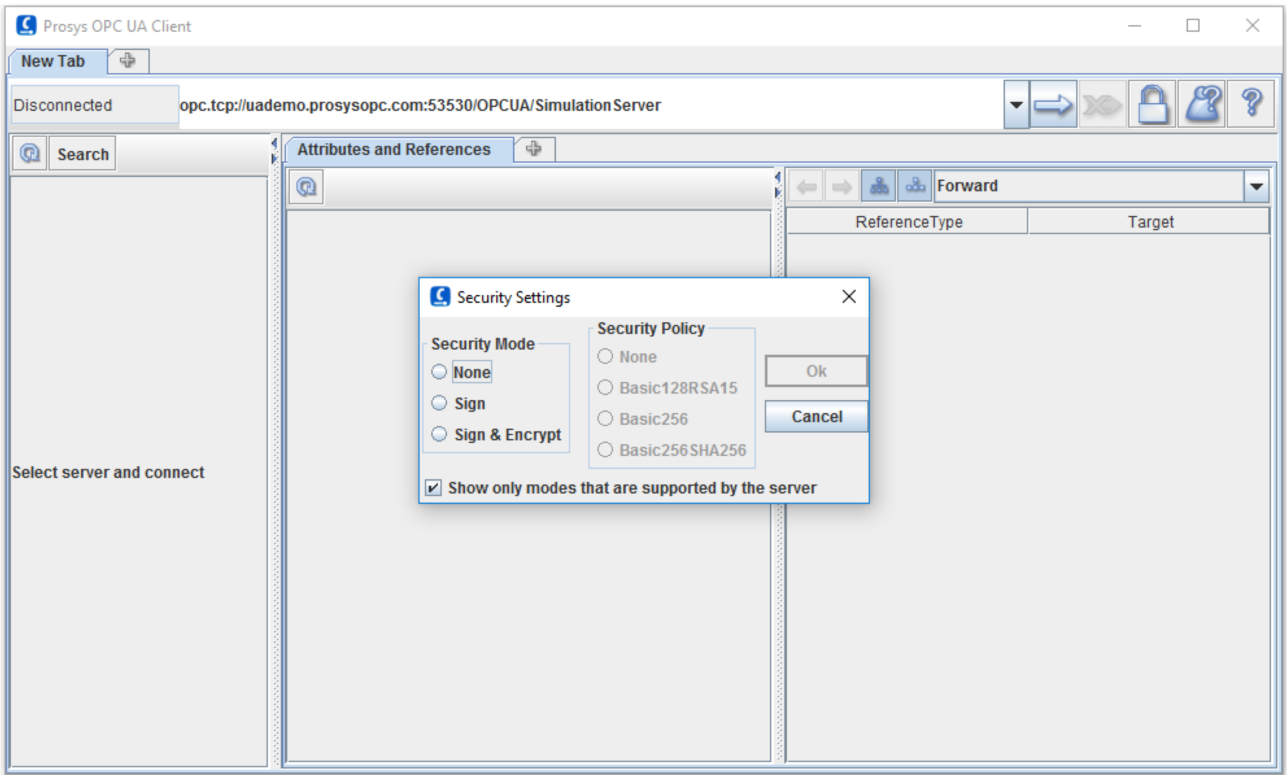


Figure 3: Security Settings

Security policy determines the algorithm for signing and encrypting. There are multiple security policies available (see Figure 3):

1. None
2. Basic128RSA15
3. Basic256
4. Basic256SHA256

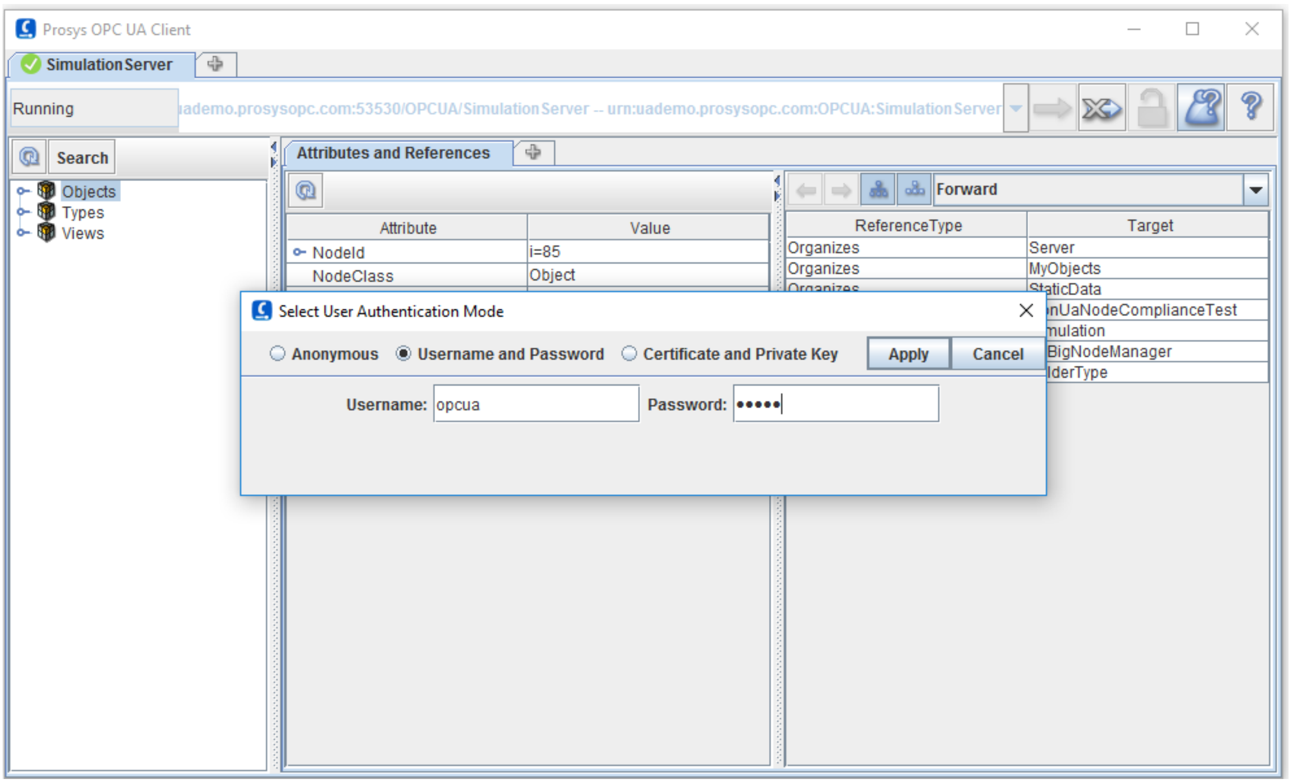
If "None" is selected, a suite of algorithms that does not provide any security settings is used. If "Basic128RSA15" is selected, a suite of algorithms that uses RSA15 as Key-Wrap-algorithm and 128-Bit for encryption algorithms is used. If "Basic256" is selected, a suite of algorithms that uses 256-Bit for encryption algorithms is used.

UserID options can be set before or after connecting to provide proper authentication settings (see Figure 4). The user authentication process is performed during the session establishment with the server. There are three options available:

1. Anonymous
2. Username and Password
3. Certificate and Private Key

If "Anonymous" is selected, user information is not made available. If "Username and Password" is selected, a user is identified by user name and password. The user is asked to type the user name and password in separate text fields in the GUI. If "Certificate and Private Key" is selected, the user is identified by an X509v3 Certificate File and its Private Key. The certificate file and private key can be searched in the hard disc using a file explorer provided by the GUI.

The user authentication setting can be selected by clicking the tool button in the upper right corner of the GUI (see Figure 2). It can be applied also when the connection is established to impersonate the session to a new user account.



Running

uademo.prosysopc.com:53530/OPCUA/SimulationServer -- urn:uademo.prosysopc.com:OPCUA:SimulationServer

Attribute	Value
NodeId	i=85
NodeClass	Object

ReferenceType	Target
Organizes	Server
Organizes	MyObjects
Organizes	StaticData
	nUaNodeComplianceTest
	mulation
	BigNodeManager
	lderType

Select User Authentication Mode

Anonymous [✓ Username and Password] Certificate and Private Key — Apply Cancel

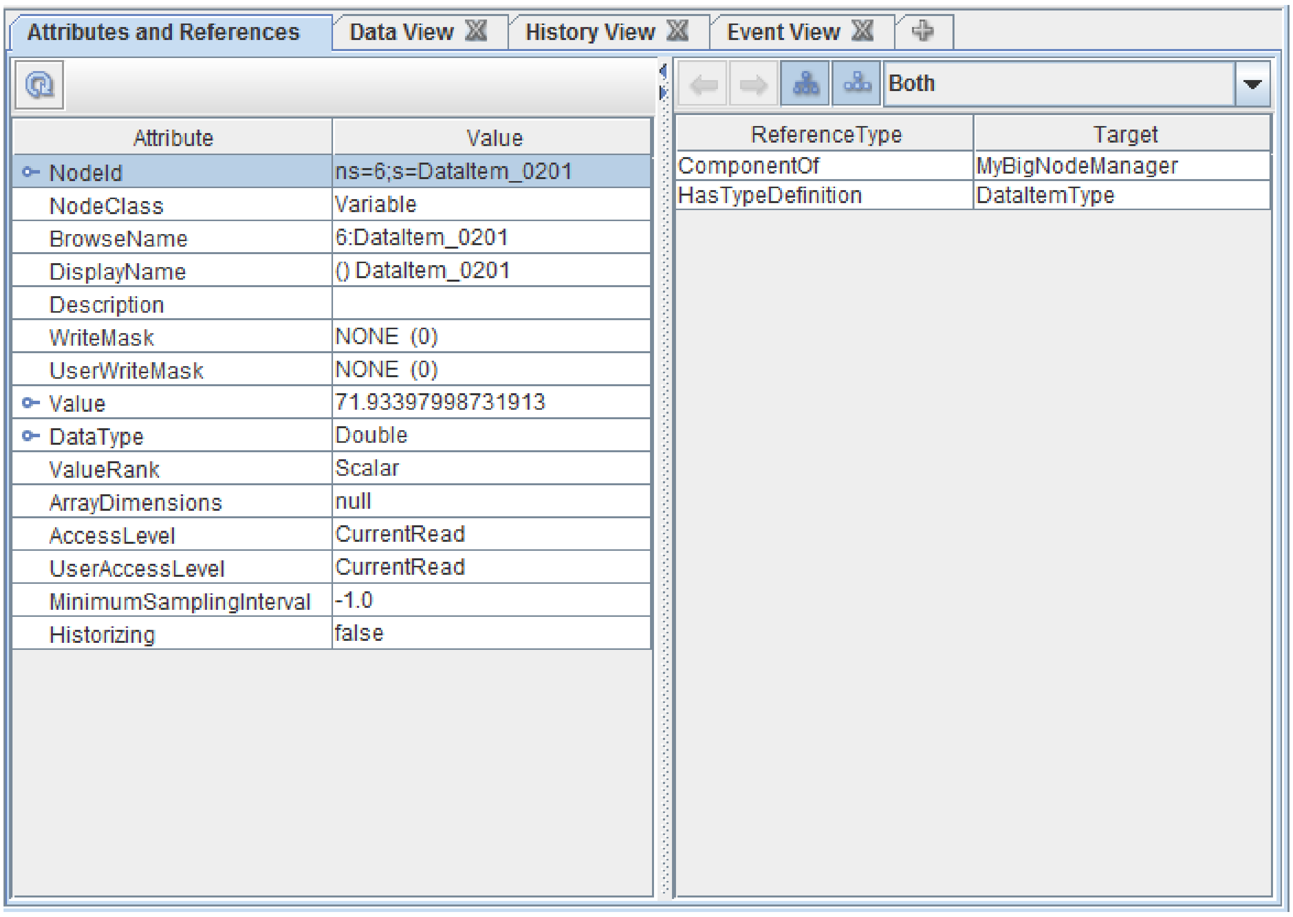
Username: opcua Password: •••••

Figure 4. Defining the user authentication mode and user account. You can change the user account after connection as well. The Simulation Server enables you to define test user accounts.

When the server URL, Security Options and UserID settings are set, the connection to the server can be established by clicking *Connect to Server*. If it is not possible to connect to the server but discovery works, the discovery dialog opens. The desired server can then be selected in the dialog.

3 Attributes and References View

The Attributes and References view consists of two tables: one to represent node attributes and another one to represent the node references. Both tables show the data from the node that is selected in the server address space browser tree. The attribute table shows the attributes of the node and their respective values. For example, in Figure 5 the NodeClass attribute of the selected node is Variable.



Attribute	Value
NodeId	ns=6;s=DataItem_0201
NodeClass	Variable
BrowseName	6:DataItem_0201
DisplayName	() DataItem_0201
Description	
WriteMask	NONE (0)
UserWriteMask	NONE (0)
Value	71.93397998731913
DataType	Double
ValueRank	Scalar
ArrayDimensions	null
AccessLevel	CurrentRead
UserAccessLevel	CurrentRead
MinimumSamplingInterval	-1.0
Historizing	false

ReferenceType	Target
ComponentOf	MyBigNodeManager
HasTypeDefinition	DataItemType

Figure 5: Attributes and References View

Reference table also has two columns: Reference and Target. Reference column represents the type of the reference, and Target column represents the display name of the node that the reference points to. For example, in Figure 5 the selected node, DataItem_0201, has two references: the first reference's type is ComponentOf, and it is targeting to MyBigNodeManager node. It means that the selected node is a child component of MyBigNodeManager node. Accordingly, MyBigNodeManager has a HasComponent reference to DataItem_0201.

The Reference Table can be filtered with the Controls on top of the table. With the buttons in the middle, you can decide whether to display Hierarchical References and/or Non-Hierarchical References in the reference table. Hierarchical references are used to model node hierarchies, such as the HasComponent relationship, and non-hierarchical references are used for other semantical relations, for example for the HasTypeDefinition, which defines the type of the object and variable nodes, referring to the respective type node in the types hierarchy. The dropdown menu includes three options:

1. Forward
2. Inverse
3. Both

Forward shows only forward references, Inverse shows only backwards references, and Both shows both directions of references. Direction of a reference is especially important for non-symmetric references: for example, a reference can be either type "has-parent" or "is-sibling-of" depending on the direction of the reference. For symmetric references direction is irrelevant: the reference type is the same for both directions.

In Figure 5 both inverse and forward, and hierarchical and non-hierarchical references are displayed. ComponentOf is hierarchical and inverse reference: DataItem_0201 is a child component of MyBigNodeManager which represents hierarchical model, and it is inverse reference because the reference is targeting to parent node. HasTypeDefinition is a non-hierarchical, forward reference. It points to the type definition of DataItem_0201 variable. Complete reference information of each reference can be seen in a tooltip.

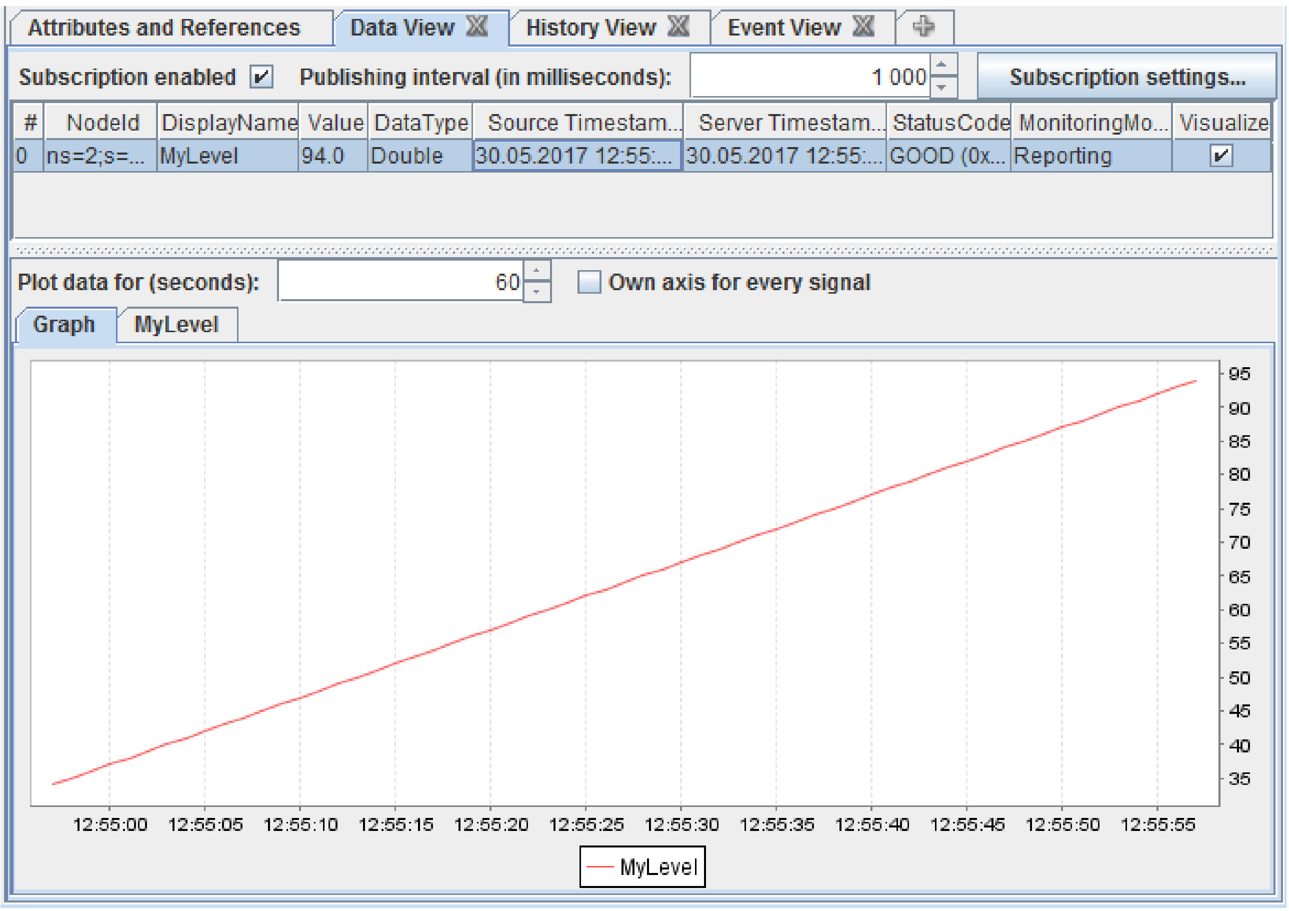
3.1 Browsing via References

It is possible to navigate from node to node also from the references view. If a reference is double-clicked, the selected node in browser tree changes to the node that the double-clicked reference points to. When the node in browser tree changes attributes and references tables also changes to corresponding node. Above attributes table there are *Back* and *Forward* pushbuttons to move back and forwards to previous nodes.

4 Data View

In the Data View it is possible to add nodes from the Address Space Browser to the monitored item list. The purpose of the monitored item list is that you can pick up the interesting nodes from the server address space and monitor their properties and value changes. It is also possible to write a new value by right clicking the item in the list and selecting Write in the context menu. The new value can then be typed in the dialog that opens. If the typed value is not of the correct type, the GUI informs you about the type mismatch and asks her to type it again. In the same context menu, one can also change the monitored item settings. Subscription settings can be modified with the button in the upper right corner (see Figure 6).

Nodes can be dragged and dropped from the Address Space Browser to the monitored item list, either one or several at the time by selecting them in the tree with shift-click or control-click. The “Monitor” –button can also be used to copy the selected nodes from the tree to the monitored item list. Data View is illustrated in Figure 6.



Subscription enabled ✓ Publishing interval (in milliseconds): 1 000

#	NodeId	DisplayName	Value	DataType	Source Timestam...	Server Timestam...	StatusCode	MonitoringMo...	Visualize
0	ns=2;s=...	MyLevel	94.0	Double	30.05.2017 12:55:...	30.05.2017 12:55:...	GOOD (0x...	Reporting	✓

Plot data for (seconds): 60 Own axis for every signal

Figure 6: Data View

The following node information is displayed in the monitored item list:

1. NodeId
2. DisplayName
3. Value
4. DataType
5. Source Timestamp
6. Server Timestamp
7. StatusCode
8. MonitoringMode

Node ID uniquely identifies a node in the server. It is used to address the Node in the OPC UA Services. Display Name is a common node attribute that contains the name of the Node. Other naming attributes exist, e.g. browse name, but DisplayName attribute is recommended when the name is displayed. Value field is simply the value of the node, and DataType represents the data type of the value. Source Timestamp and Server Timestamp are the two basic timestamps of OPC UA. Source Timestamp is used to reflect the timestamp that was applied to a variable value by the data source. It should indicate the last change of the value or status code, and it should always be generated by the same physical clock. Server timestamp is used to reflect the time that the server received a variable value or knew it to be accurate in case the changes are reported by exception and the data source connection is operating. StatusCode is a success code that indicates the read operation or quality of the read value. MonitoringMode represents whether sampling of the node and reporting of notifications are enabled.

The checkbox in the last column is used to select if the user wants to plot the value changes of a node. When visualizing is enabled, the values are shown in a graph below the monitored items list. The values are also shown with status code and time stamp in a table in another tab. The user can select for how many seconds the values are shown and choose if each plotted signal has an own value axis.

5 History View

The purpose of the History View is to represent historical data of Variables and thus to provide OPC UA History Access functionality. The History View consists of a table for adding monitored nodes and a set of tools to define the time axis and chart view. The layout is illustrated in Figure 7.

Variable Nodes that have history data, i.e. Nodes that have HistoryReadable in the AccessLevel and UserAccessLevel, can be dragged and dropped from the Address Space Browser into the table at the upper left corner of the view (under the “Drag Nodes Here” title). Several items can be moved into the table at once by selecting them in the tree and dragging and dropping When nodes have been selected, it is possible to create the history chart by clicking Update-button.

The History Mode Selection includes tools for configuring the time axis. If *Last* is selected from the first drop down selector, the time axis is determined by the length of the observation period. The edit box and the time unit selector define the length of the history period to draw. You can choose from the following time units:

- Milliseconds
- Seconds
- Minutes
- Hours
- Days

On the second row there is a date and time field for setting the end of the history period. You can also check the Now check box, which fixes the end time to the current moment. In Figure 7 the time axis has been set to be 3 minutes, starting 3 minutes before the end time, 30/05/17 13:32:45.

The second mode, *From*, allows you to define manually both the start and end of the time axis. There are two text fields for setting them. You can also select *Now* as the end time, which fixes the end time to the current moment like in the “Last” mode. *From* mode is illustrated in Figure 8.

When you click the *Read* button, the history graph appears in the graph view. There are two tabbed pages in the view: Graph and a table representing all the history values (e.g. MyLevel tab in Figures 7 and 8).

There is also an option to read aggregated values. The aggregate functions that the server supports can be used to process the requested data. For example, in Figure 8 the minimum values of MyLevel with interval of 5 seconds are requested.

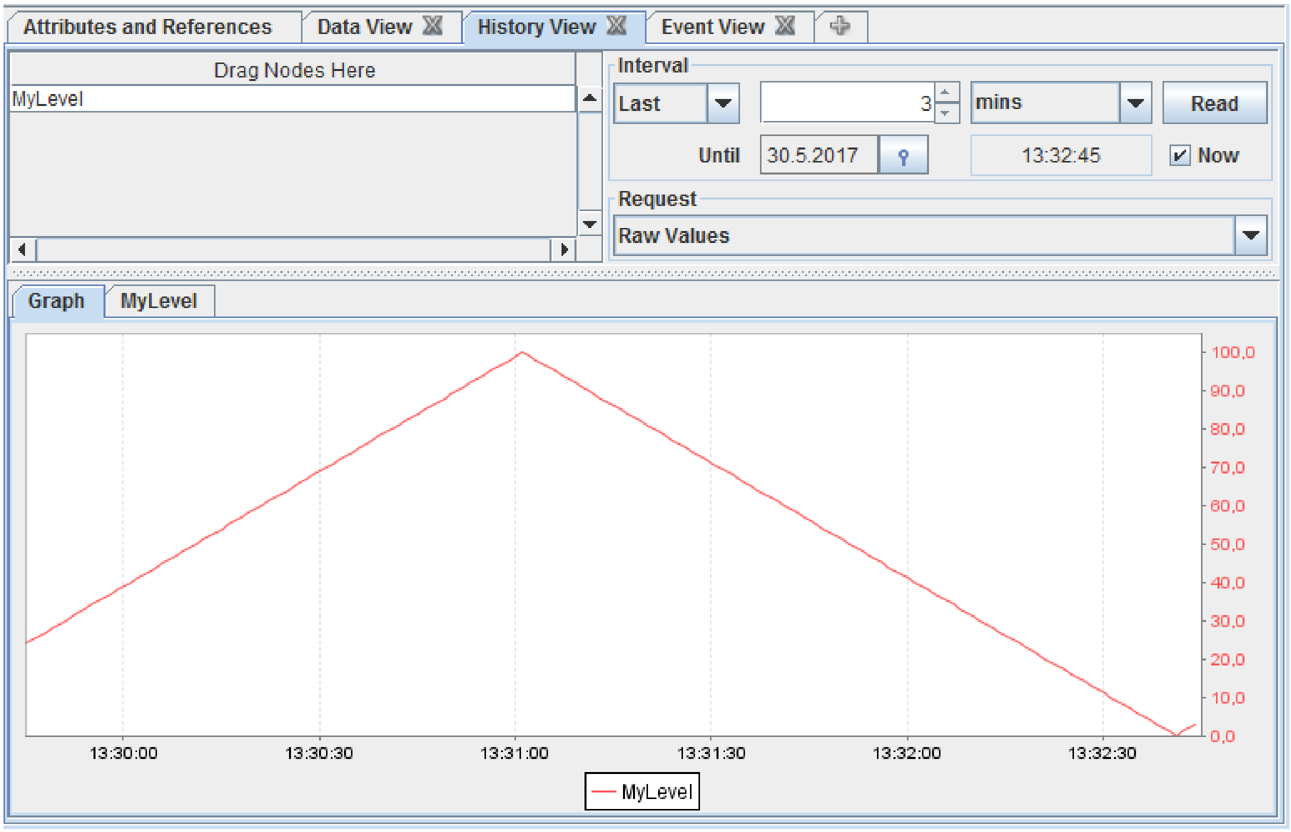


Figure 7: History View, Fixed Time Period Selected

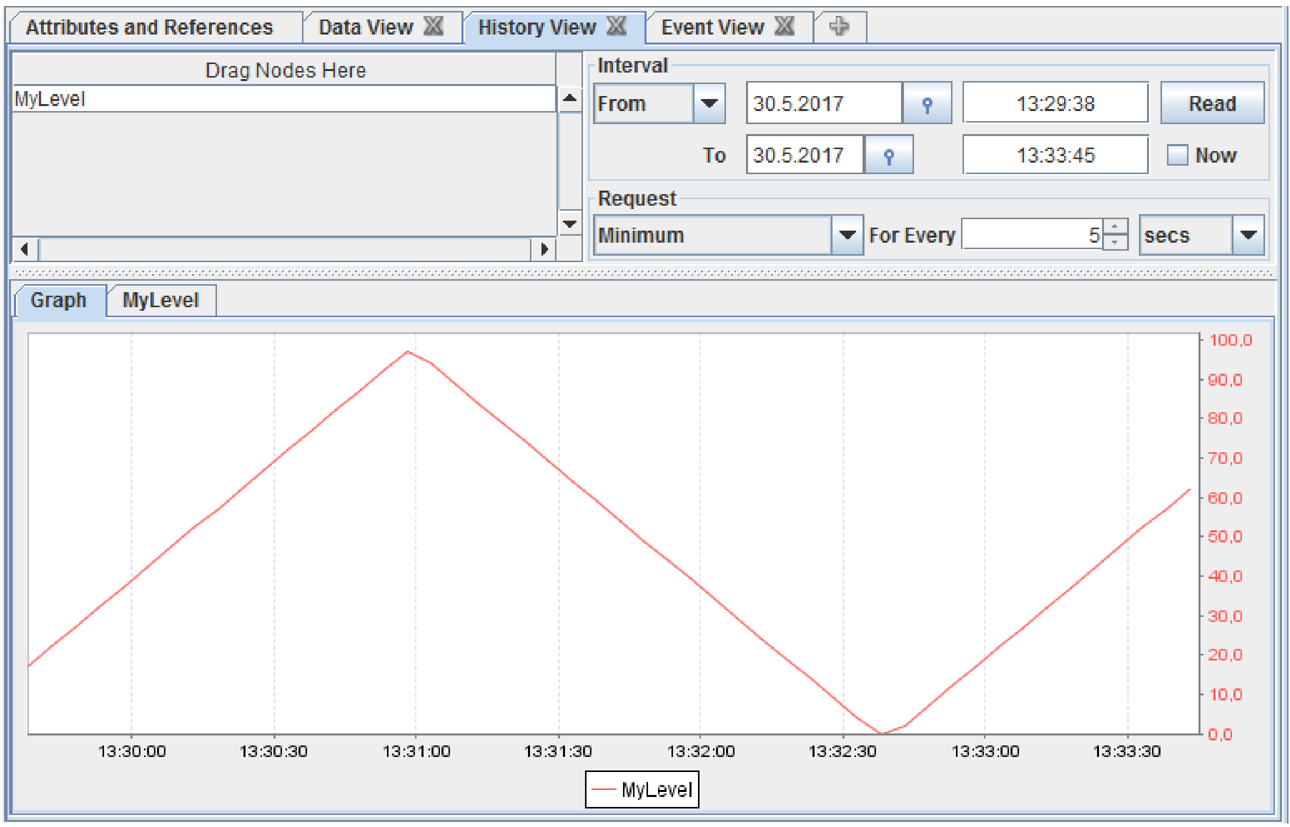
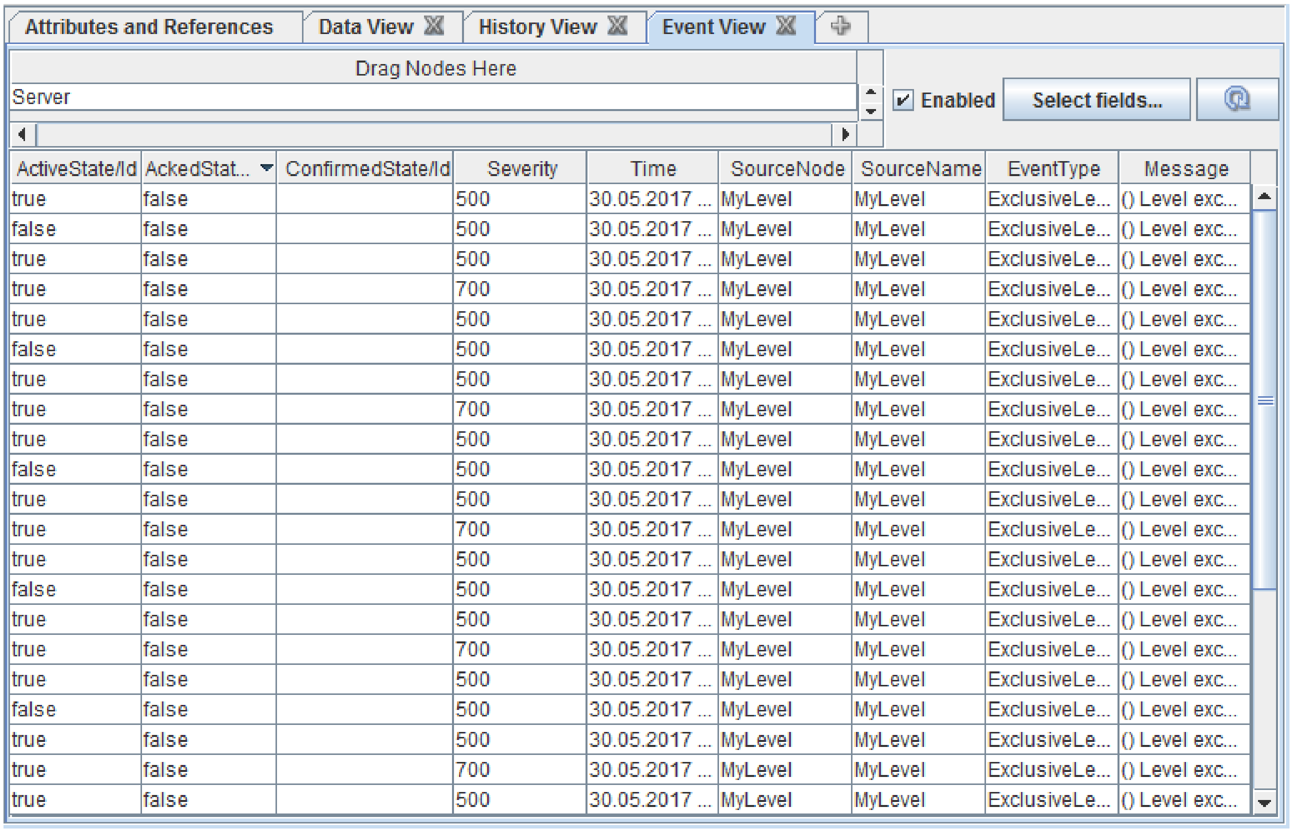


Figure 8: History View, Manual Time Period Selected

6 Event View

The Event view is designed for receiving Event and Alarm notifications from the server. The main component in the Event View is a table in which the received events are listed. In addition to the table there are buttons for changing the monitored node and changing the requested event fields. The layout of the Event view is illustrated in Figure 9.



ActiveState/Id	AckedStat...	ConfirmedState/Id	Severity	Time	SourceNode	SourceName	EventType	Message
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
false	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		700	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
false	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		700	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
false	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		700	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
false	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		700	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
false	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		700	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...
true	false		500	30.05.2017 ...	MyLevel	MyLevel	ExclusiveLe...	() Level exc...

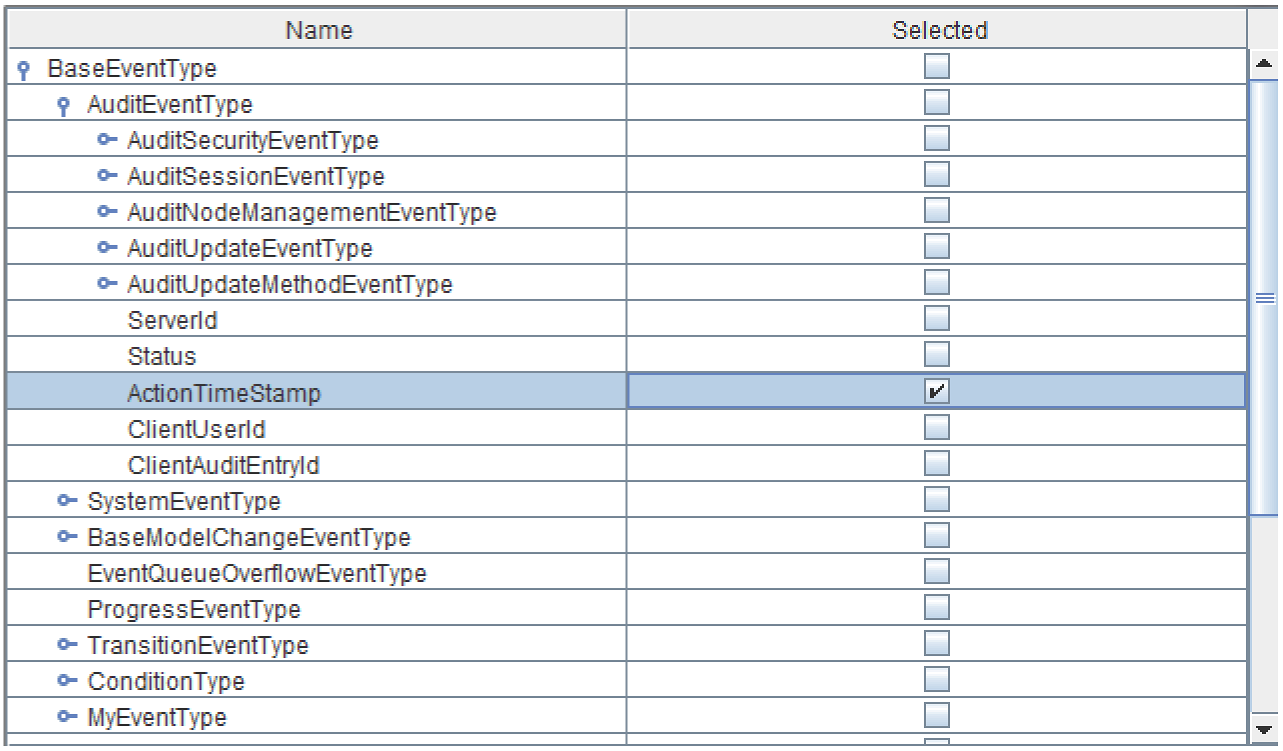
Figure 9: Event View

You can select which event columns are displayed and requested from the server. In Figure 9 nine fields are shown. ActiveState, AckedState and ConfirmedState present whether the event or alarm is active, acknowledged and confirmed, respectively. Severity shows the severity of each event. Servers can use different severity values for an event, varying from 0 to 1000 to indicate the importance of reacting to an event. Table 1 represents a mapping of five different severity levels. Time is the event timestamp. SourceNode and SourceName identify the node that sent each event, identified by the NodeId and the BrowseName of the respective node. EventType shows the ID of the event type. The last field, Message, shows the message associated with the event.

Table 1: Five severity levels

Client Severity	OPC Severity
HIGH	801-1000
MEDIUM HIGH	601-800
MEDIUM	401-600
MEDIUM LOW	201-400
LOW	1-200

You can edit the fields that are represented in the table by clicking “Select fields…” button. When the button is clicked a dialog illustrated in Figure 10 opens. You can then select the desired fields in the list of all the available event fields. The events usually include a lot of information and not all of the data is necessary to be shown. That is why it is important for you to be able to select which data is being shown.

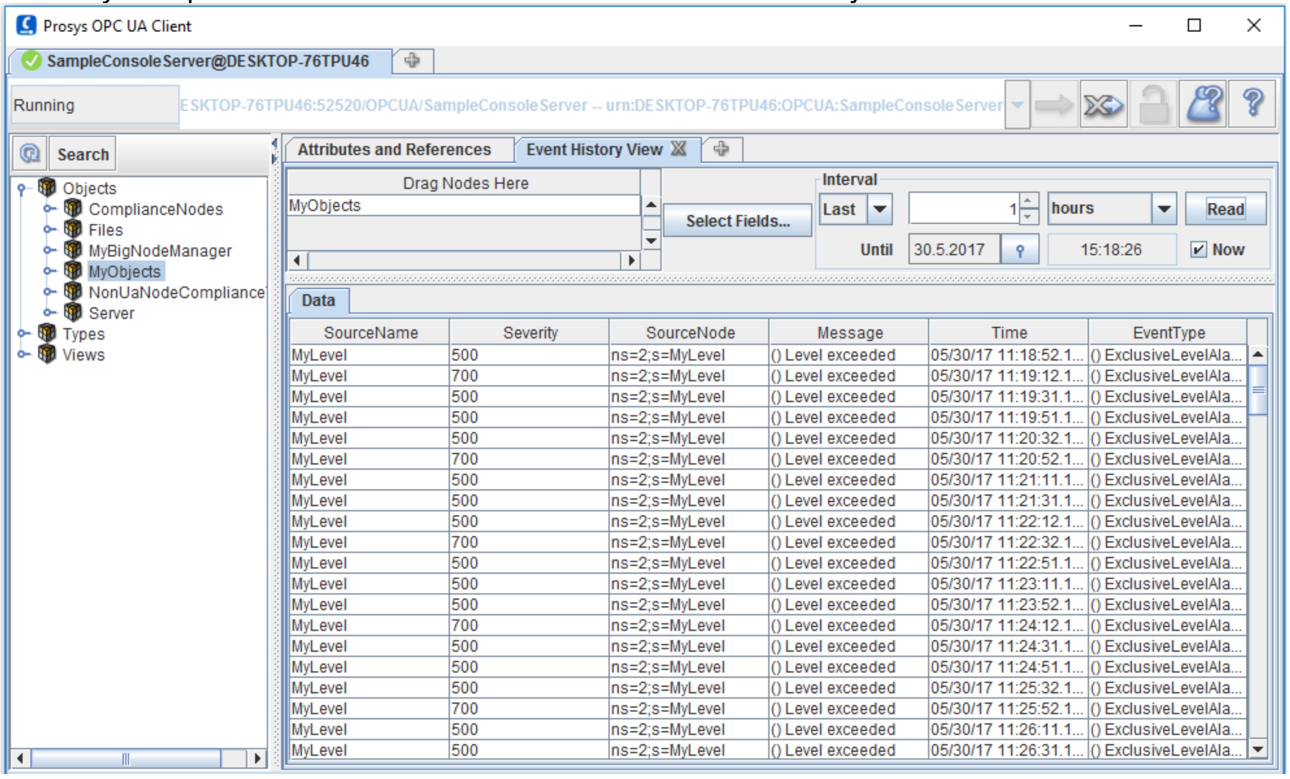


Name	Selected
BaseEventType	☐
AuditEventType	☐
AuditSecurityEventType	☐
AuditSessionEventType	☐
AuditNodeManagementEventType	☐
AuditUpdateEventType	☐
AuditUpdateMethodEventType	☐
ServerId	☐
Status	☐
ActionTimeStamp	✓
ClientUserId	☐
ClientAuditEntryId	☐
SystemEventType	☐
BaseModelChangeEventType	☐
EventQueueOverflowEventType	☐
ProgressEventType	☐
TransitionEventType	☐
ConditionType	☐
MyEventType	☐

Figure 10: Changing Event Fields

7 Event History View

OPC UA Objects, which define HistoryRead in their EventNotifier attribute, may be requested for event history.
The Event History View enables you to drag a valid node to the upper left corner of the view, select a suitable time interval and 'Read' the history. The results are displayed in a list.
You may also press "Select Fields…" to define which information you want to read.

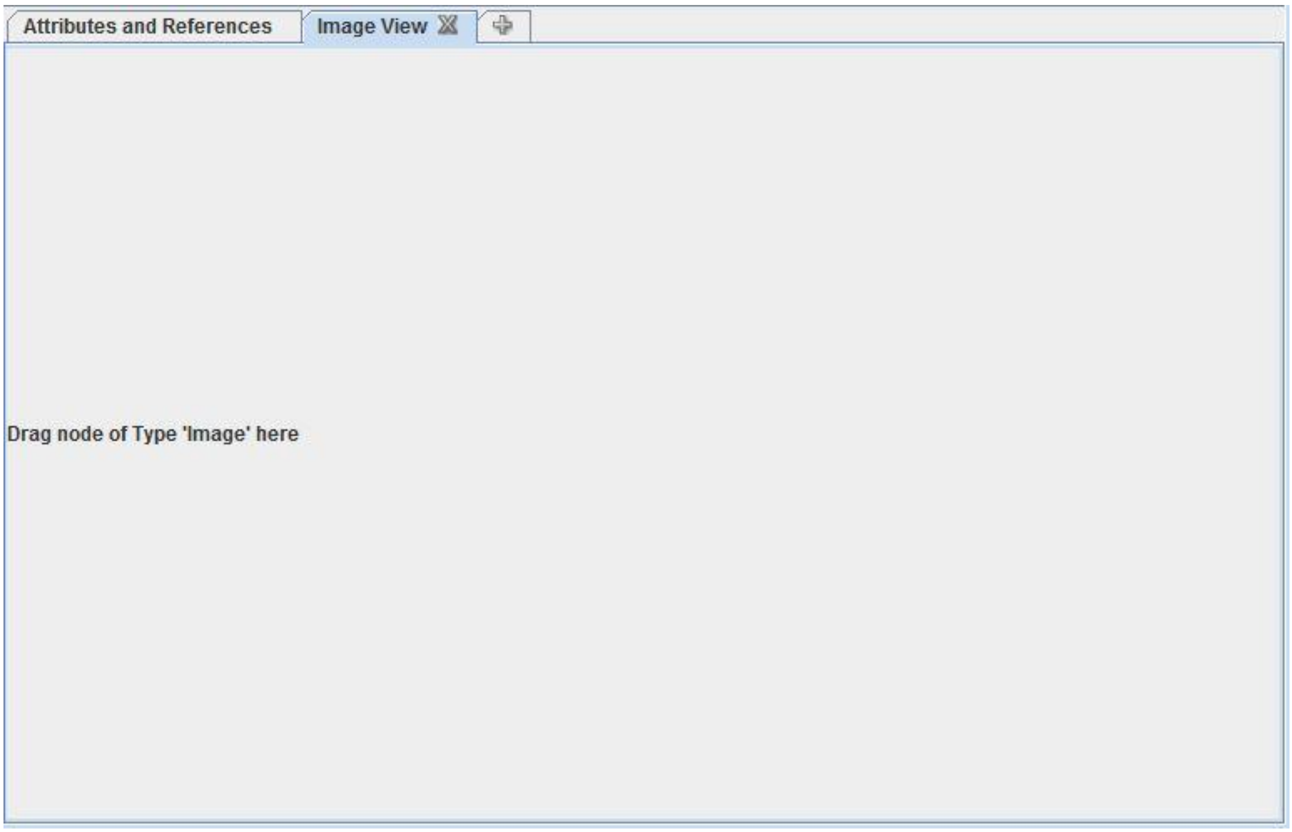


SourceName	Severity	SourceNode	Message	Time	EventType
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:18:52.1...	() ExclusiveLevelAla...
MyLevel	700	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:19:12.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:19:31.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:19:51.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:20:32.1...	() ExclusiveLevelAla...
MyLevel	700	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:20:52.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:21:11.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:21:31.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:22:12.1...	() ExclusiveLevelAla...
MyLevel	700	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:22:32.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:22:51.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:23:11.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:23:52.1...	() ExclusiveLevelAla...
MyLevel	700	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:24:12.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:24:31.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:24:51.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:25:32.1...	() ExclusiveLevelAla...
MyLevel	700	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:25:52.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:26:11.1...	() ExclusiveLevelAla...
MyLevel	500	ns=2;s=MyLevel	() Level exceeded	05/30/17 11:26:31.1...	() ExclusiveLevelAla...

Figure 11. Requesting Event history for MyObjects of the Simulation Server.

8 Image View

In image view, you can view image nodes. Drag a node that is of type Image into the node drag area and the node value will be displayed as image.

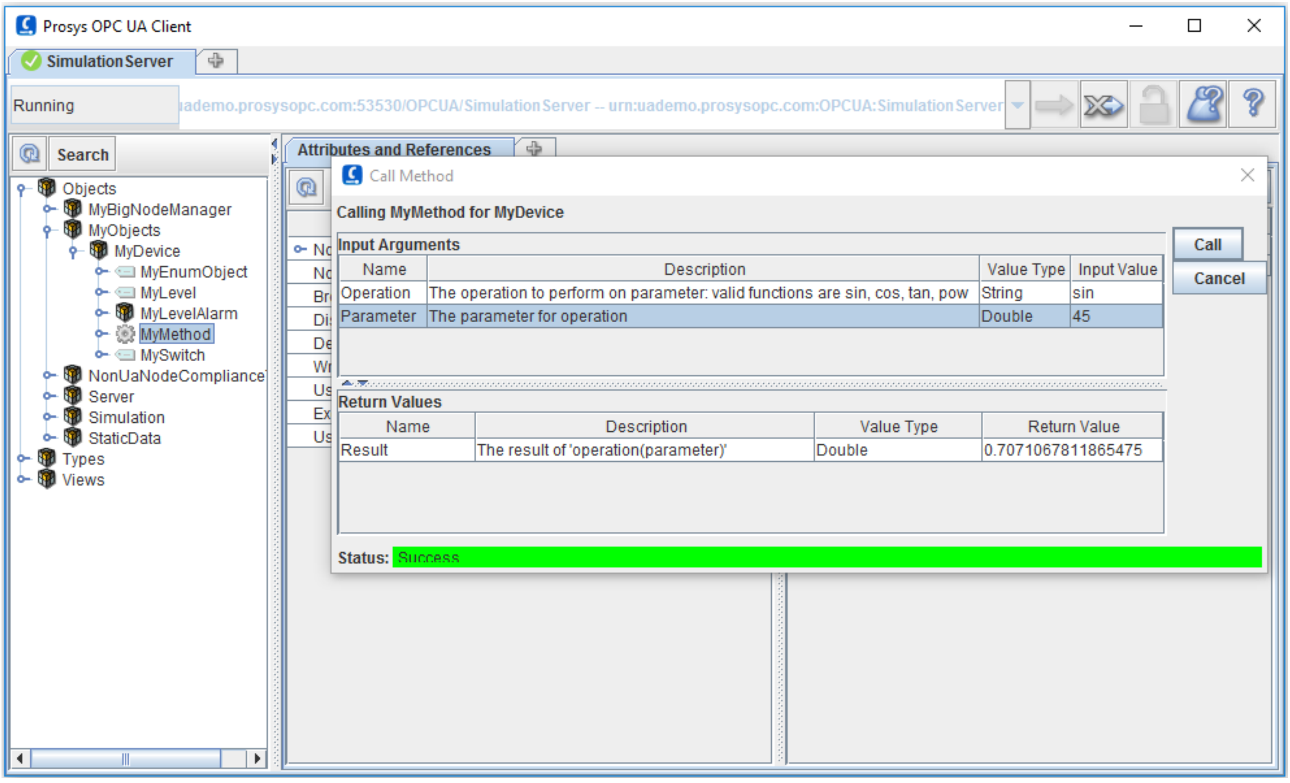


Drag node of Type 'Image' here

Figure 12: Image view.

9 Calling a Method

Calling a method is simple: just find the desired method in the server address space browser tree, right-click it and select Call Method. In the dialog box you can set input and then click Call. Status text field shows whether the call succeeded and output values appear in Return Value column on the right side table view.



Calling MyMethod for MyDevice

Input Arguments

Name	Description	Value Type	Input Value
Operation	The operation to perform on parameter: valid functions are sin, cos, tan, pow	String	sin
Parameter	The parameter for operation	Double	45

Return Values

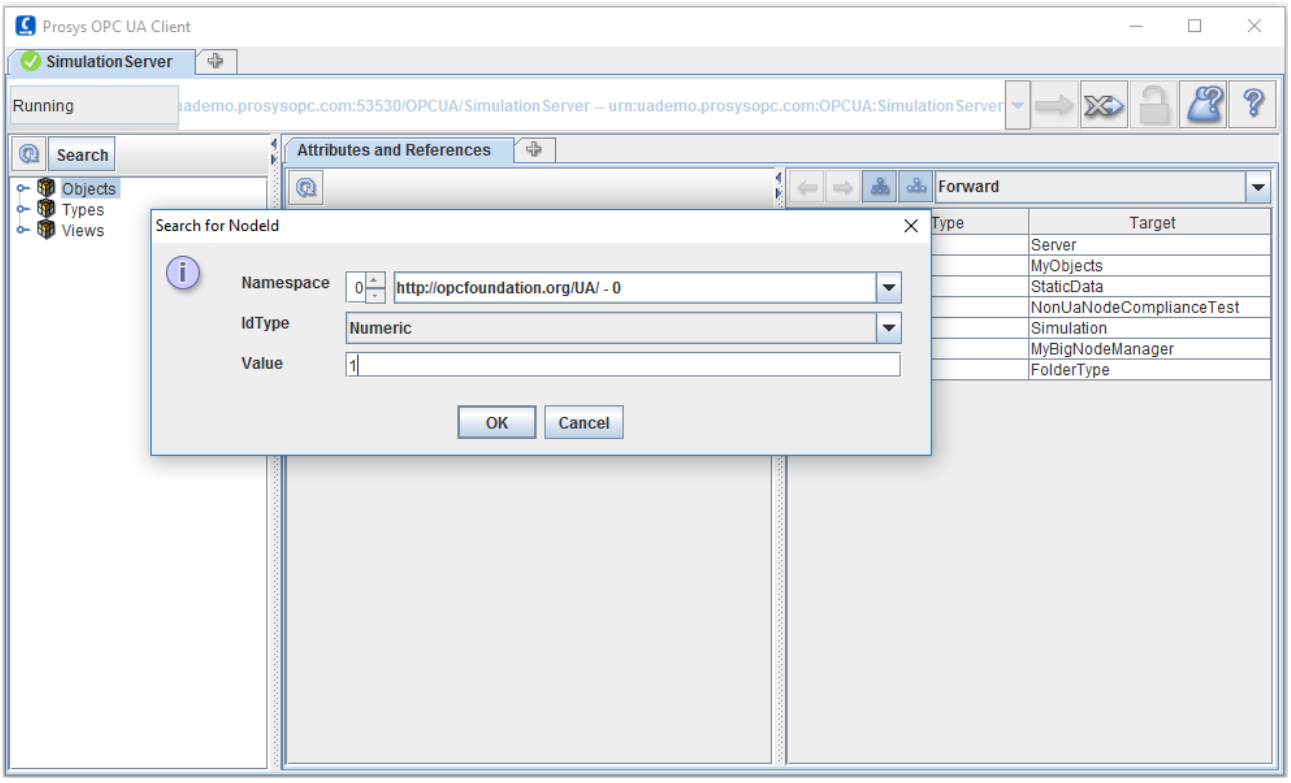
Name	Description	Value Type	Return Value
Result	The result of 'operation(parameter)'	Double	0.7071067811865475

Status: Success

Figure 13: Calling a Method. MyMethod of the SimulationServer enables you to define an operation (sin/cos/tan/pow) and a parameter (in degrees for the trigonometric functions).

10 Searching for node

Above the browser tree, there is a Search button, which is used to search for a node with a particular NodeId. When the button is pressed, a dialog shown in Figure 13 appears. In the search dialog, one can specify the namespace, IdType and value of the NodeId. If a NodeId is found, the corresponding node is selected in the browser tree.



Namespace: 0 http://opcfoundation.org/UA/ - 0
IdType: Numeric
Value: 1

Figure 14: The node search dialog.