OPC UA Browser

User Manual

Version 2026.1.2

Table of Contents

1. Introduction . . . 1
2. Installing the Application . . . 1
    2.1. Windows . . . 1
    2.2. Linux . . . 1
    2.3. macOS . . . 1
3. Uninstalling the Application . . . 2
    3.1. Windows . . . 2
    3.2. Linux . . . 2
    3.3. macOS . . . 2
4. Editions . . . 3
5. How to start using the application . . . 4
6. About OPC UA Servers . . . 8
    6.1. Discovery Servers . . . 8
7. Client/Server Connection . . . 10
    7.1. Connecting to a Server . . . 10
        7.1.1. Security Options . . . 11
        7.1.2. Application Instance Certificates . . . 12
        7.1.3. User Authentication . . . 13
        7.1.4. Reverse Connection . . . 14
    7.2. Address Space Browser . . . 16
        7.2.1. Nodes . . . 16
        7.2.2. Searching for a Node . . . 17
        7.2.3. Sorting Nodes . . . 17
        7.2.4. Copying BrowsePath . . . 18
        7.2.5. Viewing DataTypeDictionary . . . 18
        7.2.6. Writing to a Variable . . . 19
        7.2.7. Calling a Method . . . 21
    7.3. Attributes and References View . . . 22
    7.4. Additional Views . . . 23
        7.4.1. Save Views . . . 23
    7.5. Data View . . . 24
        7.5.1. Adding Variables to Data View . . . 24
        7.5.2. Data View Columns . . . 25
        7.5.3. Subscription Settings . . . 26
        7.5.4. Monitored Item Settings . . . 26
        7.5.5. Data Logging . . . 29
    7.6. History View . . . 30
        7.6.1. Adding Variables to History View . . . 30
        7.6.2. Reading History . . . 31
        7.6.3. Aggregate Calculation . . . 31
    7.7. Event View . . . 33
        7.7.1. Selecting the Monitored Object . . . 33
        7.7.2. Selecting Event Fields . . . 34
        7.7.3. Basic Events and Conditions . . . 36
        7.7.4. Condition Refresh . . . 36
    7.8. Event History View . . . 37
        7.8.1. Selecting the Object . . . 37

7.8.2. Reading History . . . 37
7.8.3. Selecting Event Fields . . . 37
7.9. Image View . . . 38
7.10. NodeSet Export View (Professional Edition only) . . . 39
7.10.1. NodeSets . . . 39
7.10.2. Use cases of NodeSets . . . 40
7.10.3. Exporting Namespaces to NodeSets . . . 40
7.10.4. Advanced NodeSet Export View . . . 42
7.11. CSV Node Export View . . . 43
8. PubSub Connection . . . 45
8.1. Connecting to a PubSub Network . . . 45
8.1.1. MQTT Network . . . 45
8.1.2. UDP Network . . . 46
UDP Multicast . . . 46
UDP Unicast . . . 47
Network Interface . . . 47
8.2. PubSub Connection View . . . 47
8.2.1. PubSub View . . . 48
PubSub Data View . . . 49
PubSub Event View . . . 50
8.2.2. PubSub Log View . . . 50
9. Preferences . . . 53
10. File Locations . . . 54
10.1. Application Logs . . . 54
10.2. Certificate Store . . . 54

1. Introduction

Prosys OPC UA Browser is a general-purpose OPC UA Client and OPC UA Subscriber. You can test connections and view data from any OPC UA Server. The main functions of the OPC UA Browser include browsing the Server AddressSpace, reading and writing Variables, monitoring Variables and Events, and reading a history of Variables and Events. Additionally, you can connect and subscribe to a PubSub Network and receive and observe messages from PubSub Publishers.

The newest versions of the OPC UA Browser can be installed with an auto-update feature offered during the installation process. If you do not want to enable the auto-update feature, you can find the option to check for updates manually from the application's *Help* menu whenever you choose.

2. Installing the Application

If upgrading from version 3.x.x or earlier, you should note that the locations for the installation and the settings have changed. All your previous settings will be lost. The older version of Prosys OPC UA Client (the previous name) is not automatically removed but can be uninstalled manually.

The installation includes a complete "embedded" Java Runtime Environment (JRE). This ensures that although the application runs in a Java environment, you do not need to install Java on your computer and worry about the Java updates. The embedded Java is only used for this application.

The application install packages are available from http://www.prosysopc.com upon your request. You should receive the correct package, depending on your target environment.

2.1. Windows

On Windows, run the installer executable prosys-opc-ua-browser-windows-x64-2026.1.2-xxx.exe and follow the instructions. By default, the application is installed to the folder *Program Files/ProsysOPC/Prosys OPC UA Browser*.

2.2. Linux

The application requires a GUI (Linux Desktop Environment) in order to run.

On Linux, first open the terminal and navigate to the directory of the downloaded *.sh* file.

Then run the installation with the command

```
sudo sh prosys-opc-ua-browser-linux-x64-2026.1.2-xxx.sh
```

This will open the installer where you can follow the steps to complete the installation. By default, the application is installed to the folder *opt/prosys-opc-ua-browser*.

2.3. macOS

On macOS, run the installer application and follow the instructions. By default, the application is installed to the folder /Applications.

3. Uninstalling the Application

3.1. Windows

On Windows the application can be uninstalled through the Control Panel or the *Apps & features* menu, or optionally with the uninstaller that is located in the installation folder.

3.2. Linux

On Linux, open the terminal and navigate to the installation folder (default folder is *opt/prosys-opc-ua-browser*) and use the command sudo ./uninstall.

3.3. macOS

On macOS you can just remove the application from the Applications folder.

4. Editions

Prosys OPC UA Browser has two editions: Free and Professional. See below, how license terms relate to editions.

- *Free license* limits the functionality of the application. Support is limited to forum-based support.
- *Evaluation license* is time-limited, but otherwise equivalent to the Professional license, including forum- and email-based support. After the expiration date, the application reverts to the Free license.
- *Professional license* is unlimited in functionality and time. Eligible for both forum- and email-based support.

The Professional Edition currently adds these features to the Browser:

- Exporting Server Namespaces to NodeSet XML files in NodeSet Export View.
- Exporting Server Nodes to CSV files in CSV Node Export View.

To receive an Evaluation or Professional license, please contact sales@prosysopc.com.

5. How to start using the application

This chapter briefly explains how you can get started with the OPC UA Browser. If you prefer to watch a video tutorial, you can find a four-part series on our YouTube channel: https://www.youtube.com/watch?v=M3rBjffmmUk&list=PLZpqH2EeqVW3P4NQGgXn9tsulWh_p5cMN

The OPC UA Browser remembers the Server and PubSub connections you have made. The connections are listed for you as *Quick Links* in the starting view of the application (see Figure 1). Your favorite connections are listed first, followed by the rest of the connections. You can add and remove favorite connections by clicking the star icon on the right side of the Quick Links in the starting view or on the left side of the address bar when connected. You can remove connections from Quick Links by clicking the 'X' icon below the star icon.

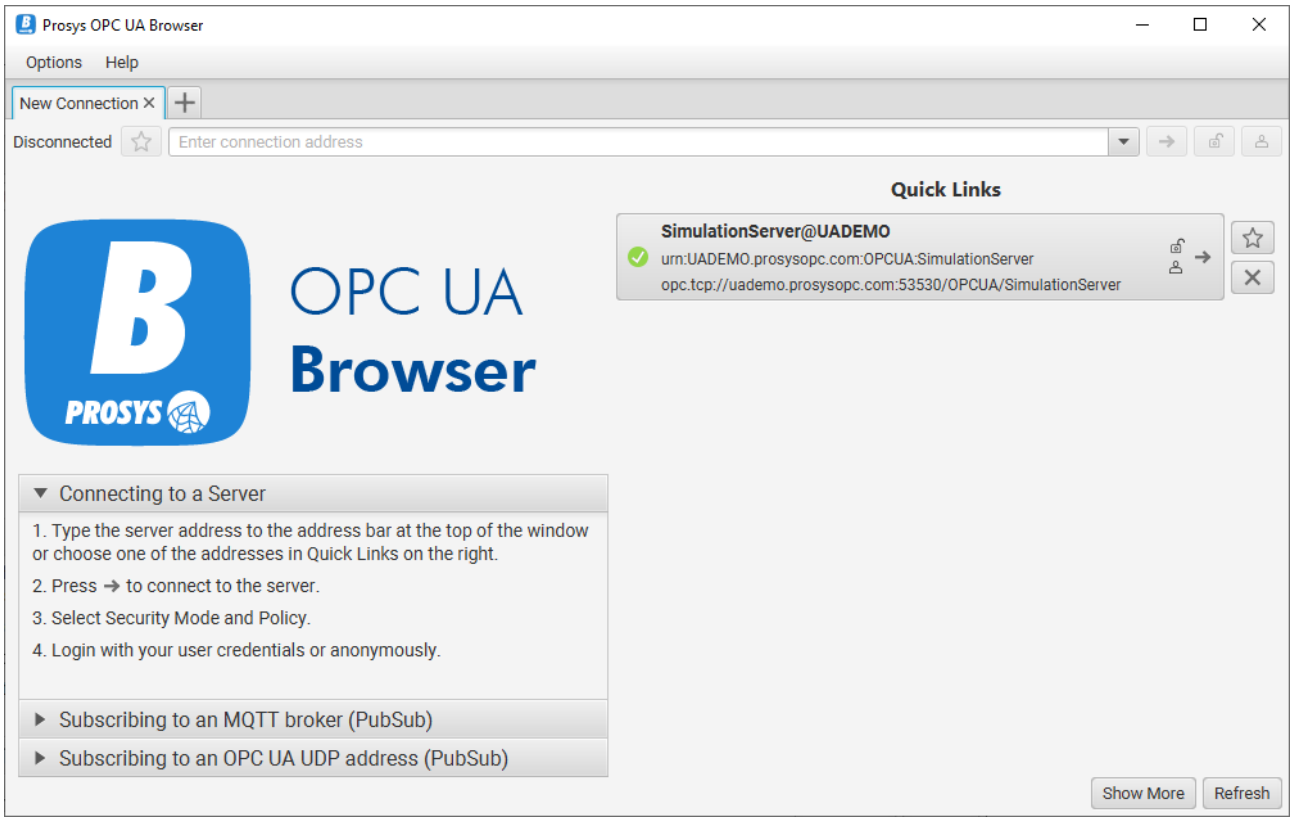

Figure 1. Starting view of the OPC UA Browser

The OPC UA Browser has the internet-based UADEMO Server connection configuration set for you by default. This way, you can get started right away, even if you do not know any other server addresses. If you wish to connect to a different Server, follow the instructions on the left side of the starting view to establish a connection.

Secure connections can not be formed to the UADEMO Server. You can read more about security in Security Options.

With the OPC UA Browser, you can have multiple Server connections and manage them using tabbed pages. The AddressSpace of each Server is represented in the *Address Space Browser* on the left side of the user interface window.

Attributes and *References* is a default view for every Server. By clicking the '+' button in the *Views Bar* highlighted in Figure 2 by the red circle, you can add any number of the other views: *Data View* for monitoring Variables, *History View* to explore historical data of Variables, *Event View* to monitor Events and Alarms and *Event History View* for requesting the history of Events from the Server. *Image View* provides a unique view for monitoring Image data. *NodeSet Export View* and *CSV Node Export View* enable exporting the Server's AddressSpace to NodeSet XML and CSV respectively. Figure 2 illustrates the layout of the OPC UA Browser when connected to a Server.

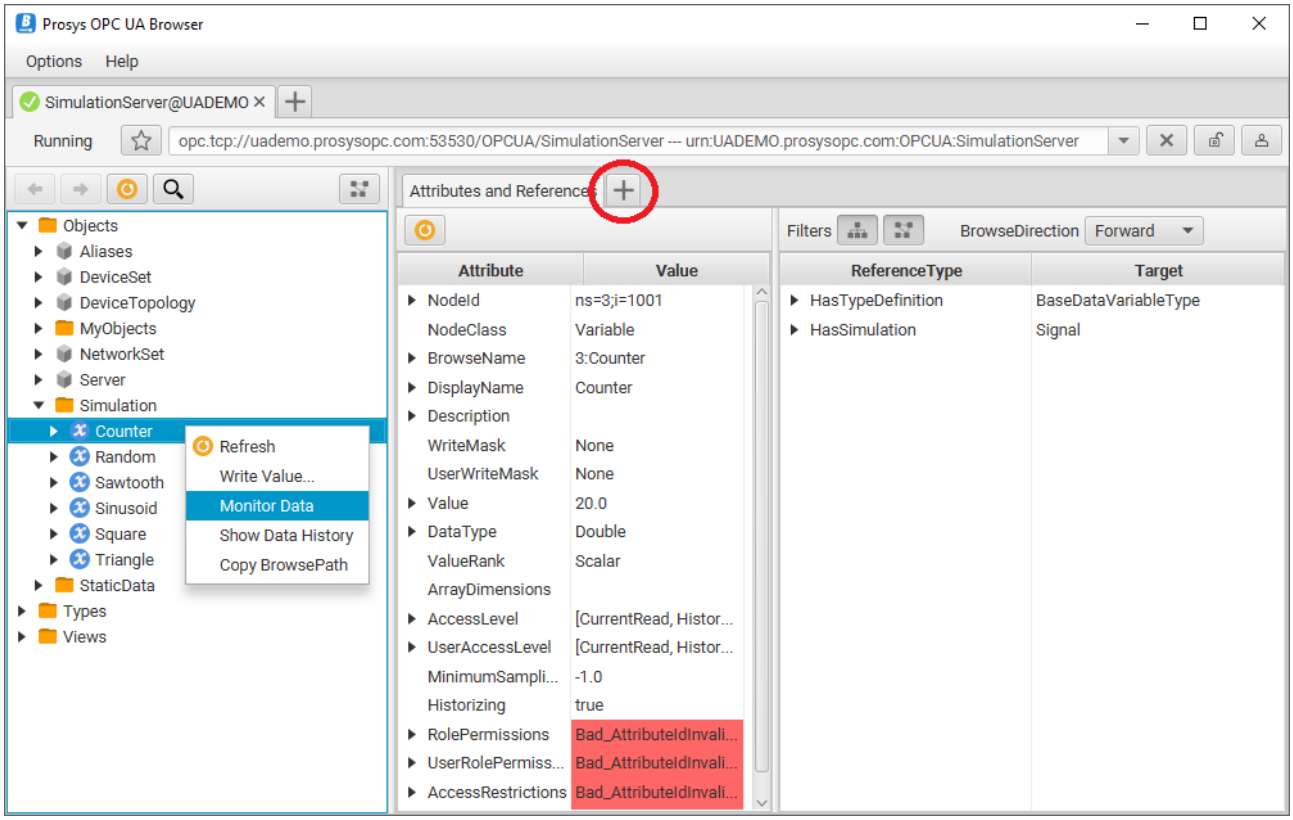

Figure 2. OPC UA Browser Overview

You can find the most useful Nodes from the *Objects* folder on the left side of the view shown in Figure 2. If you want to monitor a Variable from the Objects folder, right-click on a Variable and select *Monitor Data* (see Figure 2) to monitor it. This will open a new Data View for you with the chosen Variable. Alternatively, you can click on the '+' button mentioned above to open a Data View and drag the Variable into the table (see Data View for more information).

In Figure 3 you can see that all six Variables under *Simulation* have been selected and dragged to be monitored in the *Data View*.

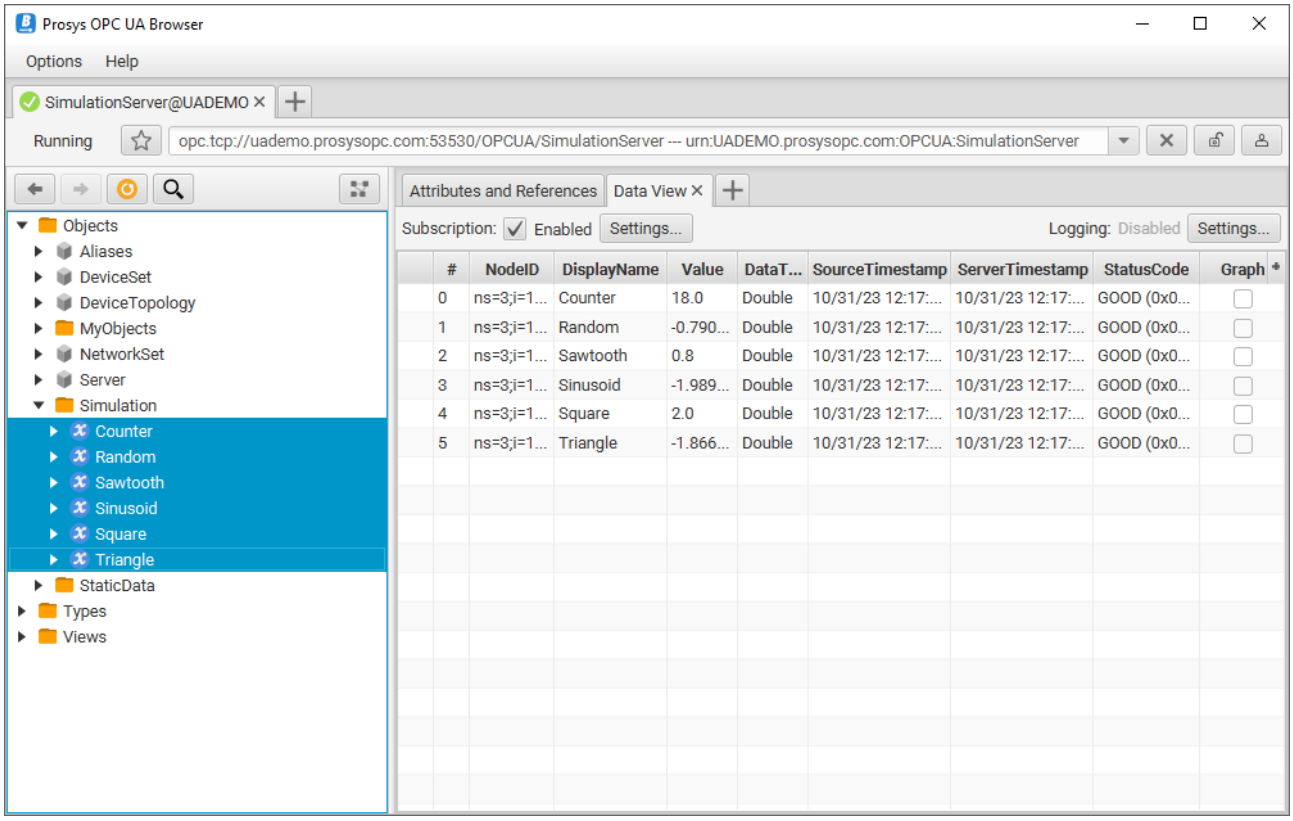

Figure 3. Data View with six Variables.

In addition to connecting to OPC UA Servers, you can use the OPC UA Browser as a OPC UA Subcriber to subscribe to PubSub Networks. The starting screen gives you instructions on how to subscribe to a MQTT Broker or an OPC UA UDP address, as seen in Figure 4.

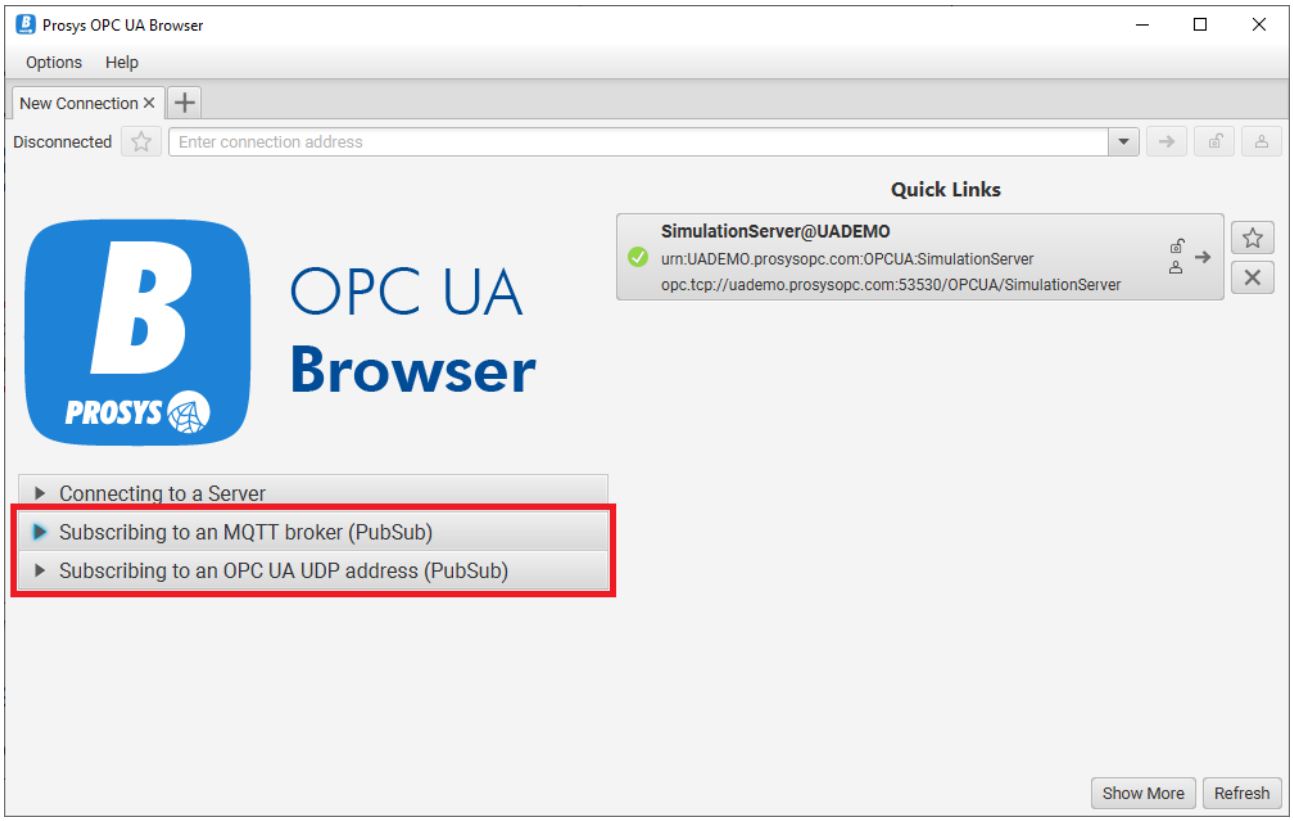

Figure 4. Instructions for the new PubSub Connection feature.

See PubSub Connection for more details.

6. About OPC UA Servers

OPC UA Servers are connected with a server address. To establish a connection, you need to find out the Server's address that you wish to connect to.

The address always takes the form

```
<Protocol>://<Hostname>:<Port>/<ServerName>
```

The address starts with a Protocol identifier, which is most often opc.tcp:// (corresponding to OPC UA TCP). The protocol identifier is followed by the hostname of the computer where the Server is running (localhost always works locally) and the TCP port the Server is listening to. OPC UA defines a standard port number as 4840, but this is often reserved for a Discovery Server, and the actual Servers use custom port numbers. The server address may also contain a ServerName part. If the ServerName is not defined, the / character may be omitted before it.

If you do not have any actual OPC UA Servers available to connect to, you can use the Prosys OPC UA Simulation Server to play around with the application. You can download the Prosys OPC UA Simulation Server from https://www.prosysopc.com website and run it locally. You can then use the following server address:

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

Alternatively, you can connect to a publicly available Simulation Server via the Internet. The address of that Simulation Server is listed below:

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

6.1. Discovery Servers

OPC UA defines two types of Discovery Servers:

- Local Discovery Server (LDS) is often installed on the PC to keep a list of installed Servers on the local computer. OPC Foundation provides a standard implementation of the LDS.
- Global Discovery Server (GDS) is an application that is designed to keep a list of Servers installed on a site-wide network. The GDS can also manage application instance certificates as a central Certificate Authority (CA). OPC Foundation provides a sample implementation of the GDS.

Also, each OPC UA Server contains an internal Discovery Server that can provide similar information about the Server application itself. The LDS typically listens to the TCP 4840 Port (having address opc.tcp://localhost:4840). Several embedded devices and PLCs, which do not have any LDS, also use Port 4840 for the actual server address.

If you have a GDS available, you need to find its address, similar to other OPC UA Server applications.

The connection address of the internal Discovery Server of the previously mentioned publicly available Simulation Server is:

```
opc.tcp://uademo.prosysopc.com:53530
```

After connecting to a Discovery Server, a list of available Servers is opened (see Figure 5). There you can select which Server you want to connect to.

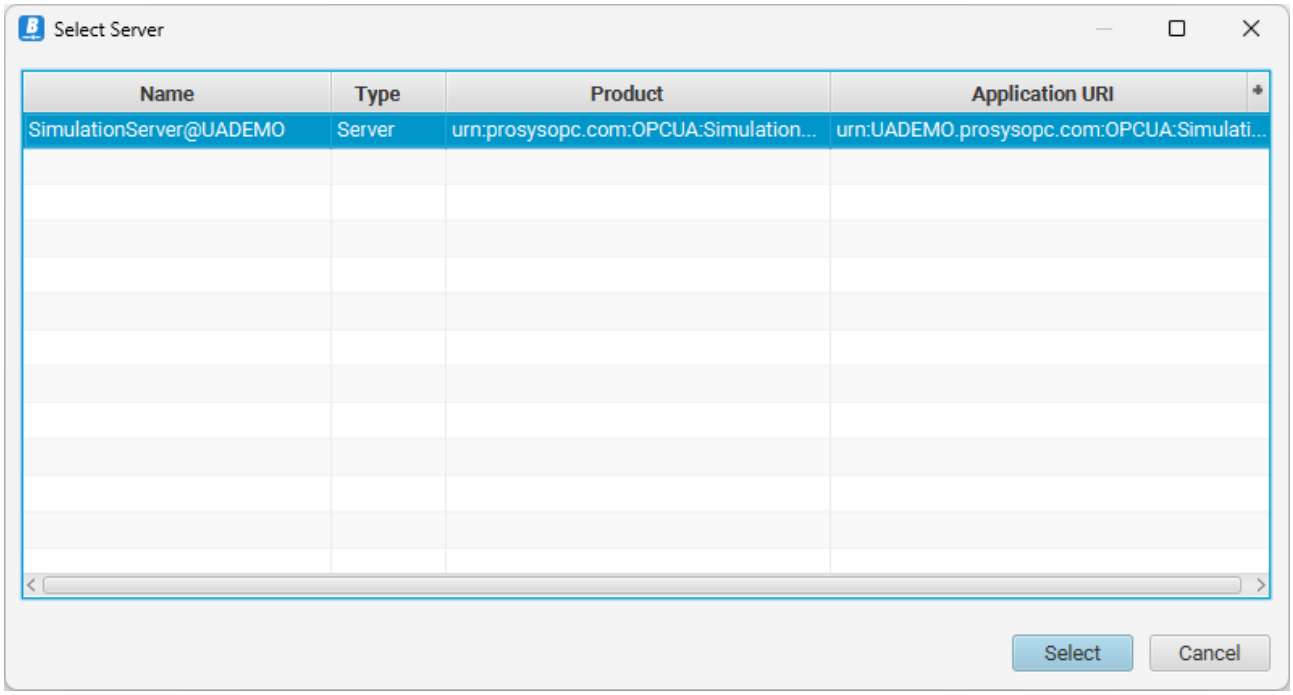

Figure 5. List of available Servers.

7. Client/Server Connection

7.1. Connecting to a Server

Connecting to an OPC UA Server is based on the server address that can be typed in the address bar at the top of the window (see Figure 6). Previously used addresses are available in the dropdown menu and as *Quick Links* in new tabs so that it is easy to reselect them. On the right-hand side of the address bar, there is a button for connecting to and disconnecting from the Server. If Browser cannot connect to the Server, an error dialog is shown, displaying the error message that explains the failure to establish a connection.

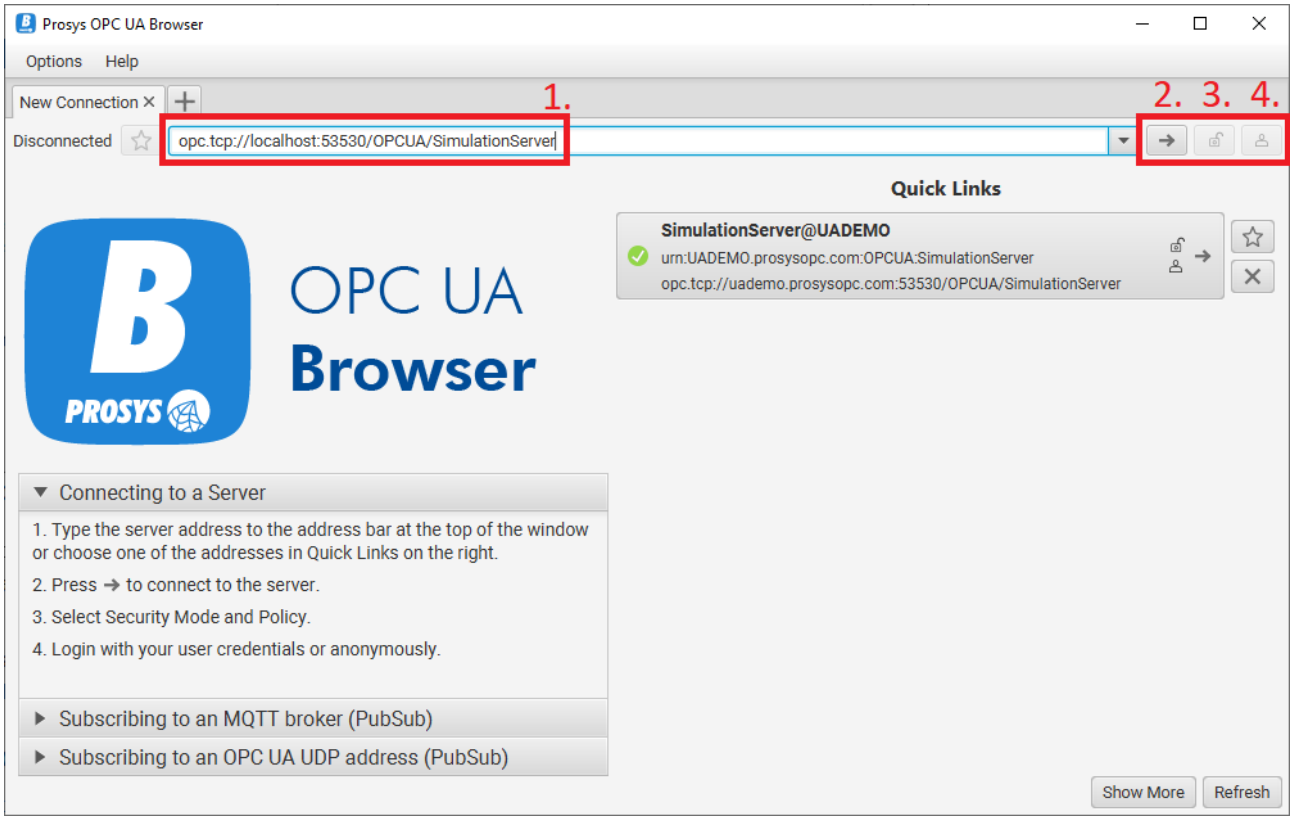

Figure 6. Server Address Bar (1.) Connect/Disconnect button (2.) Security Settings (3.) and User Authentication Settings (4.)

If the Client manages to connect but determines that the target is an OPC UA Discovery Server (see Discovery Servers above), it prompts for a selection from the available server addresses provided by the Discovery Server. You can then choose one of the actual server addresses to make a new connection attempt.

Prosys OPC UA Browser accepts partial addresses as well: if you only enter the hostname, opc.tcp is assumed to be the protocol, and the port number will be set to 4840.

7.1.1. Security Options

Whenever you make a connection to a Server, you must choose the level of security to use. The security options include Security Mode and Security Policy. The available Security Modes are:

1. Sign & Encrypt
2. Sign
3. None

Choosing Sign & Encrypt gives the highest level of security, protecting the contents of the communication from being seen by any third parties. If Sign is selected, the messages used in the communication are signed to protect them from alteration during transfer. If the selected Security Mode is None, the connection is not secured at all.

The Security Policy determines the algorithm for signing and encrypting. As defined by the OPC UA Specification (Figure 7), there are multiple Security Policies available. Each one of them defines a set of security algorithms and parameters to use for the OPC UA TCP communication:

1. Aes256Sha256RsaPss
2. Aes128Sha256RsaOaep
3. Basic256Sha256
4. Basic256 (deprecated in OPC UA version 1.04)
5. Basic128Rsa15 (deprecated in OPC UA version 1.04)

Basic256 and Basic128Rsa15 are deprecated policies due to known security vulnerabilities, so they are not recommended to be used, if other policies are available.

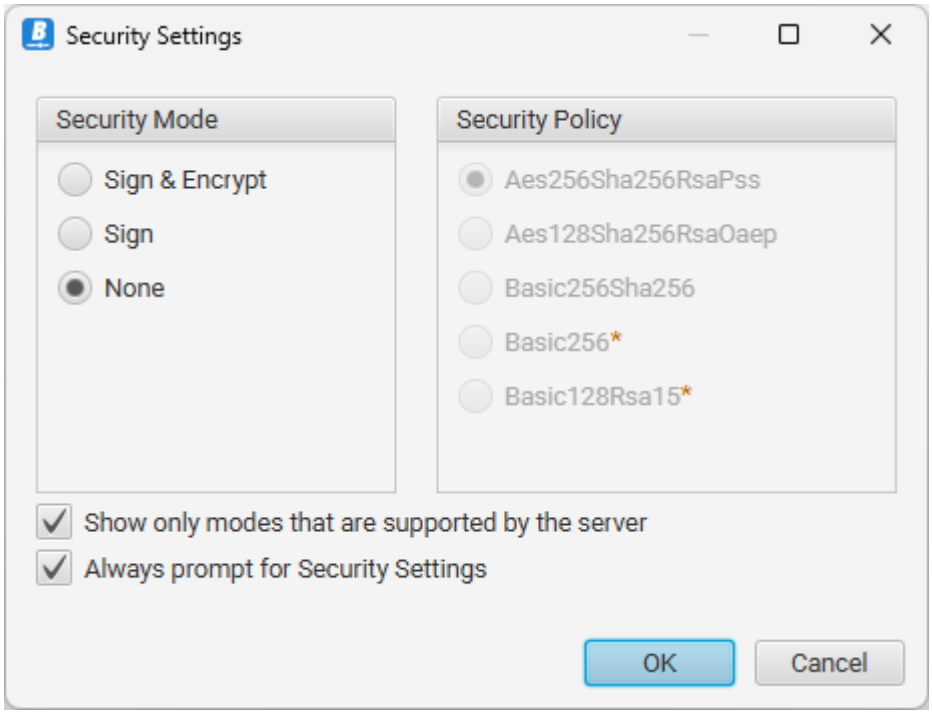

Figure 7. Security Settings

To establish a connection, the selected mode and policy must match the ones supported by the Server application. To help find an applicable mode, you can use the *Show only modes that are supported by the server* option.

If you wish to use a secure connection (Sign or Sign&Encrypt), the Client and Server applications must trust each other. To build a trust relationship, the applications use Application Instance Certificates.

7.1.2. Application Instance Certificates

OPC UA applications are identified using Application Instance Certificates. All applications that wish to communicate securely need to have a certificate. The certificates enable identifying the applications reliably. The certificates are based on Public Key Infrastructure (PKI) principles defined by the X.509 standard.

To build the trust relationship between each other, the applications use a Certificate Store where they keep the certificates of other known applications and where they can define which certificates (and applications) are trusted.

Typically OPC UA applications can create the certificate for themselves. Such certificates are called self-signed certificates. In the case of self-signed certificates, each application must form trust with other applications separately. Therefore, when you try to connect securely to a new Server for the first time, you are prompted to trust the Server certificate (see Figure 8). After trusting the Server certificate, you will usually get a Bad_SecurityChecksFailed error, indicating that the Server does not trust Browser. You must then go to the Server configuration, locate the certificate of Browser, and define that it can be trusted.

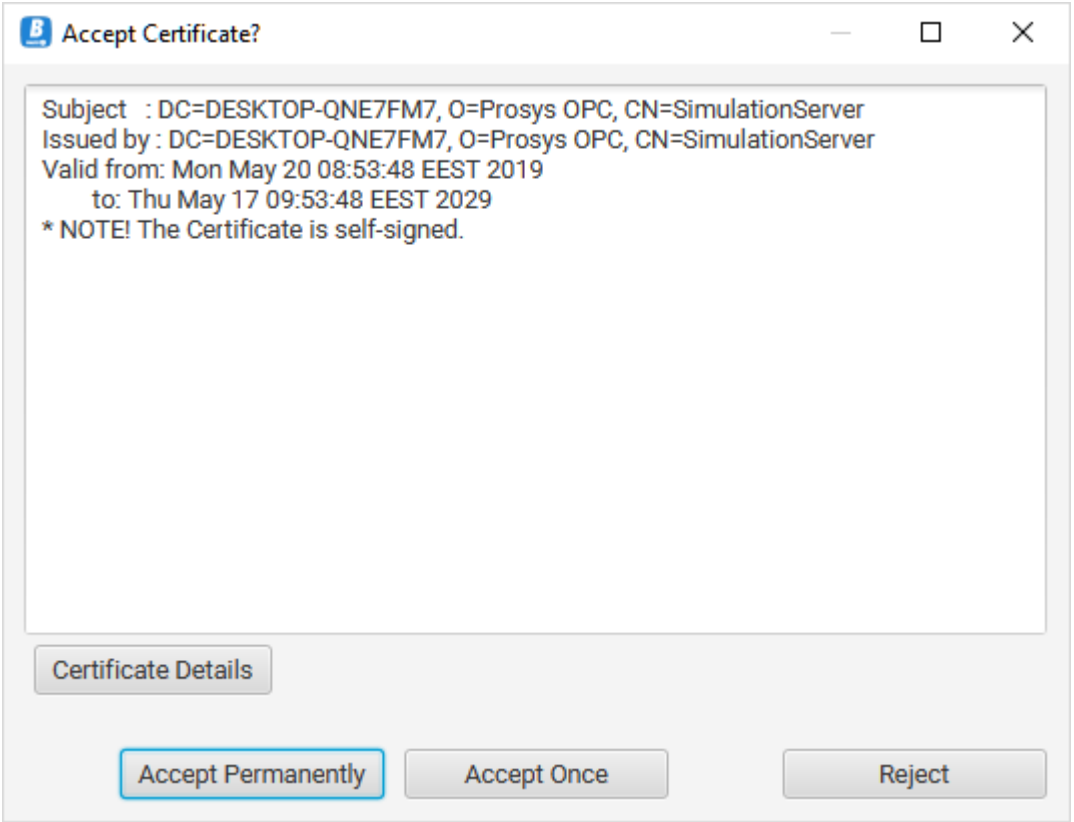

Figure 8. Certificate verification dialog

If you select *Accept Permanently*, the certificate is moved to the list of trusted certificates in the Client. From now on, as long as the certificates are valid, both Client and Server trust each other and can open secure communication with each other.

The Certificate contains information about the application and a cryptographic Public Key. A separate Private Key accompanies it. These two keys enable asymmetric encryption, which is the basis of the OPC UA security mechanism. To keep things secure, the private key must be kept secret from other people and applications. It should be available only for the application itself.

The *Certificates* dialog can be opened from the *Help* menu. There you can find the certificate for the Prosys OPC UA Browser itself and all the certificates of known Server applications (see Figure 9). In this dialog, you can examine, trust and reject certificates.

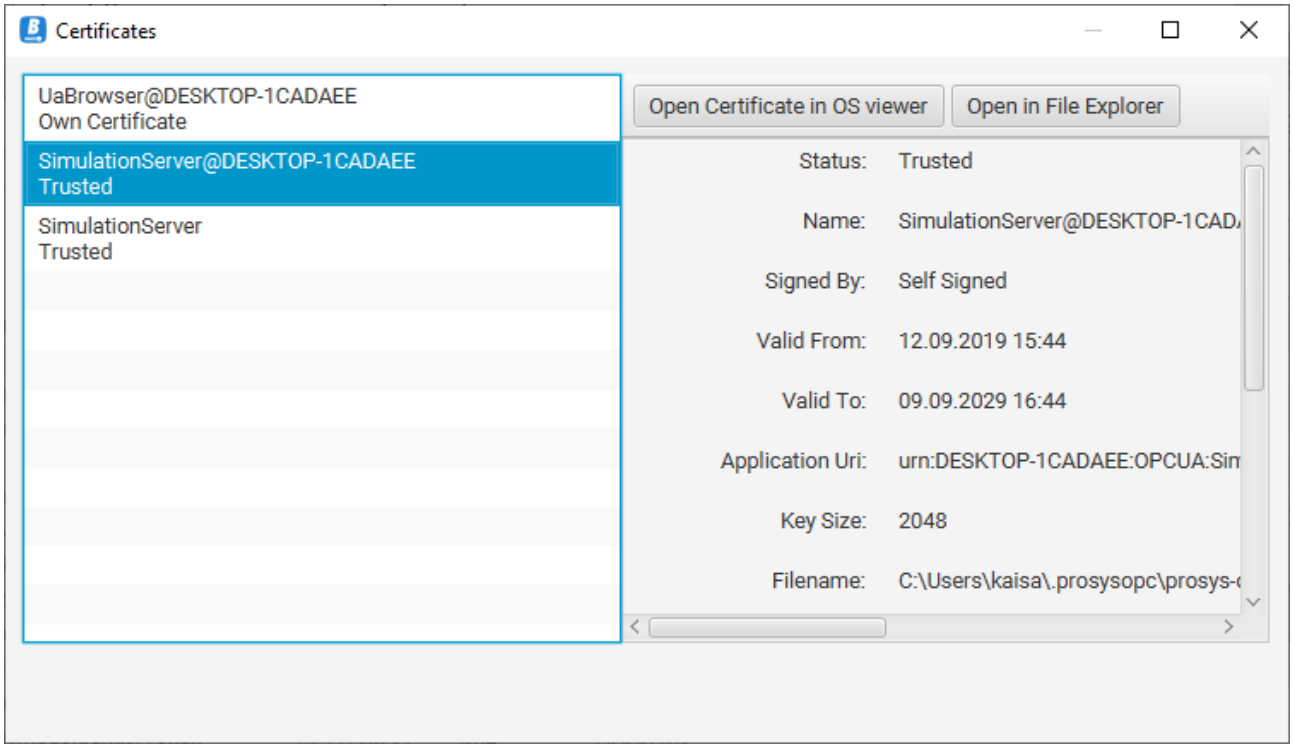

Figure 9. Certificates dialog lists all certificates that have been given to the Prosys OPC UA Browser

7.1.3. User Authentication

In addition to application-level authentication, OPC UA also supports user authentication. Users can be identified with standard Username & Password combination or X.509 certificates (similar to the applications) or by some external user authentication system. OPC UA also enables anonymous connections without any user identification. Prosys OPC UA Browser currently supports all but external authentication systems.

The user identity may be defined before establishing the connection or while the connection is open. Select the user authentication setting by clicking the user icon in the upper right corner of the user interface (see Figure 6 and Figure 10). It can also be applied when the connection is established to impersonate the Session to the identified user account. Note, that the username and password must be set on the Server in order for you to use them.

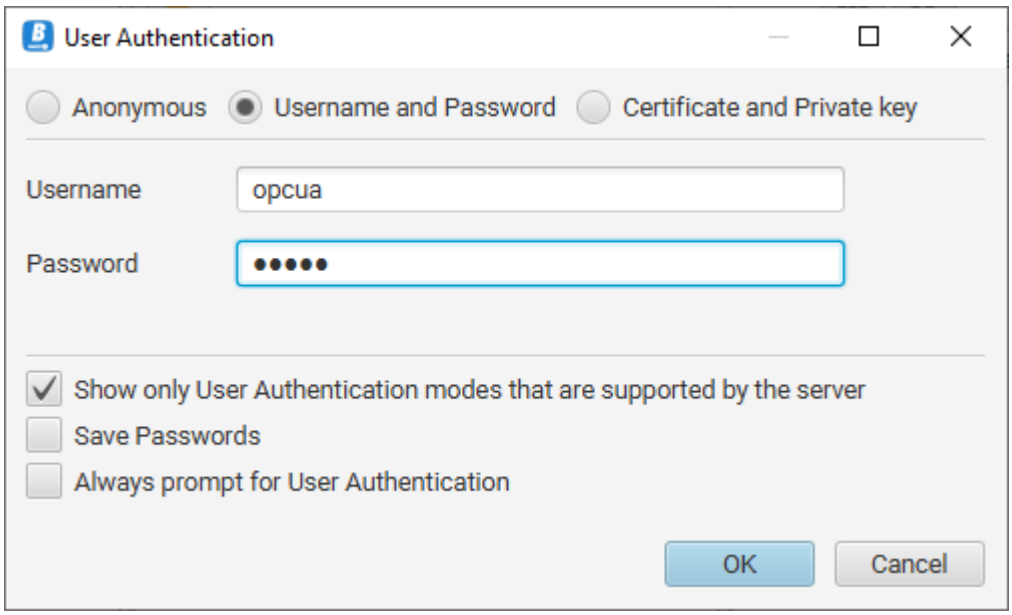

Figure 10. Defining the user authentication mode and user account. You can change the user account before or during a connection. Prosys OPC UA Simulation Server enables you to define your usernames and passwords to test with.

7.1.4. Reverse Connection

Prosys OPC UA Browser supports the connect mechanism where the OPC UA Server initiates the connection. This can be used to access Servers through firewalls without opening any ports from the Client to the Server, since the connection is opened in practice from the Server to the Client.

To establish the reverse connection, we need to make the Client listen to connections in one TCP port. To do this, you can use the inv+ prefix in the protocol of the connection address in the following format:

```
inv+opc.tcp://<Hostname>:<Port>
```

for example

```
inv+opc.tcp://EXAMPLE:61610
```

Alternatively, you can use one of the following shorthand notations inv, inverse, rev, reverse with a port number, as follows:

```
reverse:61610
```

which will produce the same address

```
inv+opc.tcp://EXAMPLE:61610
```

when the respective port is opened for connections.

In order to open the connection from the Server side, you will need to copy this address to the Server that supports reverse connections.

The hostname part of the address is used to determine, which local interfaces will be listened for incoming connections. If you use the hostname of the computer, it will listen to all interfaces (all IP addresses). If you use a specific IP address, only the respective network interface will be listened to for incoming connections. 'localhost' is treated as the local loopback address.

You can use any free TCP port number and the number 61610 is just an example here.

When entering the address in the Server, you may need to change the hostname to an address that the Server can use to open the connection to this computer.

7.2. Address Space Browser

Once you are connected to a Server, you can browse the Server's AddressSpace to locate the data and information you are interested in. You can do this with the Address Space Browser, which is on the left-hand side of the application window (see Figure 11). As defined by the OPC UA Specification, AddressSpace always contains three main folders: Objects, Types, and Views.

The *Objects* folder contains all data of the Server: Objects and Variables. We also call these instances.

The *Types* folder contains all metadata of the Server: ObjectTypes, VariableTypes, DataTypes, EventTypes, and ReferenceTypes. The metadata helps to understand the definition of the instances, the semantics of the information model that the Server uses to group the data.

The *Views* folder is meant to be used to define alternate layouts of the Server's information model, but it is not commonly in use.

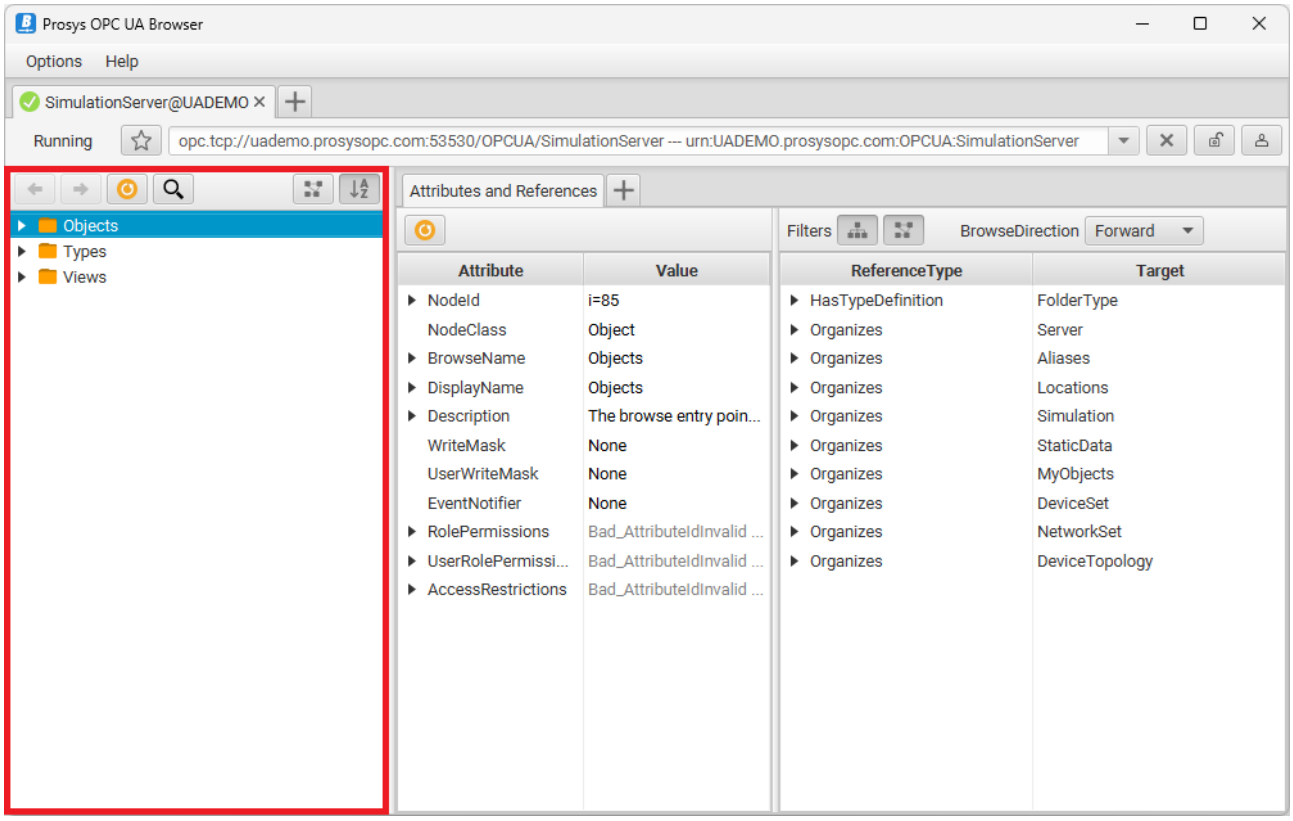

Figure 11. The Address Space Browser is used to locate instances (in the Objects folder) and types (in Types folder) from the Server. Views folder is used only very seldom.

7.2.1. Nodes

All items in the AddressSpace are called Nodes: Objects, Variables, Types, et cetera. The Nodes have several Attributes that define specific details of each Node. For example, Variables have a Value attribute.

The main attribute of each Node is the NodeId; a unique identifier used to identify the Node in the Server.

In addition to Attributes, the Nodes also have References. The References are used to define relations between the Nodes: hierarchical References enable the AddressSpace to be viewed as a tree, and additional non-hierarchical References can define other relations, such as the type of an Object.

See chapter Attributes and References View for more information.

7.2.2. Searching for a Node

At the top of the Address Space Browser, you can find a Search button, which helps you to locate a Node with a particular NodeId. In the search dialog (see Figure 12), you can either parse the NodeId from text or specify the Namespace, IdType, and Value of the NodeId using the available controls. If the respective Node exists, it is activated in the Address Space Browser.

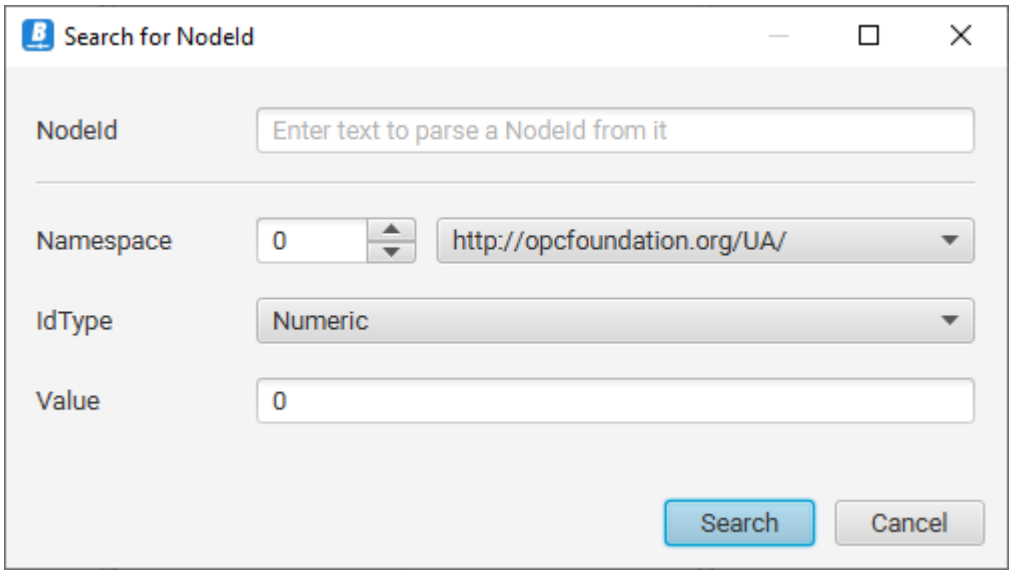

Figure 12. The Node search dialog.

7.2.3. Sorting Nodes

Nodes in the Address Space Browser can be sorted either in alphabetical order based on their DisplayNames or in the order they were received from the Server. This sorting can be configured in the menu opened by clicking on the Menu icon in the Address Space Browser (see Figure 13) and enabling or disabling the *Sort Alphabetically* option (see Figure 14). By default, alphabetical sorting is used.

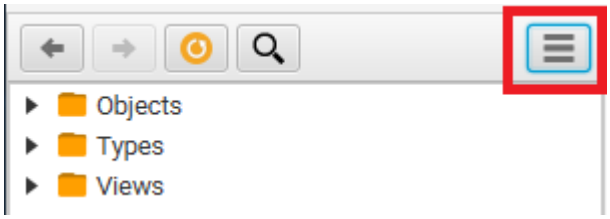

Figure 13. The Menu icon.

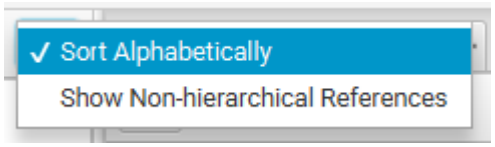

Figure 14. The Sort Alphabetically option.

7.2.4. Copying BrowsePath

Every Node in the Address Space Browser has a context menu option to *Copy BrowsePath* (see Figure 15). The whole BrowsePath will be copied to clipboard, so you can paste it somewhere useful.

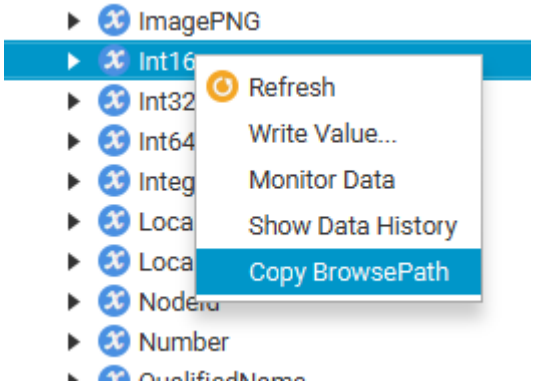

Figure 15. Option to Copy BrowsePath in the context menu.

When you paste the BrowsePath somewhere, it will be in a format like this:

```
/Root/Types/DataTypes/OPC Binary/Opc.Ua
```

7.2.5. Viewing DataTypeDictionary

For DataTypeDictionaryType Nodes, Address Space Browser offers an option to view their DataTypeDictionary. This option can be found in the context menu of the Address Space Browser (see Figure 16).

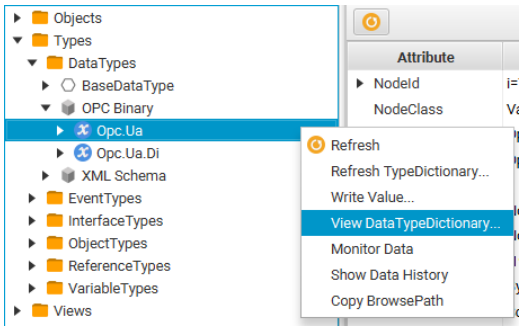

Figure 16. You can view a DataTypeDictionaryType Node's DataTypeDictionary from the Address Space Browser context menu.

A new dialog will appear, representing the DataTypeDictionary in a format that can easily be copied and pasted somewhere else (see Figure 17).

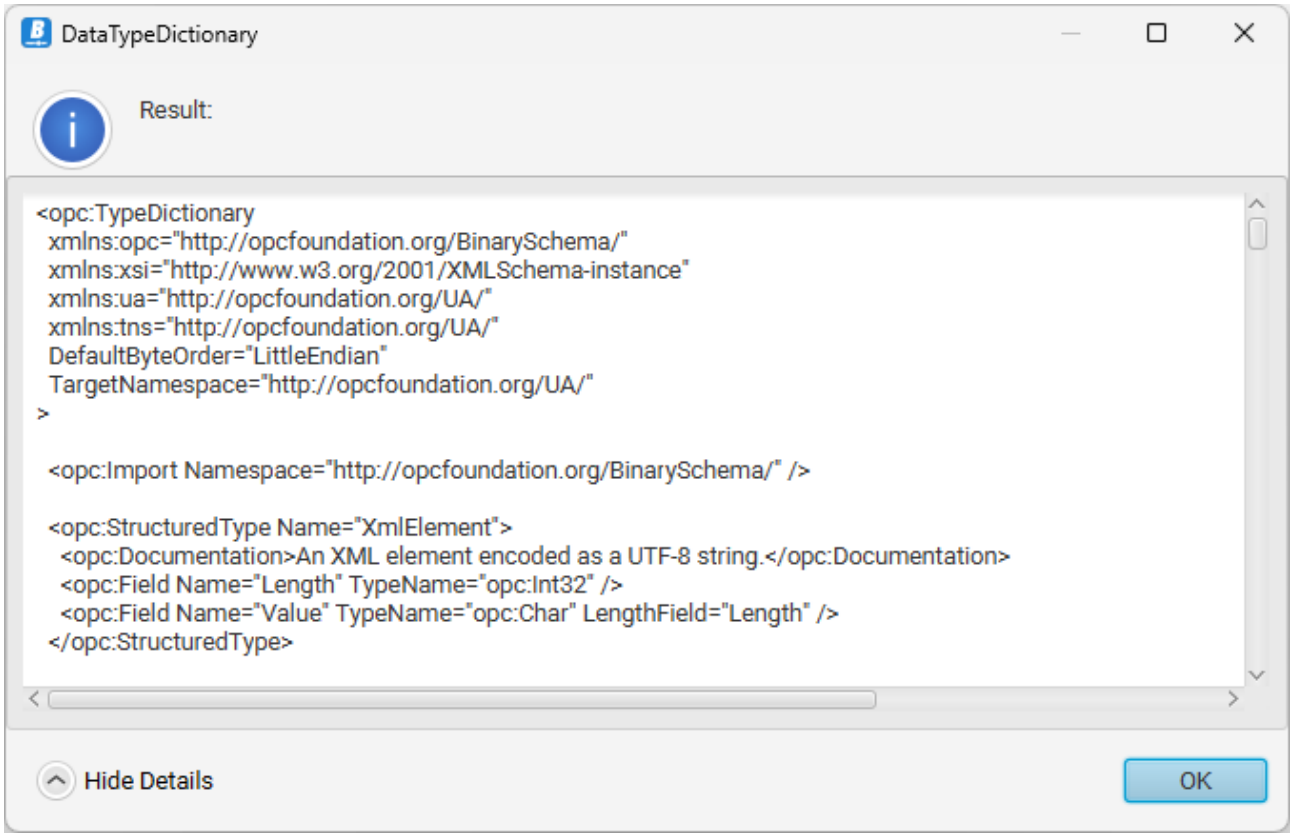

Figure 17. A dialog showing DataTypeDictionary of a DataTypeDictionaryType Node.

7.2.6. Writing to a Variable

A context menu option *Write Value...* is available for every Variable (see Figure 18). The latest version of the OPC UA Browser supports writing to more complex Variables, such as Structures, Arrays and abstract DataTypes.

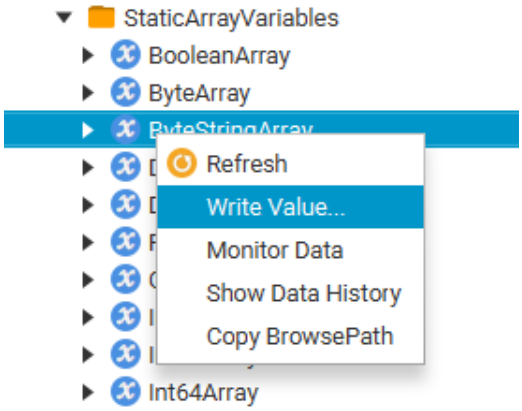

Figure 18. You can Write the Value of a Variable from the Address Space Browser context menu.

The dialog for writing to Variables looks like Figure 19. In the first cell you have the name of the Variable, the second cell defines the DataType of the Value and the cell for *Value* is where you will write or select the new value. If the DataType of a Variable is abstract, the dialog will give you a drop-down box of options to define a non-abstract DataType.

Figure 19. Write dialog

For array Variables, you can define values individually for every cell in the array. In Figure 20 the dialog lets the user select the DataType to be defined for every array cell. Next, the user can proceed to define the information for the cells. The required fields vary according to the DataType.

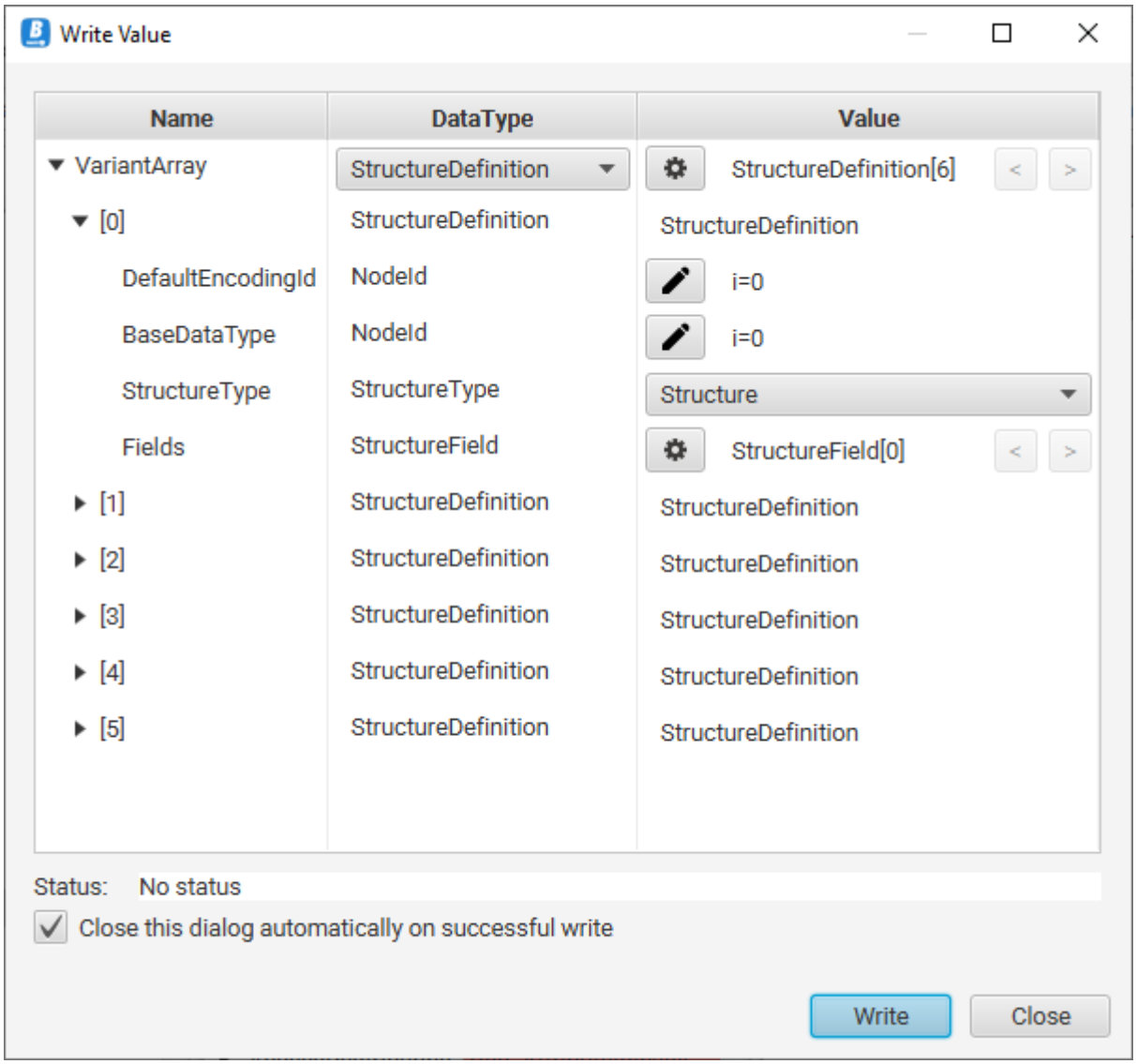

Figure 20. Write dialog for an array

7.2.7. Calling a Method

OPC UA includes a standard way for the Server to define Methods and their Input- and OutputArguments. This enables the Client to use a generic Method Call dialog that gets the information about the Arguments from the Server. The latest version of the OPC UA Browser supports a variety of DataTypes and ValueRanks to use as InputArguments.

A context menu option *Call Method...* is available for every Method (see Figure 21). In the dialog box, you can set input values and then press Call to execute the Method on the Server (see Figure 22). If the Call succeeds, you might see the return values in the lower table. Possible errors in parameters, for example, are shown in the Status field.

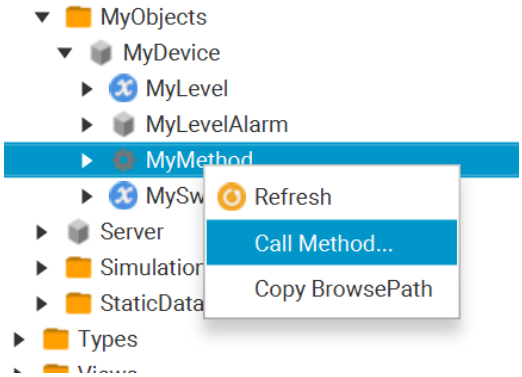

Figure 21. You can Call a Method from the Address Space Browser context menu.

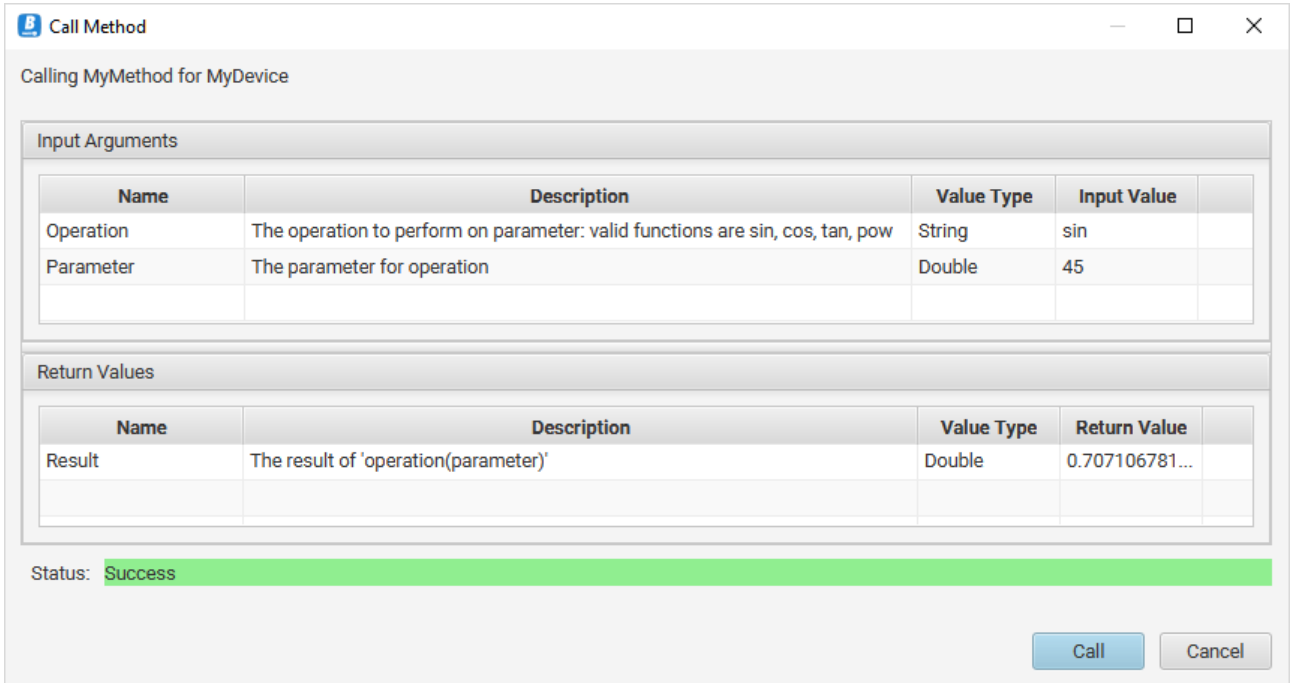

Figure 22. Calling a Method. MyMethod of the Prosys OPC UA Simulation Server enables you to define an operation (sin/cos/tan/pow) and a parameter (in degrees for the trigonometric functions).

7.3. Attributes and References View

The *Attributes and References View* is the default view displayed in the *Views Bar* on the right-hand side of the main window (see Figure 23). It consists of two tables: one representing the Attributes and another representing the References of the Node selected in the Address Space Browser.

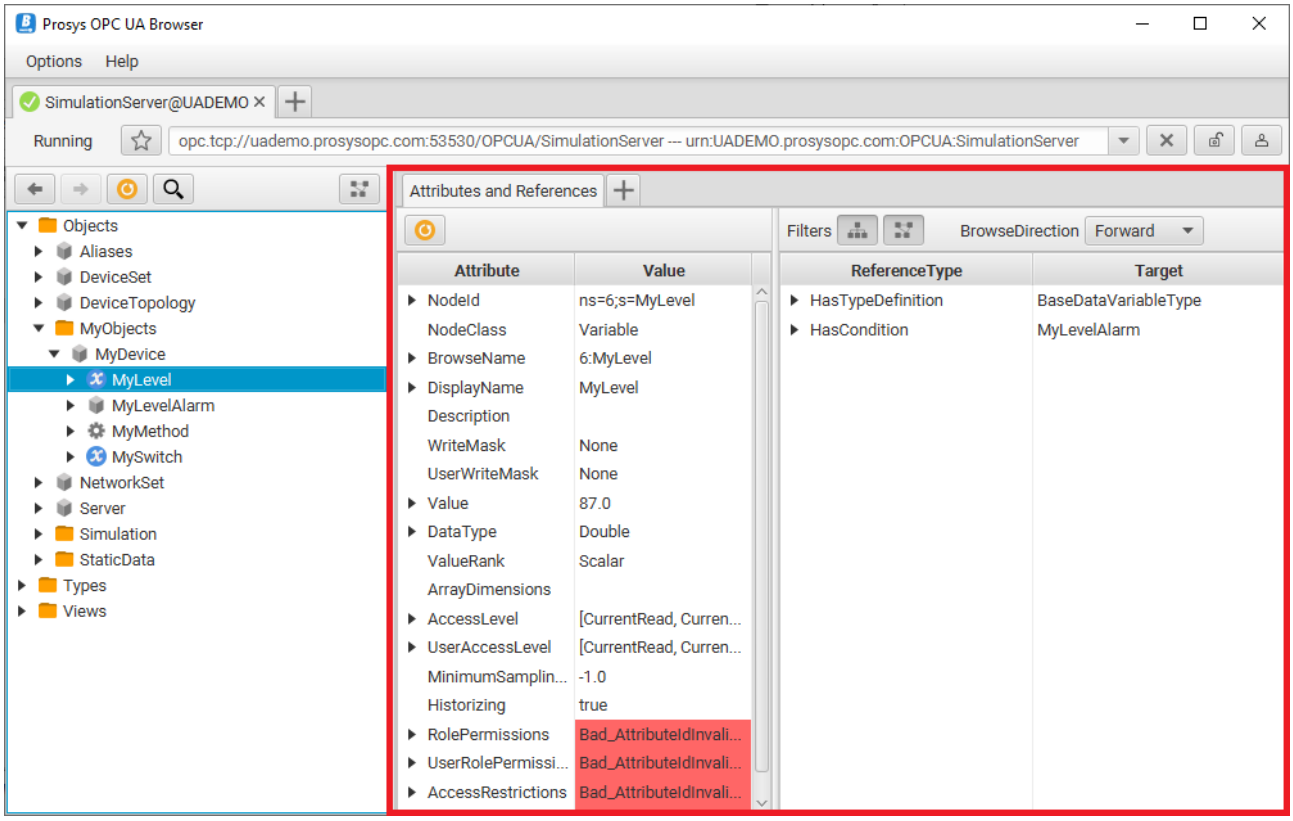

Figure 23. Attributes and References View showing the Attributes and References of a Variable Node.

The *Attribute Table* shows the Attributes of the Node and their respective values. The shown Attributes depend on the NodeClass of the selected Node.

The *Reference Table* shows the References of the Node: *ReferenceType* represents the type of the Reference, while Target represents the Node the Reference points to.

You can filter the Reference Table with the controls on top of the table. With the Filters buttons, you can select whether the table should show hierarchical, non-hierarchical, neither or both types of References. The *Browse Direction* selection has three options, to display only *Forward* or *Backward* References or *Both*.

Note that you can see complete Reference information of each Reference if you hover the mouse over the table or expand the individual rows.

7.4. Additional Views

Pressing the '+' button on the Views Bar allows opening additional Views (see Figure 2). If needed, multiple copies of the same type of View can be opened.

Pressing the 'X' button on a View in the Views Bar closes the View. Right-clicking on a View in the Views Bar opens the context menu for the View. There *Rename* action can be used to rename the View.

Attributes and References View can't be closed or renamed.

7.4.1. Save Views

When closing the Client/Server connection, opened Views are saved to a configuration file, if the user has enabled saving View configurations. This setting is available in *Preferences* > *Views* > *Save Views*. When this connection is opened at a later time, Views are loaded from the saved configuration file.

7.5. Data View

The Data View corresponds to an OPC UA Subscription, which can monitor data changes in the OPC UA Server (see Figure 24). The main component of Data View is the *Monitored Items Table* that represents Monitored Items of the Subscription as rows of the table.

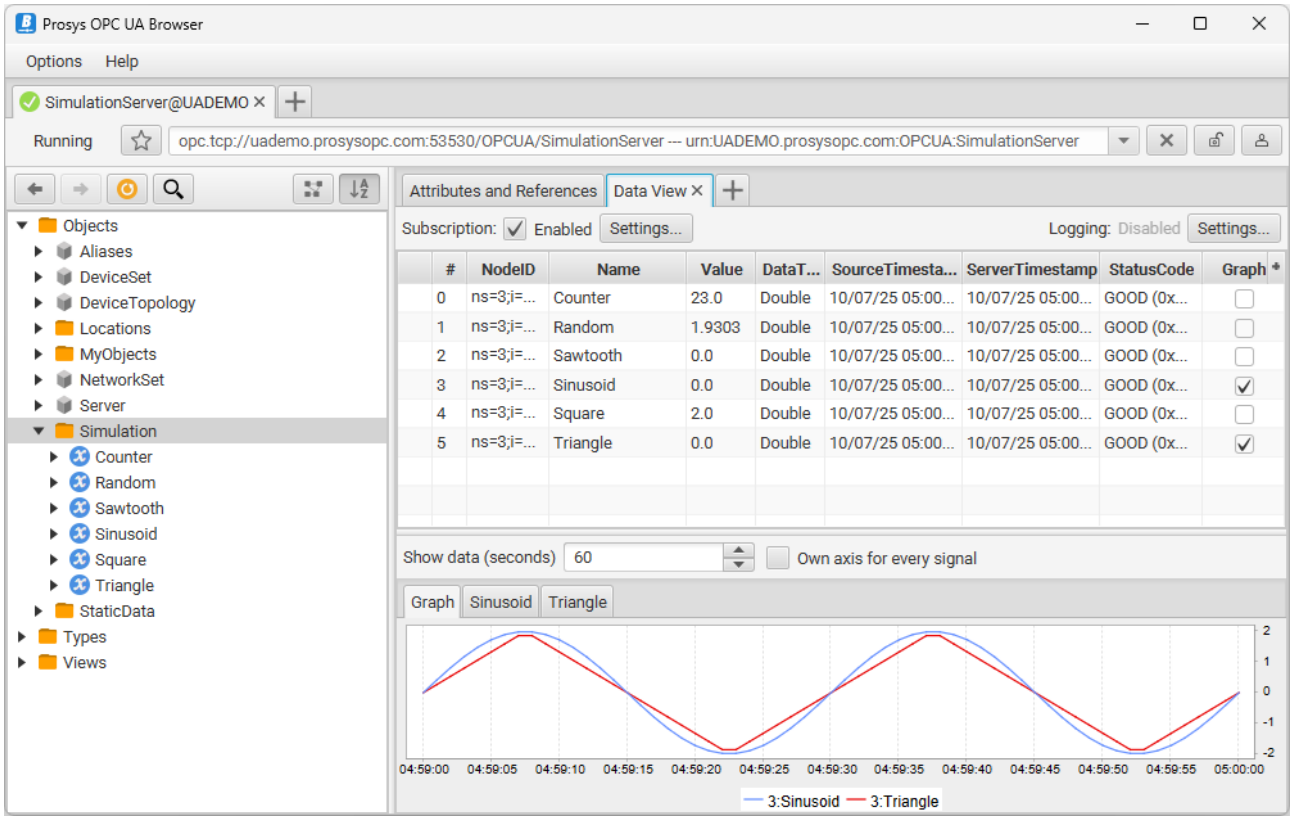

Figure 24. The Data View corresponds to an OPC UA Subscription. It contains a Monitored Items Table and Trend Graph, which is shown when the Graph column has been checked for at least on Monitored Item.

7.5.1. Adding Variables to Data View

You can add Variables to Data View in the following ways:

1. By dragging Variables from the Address Space Browser to the *Monitored Items Table* of the Data View.
2. By dragging Objects from the Address Space Browser to the *Monitored Items Table* of the Data View to add Variables below them. Optionally, Variables can be searched recursively for all Object Nodes below the dragged Object Nodes.
3. By using the *Monitor Data* action in the context menu of the Address Space Browser. This action is available for both Variable and Object Nodes.
4. By using the *Add Custom Node...* action in the context menu of the *Monitored Items Table*. See Searching for a Node for more information on how to use the dialog.

7.5.2. Data View Columns

OPC UA Browser creates a Monitored Item for each Variable that is added to the Subscription. By default, the following information is displayed for each Monitored Item (see Figure 24):

1. NodeId
2. Name
3. Value
4. DataType
5. SourceTimestamp
6. ServerTimestamp
7. StatusCode

NodeId uniquely identifies the Variable in the Server.

Name is the name of the Variable Node. Depending on the "Display Node Names As" preference (see Preferences), it is either the DisplayName or BrowseName with or without the Path to the actual owner of the Variable (the default is DisplayName without the Path).

Value field is simply the current value of the Node.

DataType represents the DataType of the Value.

SourceTimestamp is used to reflect the timestamp applied to a Variable Value by the data source. It should indicate the last change of the Value or StatusCode and be generated by the same physical clock.

ServerTimestamp is used to reflect when the Server received a Variable Value or knew it to be accurate.

StatusCode represents the quality of the Value. If the Value is not available due to any errors in the device or connection to the device, the status may be Bad with a more specific code explaining the reason for the problem. When the status is Bad, then Value may not be available.

By pressing the '+' button on the right side of the title row of the table, you can choose which columns are displayed. This enables hiding columns you don't need. You can also show two additional columns that are not shown by default: *ItemCode* and *Mode*.

ItemCode is the status of the Monitored Item itself. If creating the Monitored Item succeeded, the status is Good. If creating the Monitored Item failed, the status is Bad.

Mode is short for MonitoringMode and defines whether sampling of the Node and reporting of notifications is enabled in the Server.

You can use the checkboxes in the *Graph column* to select the Variables displayed in a trend graph. The length of the time axis is 60 seconds by default, and it can be changed between 1 and 9999 seconds. By default, the Variables share one common Y-axis. Use the *Own axis for every signal* option to create a separate Y-axis for each Variable.

The values that are displayed in the graph are also available on separate tabbed pages.

7.5.3. Subscription Settings

You can modify the *Subscription Enabled* parameter of the Subscription at the top of the screen. You can also open the *Subscription Settings* dialog (see Figure 25), which enables a more advanced configuration of standard OPC UA parameters.

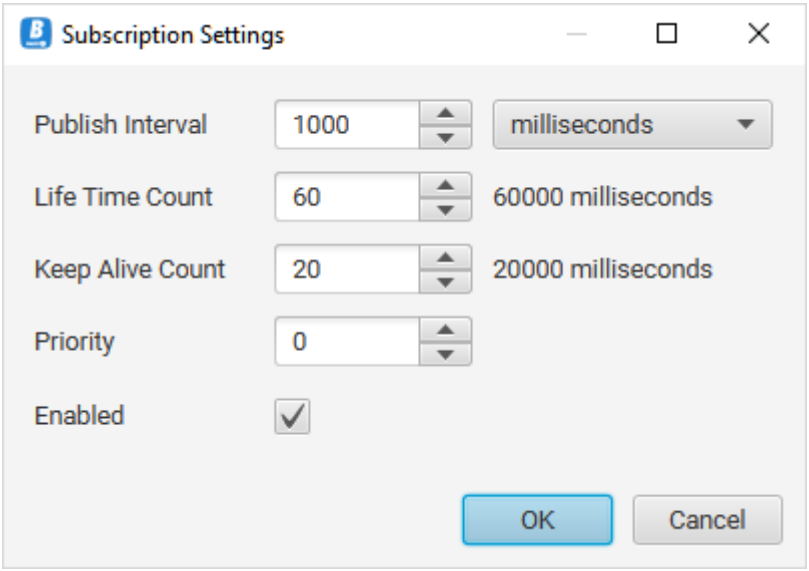

Figure 25. Subscription Settings dialog in Data View.

Publishing Interval defines how often the Server should send notification messages. This is the maximum rate that the Server may send notifications at. If there are no changes to send, the Server sends an empty *Keep Alive Message* after several intervals defined by the *Keep Alive Count*. If the Client does not receive anything from the Server for a long time, it may determine that the Subscription has timed out. Likewise, the Server is monitoring the lifetime of the Client application. If the Client does not send any acknowledgements back to the Server for the time corresponding to the *Life Time Count*, the Server may stop the Subscription and remove all the resources related to it. *Priority* may be used to define the relative importance of each Subscription: higher numbers correspond to a higher priority. Finally, each Subscription may be enabled or disabled without removing it from the Server.

7.5.4. Monitored Item Settings

The Monitored Items Table has a context menu, which enables a few actions. You can write a new value to a Variable, remove the selected items from the Subscription, or add custom Nodes.

MonitoringMode defines the current state of monitoring in the Server:

- *Disabled* means that the Server is not monitoring the Value
- *Sampling* means that the Server should sample the Variable but should not send any notifications
- *Reporting* means that the Server should sample and report changes to the Client via notifications

You can configure individual settings for each Monitored Item with *Monitored Item Settings...* action in context menu (see Figure 26).

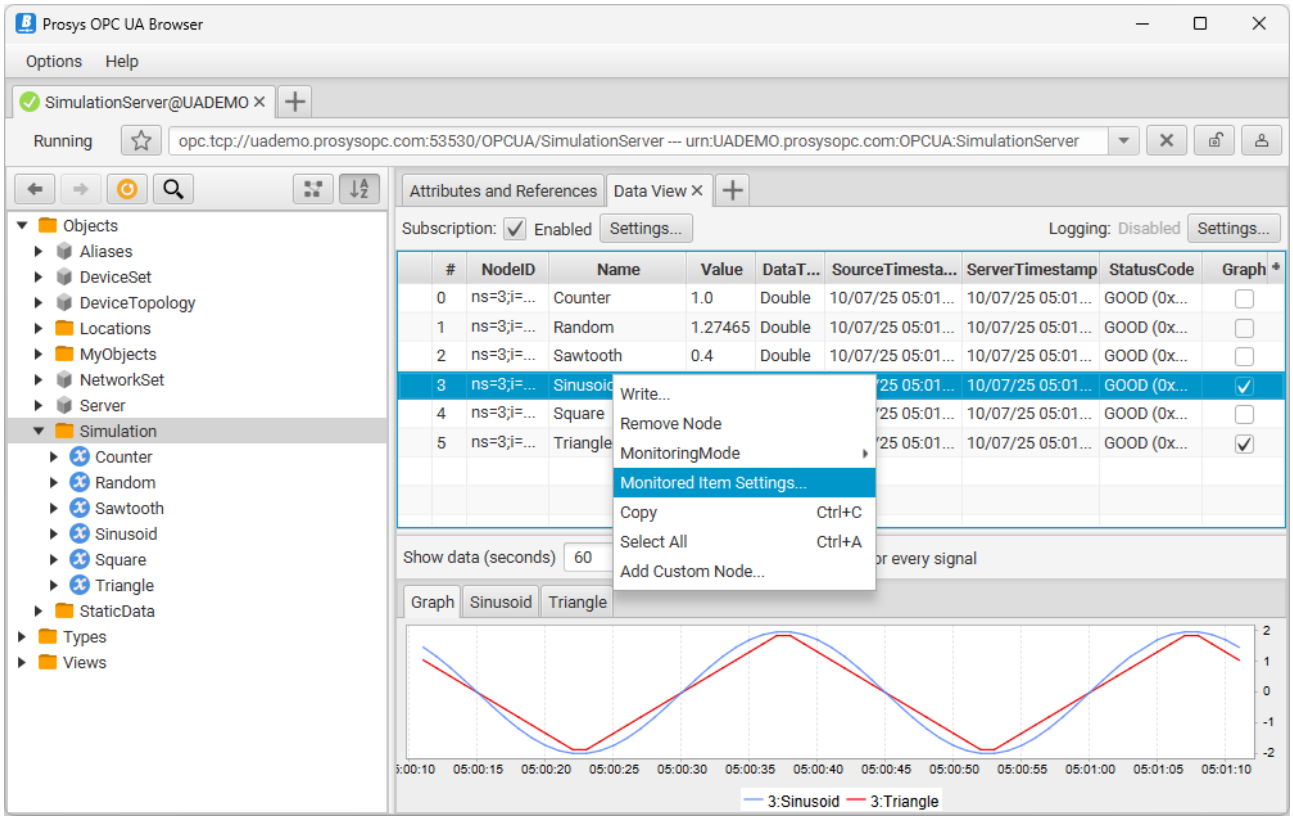

Figure 26. Context menu of the Monitored Items Table.

The settings for Monitored Items include the following (see Figure 27):

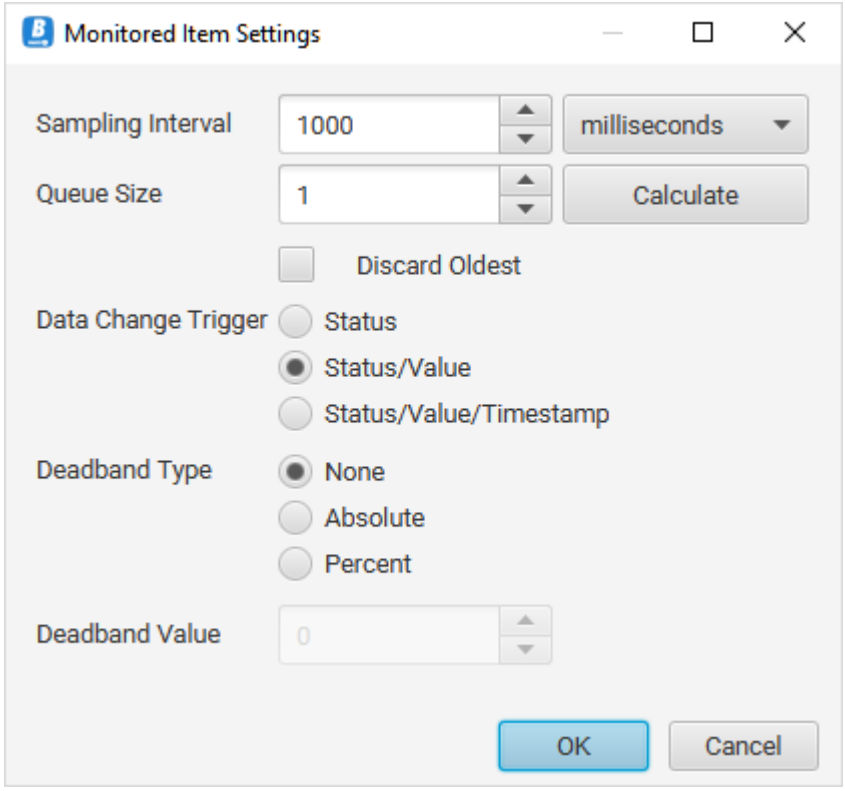

Figure 27. Monitored Item Settings dialog in Data View.

Sampling Interval is typically equal to the Publishing Interval of the Subscription. However, you may also use lower values to request faster sampling in the Server. In this case, you also need to configure *Queue Size* so that it can hold enough samples for the whole *Publishing Interval* (use the Calculate button to get

this filled with a valid number). Regardless, it may be better to configure more space in the queue. In case of communication problems, the Server can record more samples, even if it cannot send the notification back to the Client. If the sampling queue in the Server runs out, it throws old samples out of the queue. If Discard Oldest is set, the oldest values are removed; otherwise, the new values are discarded, except the latest (i.e., current) value, which is always available as the last value in the queue. These settings are only necessary if you need to record all changes; if you are only interested in the current value, the default sampling might be more suitable for you.

Data Change Trigger defines when the Server should record and send new samples. By default, any change in Status, Value or Timestamp triggers a change. Alternatively, you can request only Status changes or Status and Value changes.

If you configure a *Deadband Type* for the item, it requests the Server to only record new samples when the Value changes more than the defined *Deadband Value*. You have two ways to define it: either as an Absolute value or as a Percentage of the Variable range. The range may be defined for the Variables with a EURange Property. EURange is defined as part of AnalogItemType Variables.

7.5.5. Data Logging

You can enable data logging to CSV files from *Data Logging Settings* (see Figure 28).

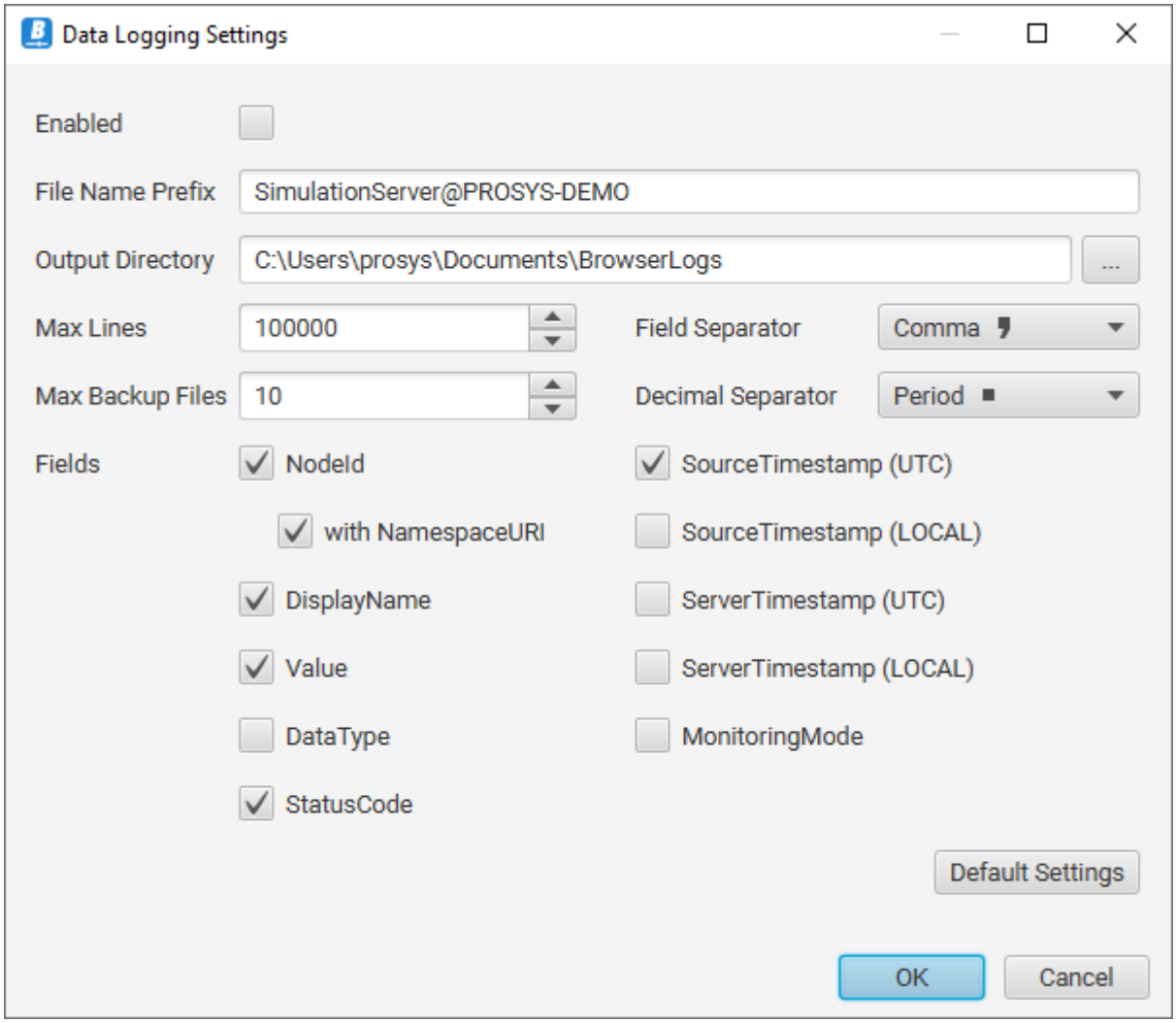

Figure 28. Data Logging Settings Dialog in Data View.

Data Logger subscribes to value changes of Nodes that are in *Monitored Items Table*. You can log files with a specific file name prefix to a specific folder by setting up the *File Name Prefix* and *Output Directory* settings respectively. Also, you can define *Max Lines* for a single file and *Max Backup Files* to control the number of files.

Fields are used for logging wanted data from Nodes. By default, Data Logger logs *NodeId with NamespaceUri*, *Display Name*, *Value*, *StatusCode* and *SourceTimestamp (UTC)*.

You can press the *Default Settings* button to apply the default settings.

7.6. 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 list of Variables and a set of tools to define the time interval to request the trends from the Server. The layout is illustrated in Figure 29.

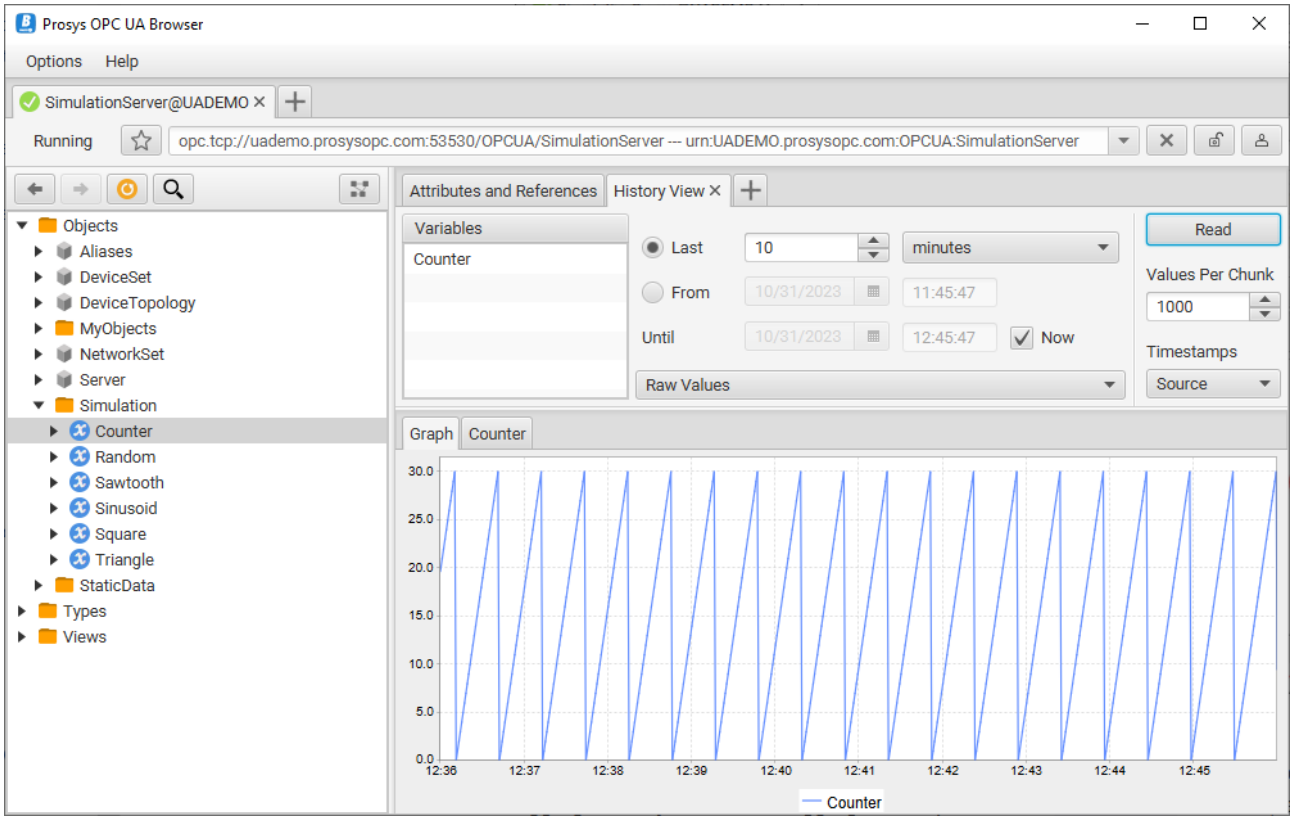

Figure 29. History View of the raw values of the Counter Variable for the Last 10 minutes until Now.

7.6.1. Adding Variables to History View

You can add Variables to History View in the following ways:

1. By dragging Variables from the Address Space Browser to the Variables list of the History View.
2. By dragging Objects from the Address Space Browser to the Variables list of the History View to add Variables below them. Optionally, Variables can be searched recursively for all Object Nodes below the dragged Object Nodes.
3. By using the *Show Data History* action in the context menu of the Address Space Browser. This action is available for both Variable and Object Nodes.
4. By using the *Add Custom Node...* action in the context menu of the Variables list. See Searching for a Node for more information on how to use the dialog.

7.6.2. Reading History

You can define the history interval in two alternative ways. The default is to define the *Last* X hours (or other time units) *Until* a specified time (or until *Now*, which is the default selection). The alternative way is to define the interval between two timestamps – From and To. See Figure 29 and Figure 30 respectively, for example.

When you click the *Read* button, the Client makes a History Read request to the Server, and displays the received data in a Graph. The values for each Variable are also available on separate tabbed pages.

7.6.3. Aggregate Calculation

By default, the Client reads raw values from the Server. Some Servers also support reading processed history in the form of Aggregates. If you select an Aggregate in the place of the Raw Values option, extra controls for specifying the processing interval and the Aggregate Configuration of the Aggregate appear. Additionally, some Servers do not correctly display their supported Aggregates, or in some cases, Servers acting as gateways do not have the information available. For this reason, there are two additional options as the last elements for the list: Standard Aggregate... and Custom Aggregate.... These can be used to open a separate dialog to select any Standard Aggregate defined in the OPC UA Specification or any NodeId via the Custom Aggregate.

Figure 30 shows an example of the Minimum value request for every 5 seconds. The button for opening the Aggregate Configuration dialog is highlighted with a red rectangle in the figure.

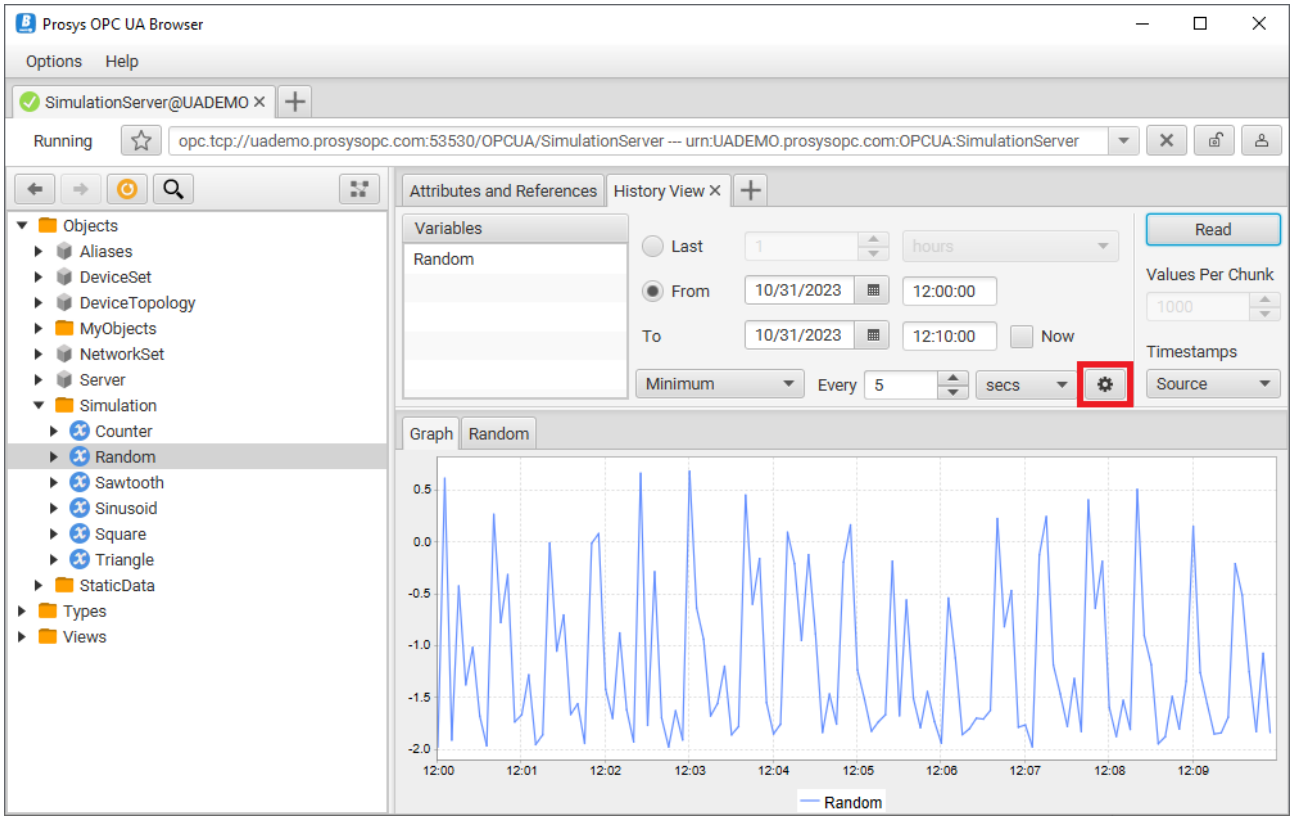

Figure 30. History View of the aggregated values of the Random Variable for a defined interval.

The Aggregate Configuration dialog (see Figure 31) is used to configure additional parameters for reading aggregated history.

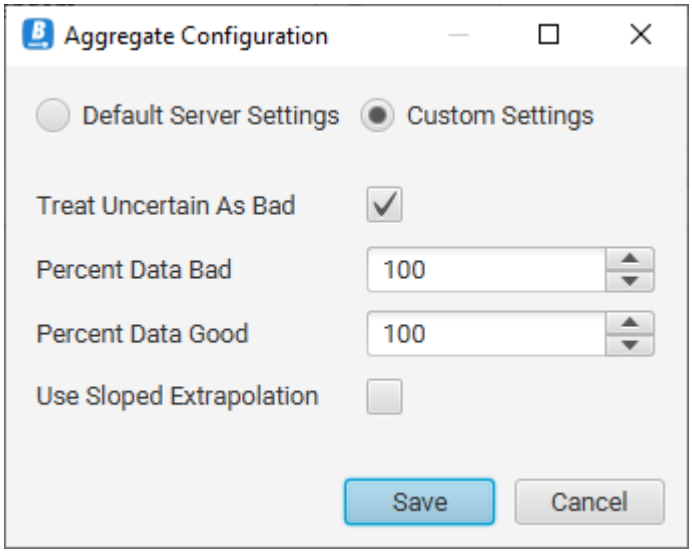

Figure 31. Aggregate Configuration dialog.

When *Default Server Settings* is selected, the Server's default values are used and any values specified in the Aggregation Configuration dialog are ignored. When *Custom Settings* is selected, the values specified in the Aggregate Configuration are used.

Treat Uncertain As Bad defines how the Server uses data with a StatusCode of Uncertain severity when performing aggregate calculations. When selected, Uncertain severity is considered equal to Bad severity. When not selected, Uncertain severity is considered equal to Good severity.

Percent Data Bad and *Percent Data Good* define the minimum percentages of Bad and Good data on the processing intervals for the StatusCode for the processing interval to be set to Bad or Good respectively. The sum of *Percent Data Bad* and *Percent Data Good* must be equal to or greater than 100.

Use Sloped Extrapolation defines how the Server interpolates data. When selected, Sloped Extrapolation is used. When not selected, Stepped Extrapolation is used.

7.7. Event View

The *Event View* is designed for receiving Event and Alarm notifications from the Server. The main component in the Event View is the *Event Table* in which the received Events are listed (see Figure 32). The Object monitored for Events is shown in the *Monitored Object* component. It is possible to monitor only one Object at a time.

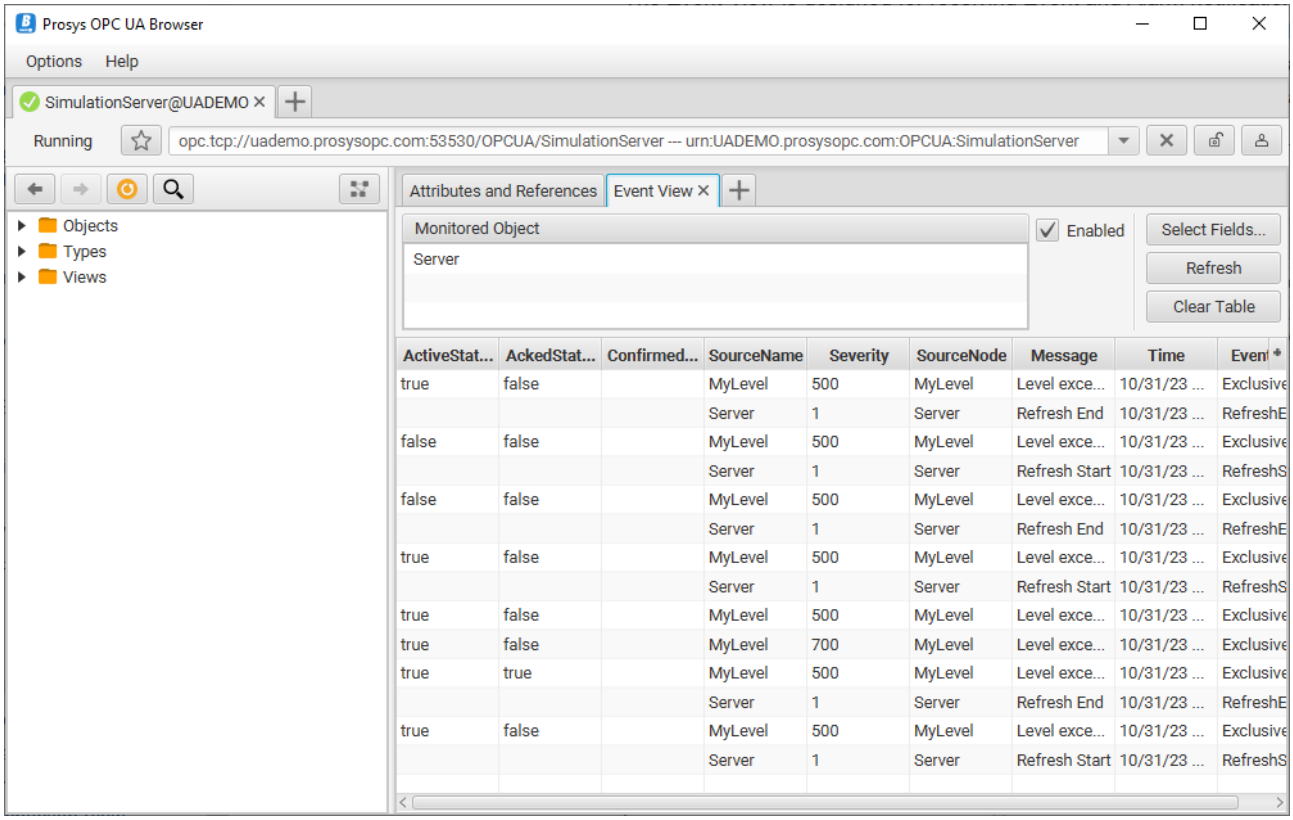

Figure 32. Event View with Server Node

7.7.1. Selecting the Monitored Object

You can select the monitored Object in Event View in the following ways:

1. By dragging an Object from the Address Space Browser to the *Monitored Object* component of the Event View.
2. By using the *Monitor Events* action in the context menu of the Address Space Browser.
3. By using the *Add Custom Node...* action in the context menu of the *Monitored Object* component. See Searching for a Node for more information on how to use the dialog.

7.7.2. Selecting Event Fields

By default, the Event Table contains nine fields:

1. ActiveState
2. AckedState
3. ConfirmedState
4. SourceName
5. Severity
6. SourceNode
7. Message
8. Time
9. EventType

ActiveState, *AckedState*, and *ConfirmedState* present whether the event or alarm is active, acknowledged, and confirmed, respectively.

SourceName and *SourceNode* identify the source of the Event in two different ways: as a name (usually as the *BrowseName* of the Node) and as a reference to the Node (as a *NodeId*, in case of an Object or Variable).

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 to normal Client severity levels.

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

Message shows the message text associated with the Event as defined by the Server.

Time is the timestamp of the Event as recorded by the Server.

EventType refers to the type of Event (as a NodeId).

Use the *Select Fields* function to define which Event fields are requested from the Server (see Figure 33). You can select from the Components and Properties of every EventType. The Events may contain much information, and the important fields depend on your application and the Events you expect to receive. The default fields are available for most EventTypes.

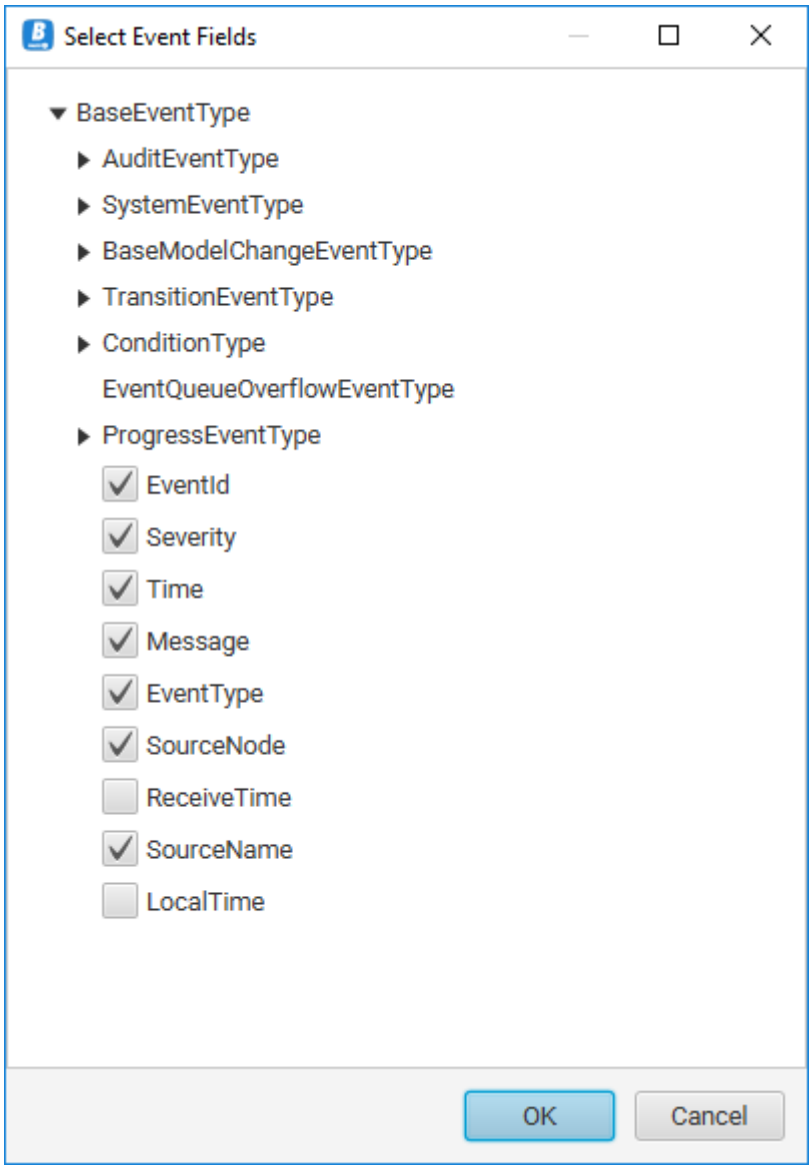

Figure 33. Selecting Event Fields.

Note that the fields are only applied to new Events: the Events already received from the Server do not get any new data. Instead, the Event History View may be able to provide that for you.

7.7.3. Basic Events and Conditions

There are two main varieties of Events: basic events and Conditions (all subtypes of ConditionType). The main difference is that Conditions have a state: for example, Alarms are active or inactive, unacknowledged or acknowledged, et cetera. The Condition types are often visible in the AddressSpace. The current state can also be read and monitored via the components of the Condition Object (for example, ActiveState and AcknowledgedState Variables). Whenever the Condition changes its state (for example, from Inactive to Active), the Server sends an Event, which can be considered a state change notification.

The EventTypes that are not Conditions can be used to trigger basic notifications: configuration change, system events, state transitions, et cetera.

7.7.4. Condition Refresh

The Refresh button in the Event View makes a ConditionRefresh Service call to the Server. It requests the Server to resend the current state of all active Alarms to the Client. Prosys OPC UA Browser makes an automatic ConditionRefresh to the Server when the Event View is initialized or the monitored Object is changed.

7.8. Event History View

OPC UA Objects, which define *HistoryRead* in their *EventNotifier* Attribute, may be requested for Event history.

7.8.1. Selecting the Object

You can select the Object for reading Event history in the Event History View in the following ways:

1. By dragging an Object from the Address Space Browser to the *Object* component of the Event History View.
2. By using the *Show Event History* action in the context menu of the Address Space Browser.
3. By using the *Add Custom Node...* action in the context menu of the *Monitored Object* component. See Searching for a Node for more information on how to use the dialog.

7.8.2. Reading History

Select a suitable time interval (see History View for details) and *Read* the history. The results are displayed in a table (see Figure 34).

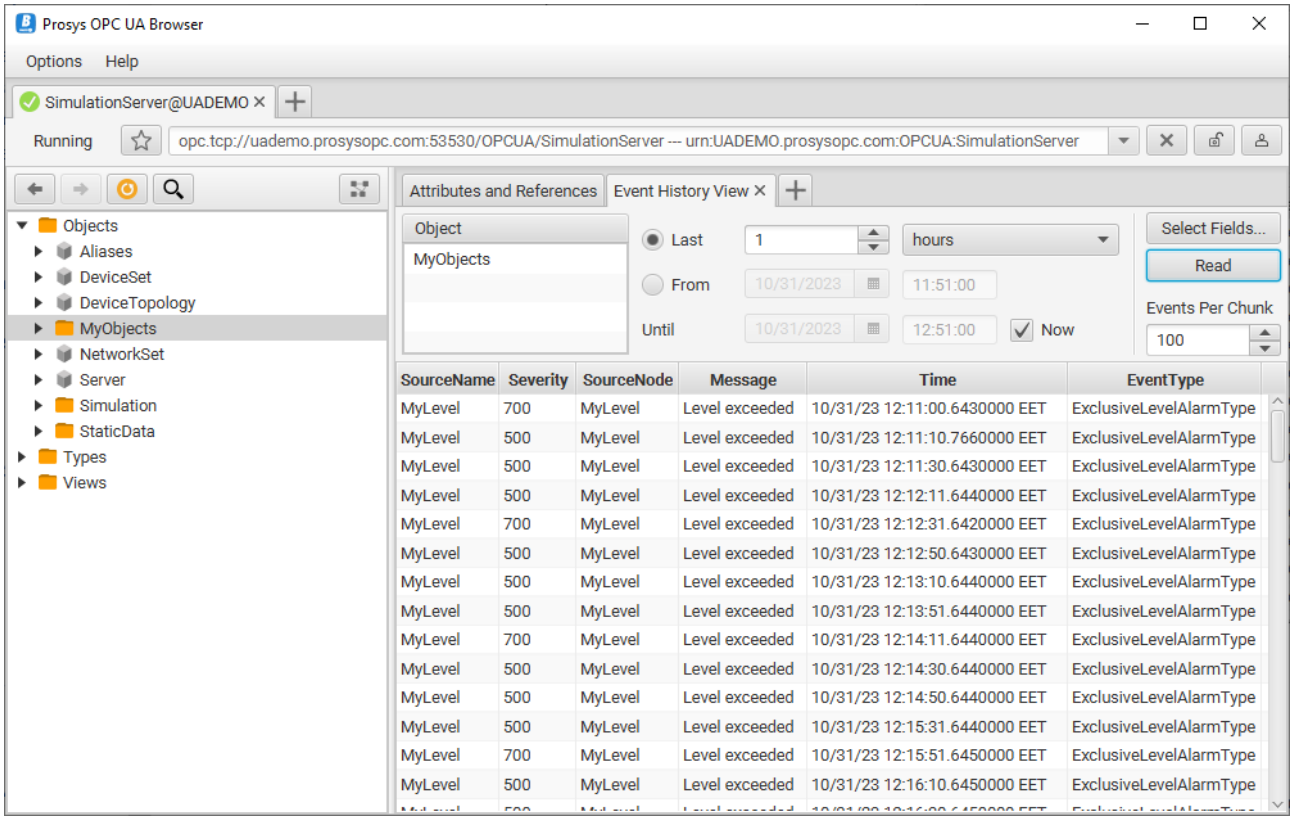

Figure 34. Requesting Event History for MyObjects of Prosys OPC UA Simulation Server.

7.8.3. Selecting Event Fields

You may also use the Select Fields... option to define which Event fields you want to read. See more about the Event fields in Event View.

7.9. Image View

In the *Image View*, you can view Image Nodes. Drag a Variable, whose DataType is one of the Image types, into the view, and the Variable's Value is displayed as an image. The Image View also makes a Subscription to the Variable, similar to the Data View subscriptions. Thus, if the image changes (i.e., snapshots from a camera), you may see the changes as they appear.

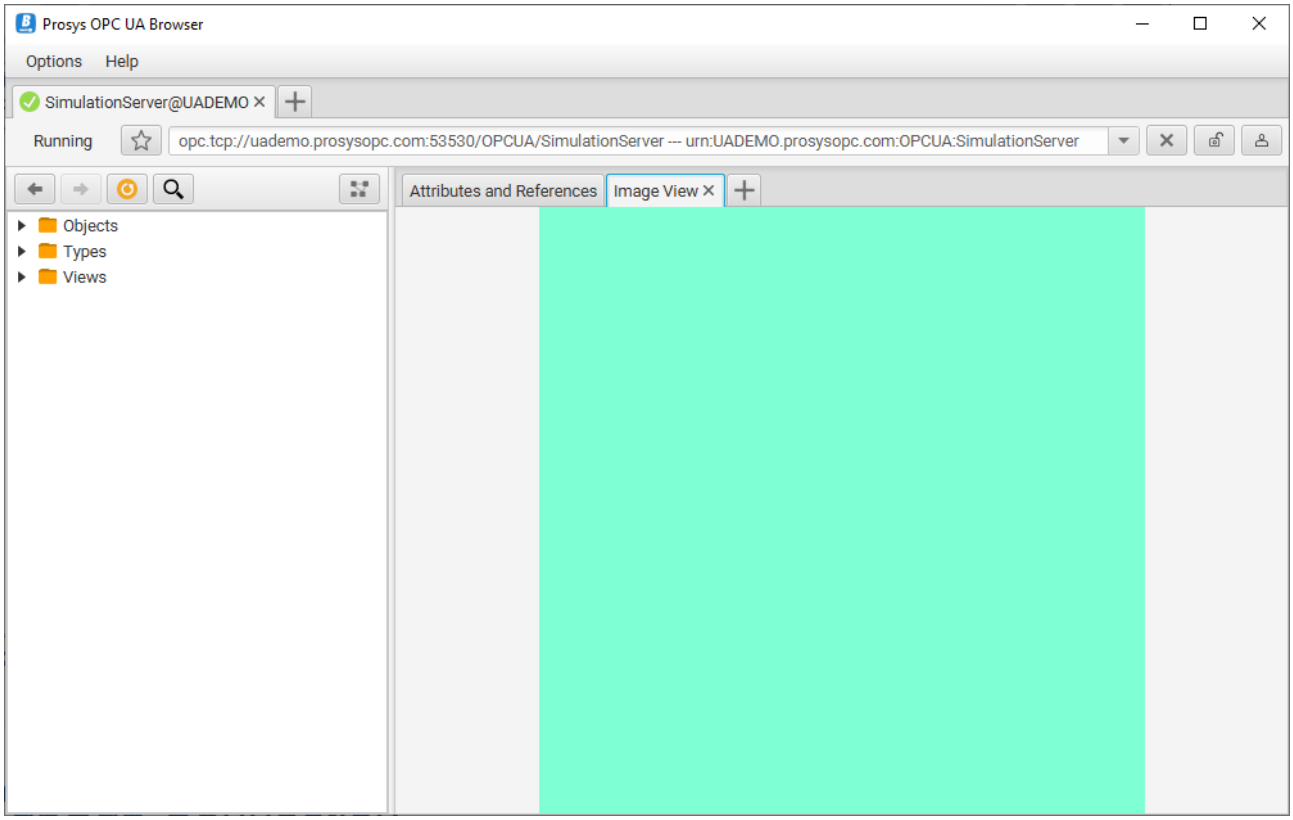

Figure 35. Image View.

7.10. NodeSet Export View (Professional Edition only)

In the *NodeSet Export View*, you can export Namespaces of the Server to NodeSet XML files.

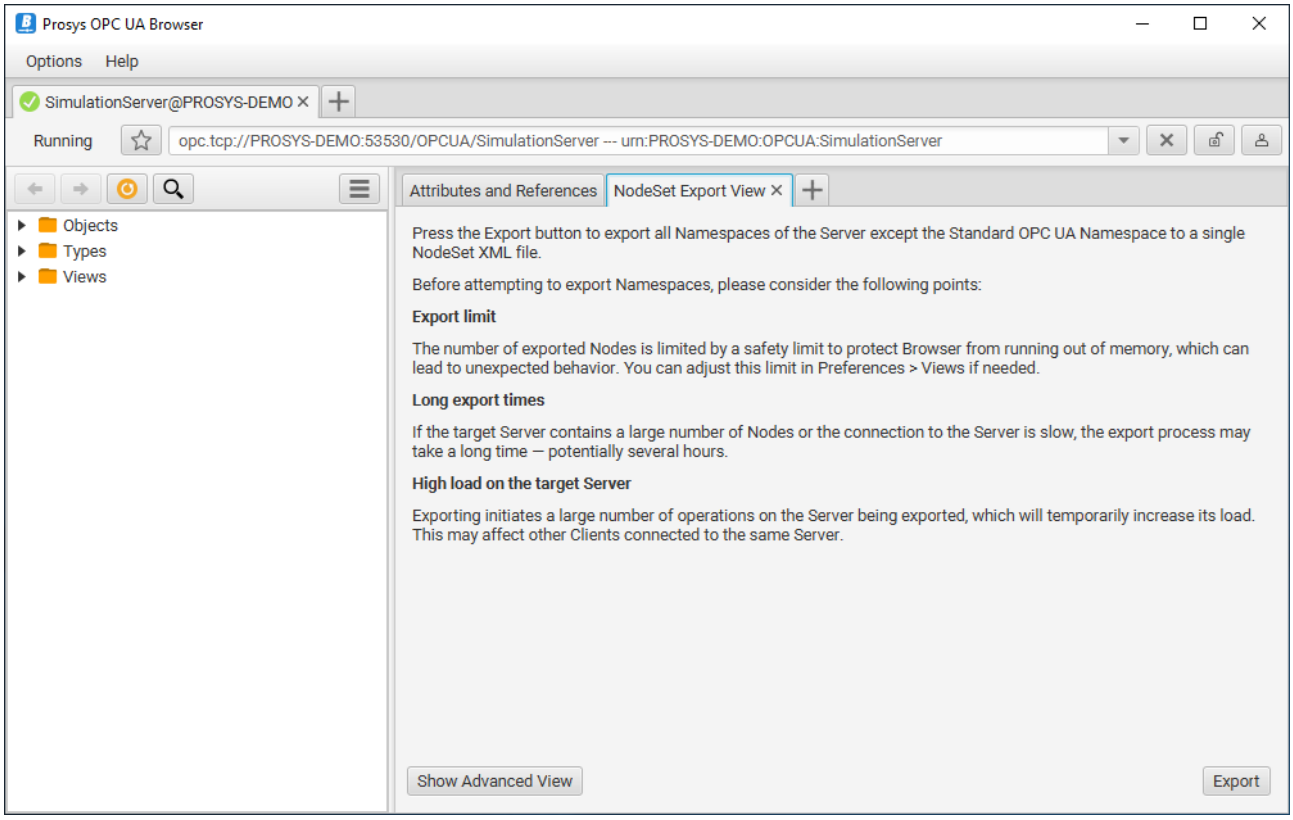

Figure 36. Basic NodeSet Export View.

7.10.1. NodeSets

NodeSets are XML files that represent Nodes and References using the Information Model XML Schema defined in the OPC UA Specification. The schema represents Nodes using different elements based on their NodeClass, such as representing Object Nodes with UAObject elements. References are represented as child elements of the elements representing the Nodes.

NodeIds in NodeSets are always defined with NamespaceIndex (e.g. 'ns=1;i=1001'), where the index refers to the position of the namespace within the NamespaceUris element at the beginning of the NodeSet file. So, the indices in NodeSets are usually different than in the actual Servers.

The index of the standard OPC UA Namespace is always zero, and it's not included in the NamespaceUris element. The index of the Namespace defined in the NodeSet itself is usually 1 and the other Namespaces in NamespaceUris correspond to indices from 2 on. Typically, a NodeSet defines Nodes for one Namespace only. So, usually indices from 2 on refer to Nodes in other NodeSets. The dependencies are then defined by the elements in the NamespaceUris.

The chain of dependencies can be longer without any complications, as long as it's unidirectional. If there are bidirectional dependencies or loops between the Namespaces, then all of them must be defined in one NodeSet file. Although References between Nodes can always be considered bidirectional, it is enough to define them only once in either Node (and NodeSet). This avoids bidirectional references between the NodeSets.

7.10.2. Use cases of NodeSets

NodeSets can be used to generate code with code generators such as Code Generator of Prosys OPC UA SDK for Java or Unified Automation UaModeler. This code can be used in both Client and Server applications to provide a convenient API for operating with ObjectTypes, VariableTypes and DataTypes defined in the NodeSet instead of relying on the generic API methods.

NodeSets can be imported directly into OPC UA Servers that support importing NodeSets such as Prosys OPC UA Simulation Server Professional Edition. The Nodes and References defined in the NodeSet are added to the AddressSpace of the Server when importing a NodeSet.

The indices of NamespaceUris in the NodeSet are not preserved when importing it to an OPC UA Server. Instead, the Server will assign indices for the new NamespaceUris in the NodeSet.

When a NodeSet is dependent on another NodeSet, it is typically required to import the other NodeSet to the OPC UA Server first.

7.10.3. Exporting Namespaces to NodeSets

After reading the displayed instructions carefully, click on the Export button to export all Namespaces except for the standard OPC UA Namespace to a single NodeSet XML file (see Figure 36). This process can take a long time depending on the connection speed and the number of Nodes in the Server's AddressSpace. The export operation does not block Browser's user interface and it can be used as usual while the export is in progress.

If the export is taking too long, it can be canceled by pressing the Cancel button displayed when export is in progress (see Figure 37).

Exporting Namespaces as NodeSets is a resource intensive operation for both Browser and the Server. It requires browsing through the entire AddressSpace to find all Nodes and References. Depending on the number of Nodes in the AddressSpace, this operation can take a very long time and consume a lot of memory.

By default, this operation is interrupted if there are more than 100 000 Nodes in the AddressSpace. This limit can be increased in *Preferences* > *Views* > *Max Nodes In NodeSet Export*, but it is possible for Browser to run out of memory on higher limits and behave in an unexpected manner.

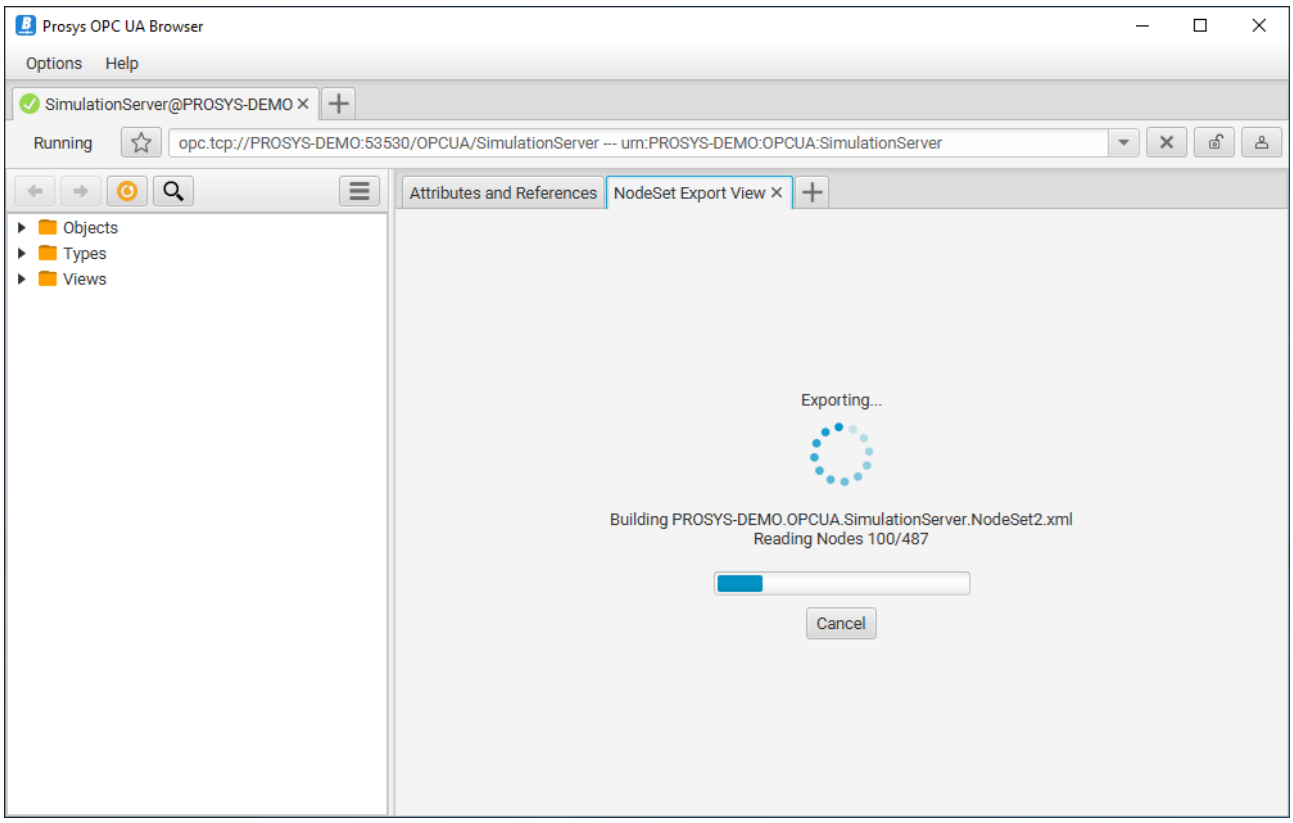

Figure 37. NodeSet Export View with export in progress.

7.10.4. Advanced NodeSet Export View

The Advanced NodeSet Export View can be opened by clicking on the "Show Advanced View" button. The Advanced View enables choosing which Namespaces to export and how to export them (see Figure 38).

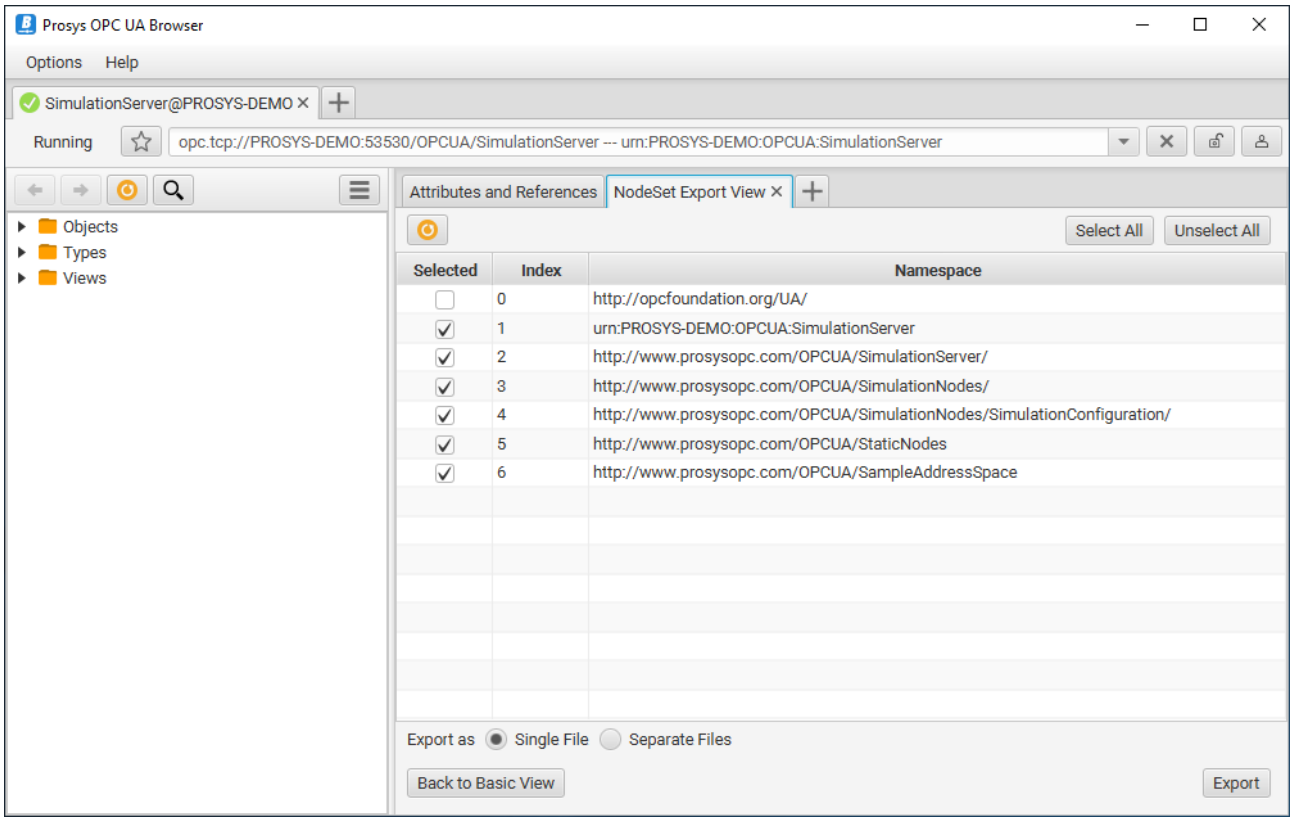

Figure 38. Advanced NodeSet Export View.

The Export column is used to select which Namespaces shall be exported to NodeSet files. By default, all Namespaces except the standard OPC UA Namespace are selected for exporting.

There are two different exporting modes available:

Export As Single File exports the selected Namespaces to a single NodeSet file and saves the file using the selected file name.

Export As Separate Files exports the selected Namespaces to separate NodeSet files and saves the files to the selected folder.

After selecting the Namespaces and the exporting mode, click the Export button to begin exporting the Namespaces.

7.11. CSV Node Export View

In the *CSV Node Export View*, you can export Nodes of the Server to a CSV file. Exporting is performed by recursively browsing the Objects Folder of the Server for Nodes. The Nodes matching the configured filters are exported to the CSV file.

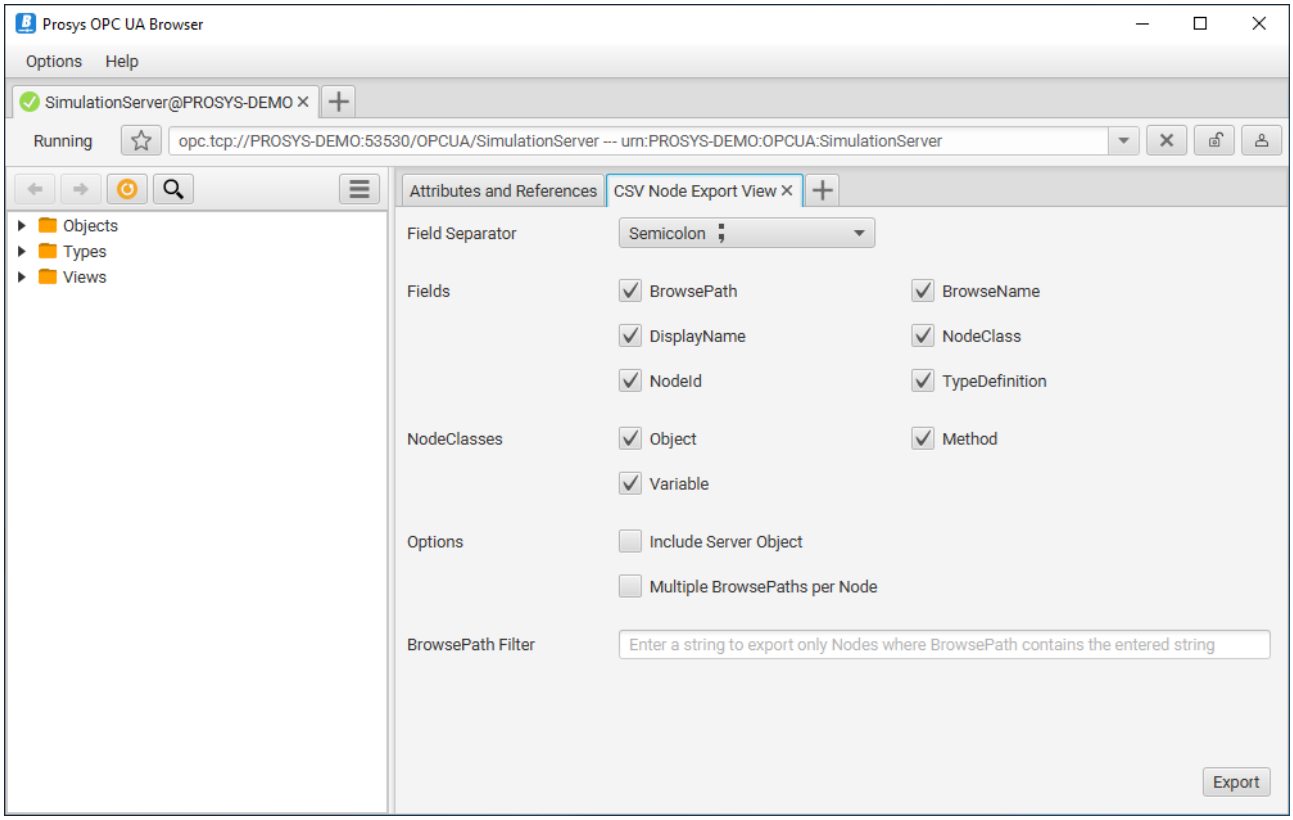

Figure 39. CSV Node Export View.

The settings for CSV Node exporting consist of the following parameters (see Figure 39):

Field Separator defines the field separator for the CSV file. The available options are comma, semicolon and tab character.

Fields define the Node information that will be generated to the CSV file.

NodeClasses define which Nodes are exported to the CSV file based on their NodeClass.

Include Server Object defines whether or not the Server Object is included when browsing the AddressSpace for Nodes and References. Typically the Server Object contains information on Server capabilities and diagnostic information. Thus, exporting it is usually not useful.

Multiple BrowsePaths per Node defines whether or not multiple BrowsePaths are exported per Node. When this option is selected, a safety limit of 10 BrowsePaths per Node is used to avoid infinite loops.

BrowsePath Filter can be used to export only Nodes with BrowsePaths that contain the given text.

After configuring the parameters, press the *Export* button to select the file for exporting. This process can take a long time depending on the connection speed and the number of Nodes in the Server's Objects Folder. The export operation does not block Browser's user interface and it can be used as usual while the export is in progress.

If the export is taking too long, it can be canceled by pressing the Cancel button displayed when export is in progress (see Figure 40).

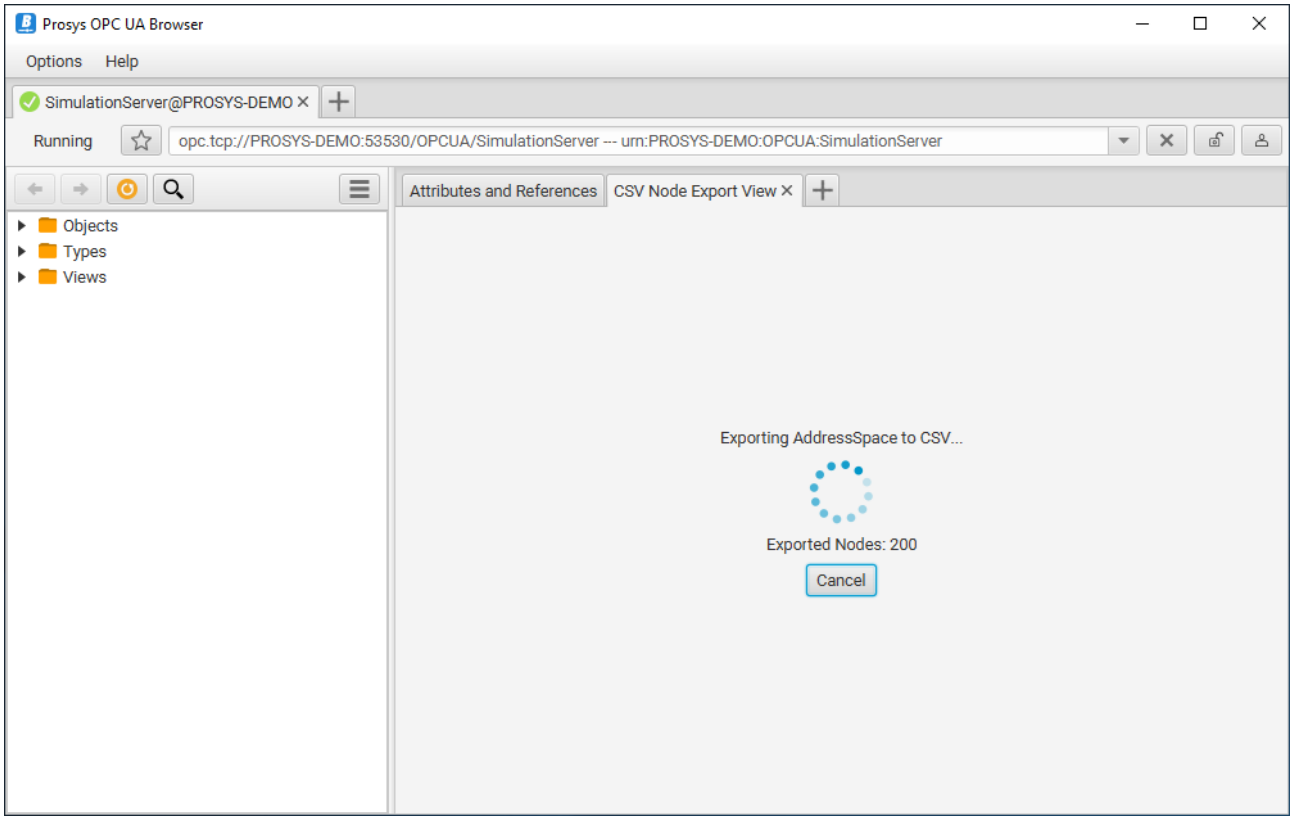

Figure 40. CSV Node Export View with export in progress.

The number of exported Nodes is limited to 25 for the Free Edition. There is a safety limit of 10 million Nodes in the Professional Edition.

8. PubSub Connection

In addition to an OPC UA Client, you can use Browser as an OPC UA PubSub Subscriber and connect to a PubSub Network.

8.1. Connecting to a PubSub Network

To connect to a PubSub Network, you need to know whether the PubSub Network uses MQTT or UDP. If you are using Prosys OPC UA Simulation Server as a Publisher, you can check the configuration in its PubSub View. You can then connect to the Network accordingly.

When a successful connection has been established, the Browser will save your new connection settings to the *Quick Links* for easier access in future.

8.1.1. MQTT Network

For an MQTT Network, you will need to connect to the MQTT Broker, whose address is defined in the format

```
mqtt://<hostname>:<port>
```

or

```
mqtts://<hostname>:<port>
```

This should match the Connection Address used in your Publisher configuration (in Simulation Server, for example). Type it to the address bar at the top and press Enter or click the Connect button (→).

If the Broker is available, you can then define the MQTT Topic Filter and the user credentials that you wish to use for the connection (see Figure 41). You can use the wildcard character "#" as the Topic Filter to subscribe to everything. Note however, that some brokers may not accept a subscription to everything, in which case you will need to know the topic tree that you need to subscribe to.

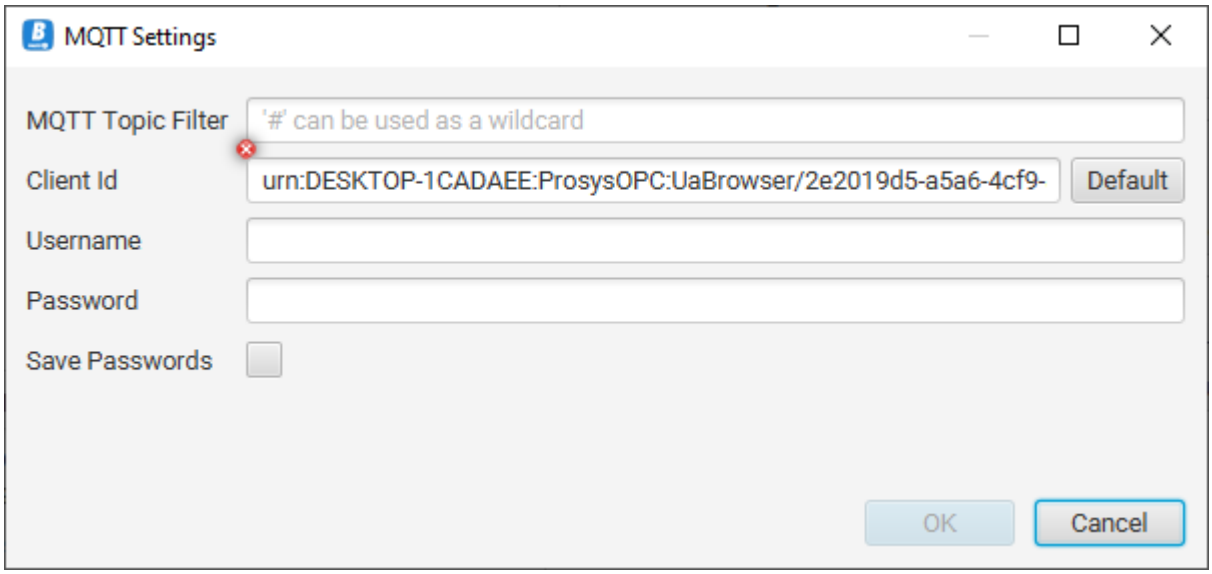

Figure 41. Dialog for the MQTT connection settings.

MQTT Topic Filter can follow the exact path of the topic that you wish to subscribe to. For example:

```
prosysopc/simserver/data
```

Or you can use wildcard characters to filter complete branches of the topic tree. For example:

```
prosysopc/#
prosysopc/+/data
```

'#' matches everything under the tree branch, whereas '+' matches with anything on that exact level in the tree.

Once you are ready, click OK to open the connection. If the MQTT Broker is available and it accepts your credentials, you will be connected. Otherwise, you should see an error that indicates why connecting failed.

If you choose Save Passwords, the password used for the connection is stored in the *conf.xml* in the application settings folder in encrypted format. However, you should be careful with the file and in case of doubt, you can choose not to save the password in the configuration.

MQTT is unencrypted communication and the password is also sent clear-text, so do not use sensitive passwords in your MQTT Broker configuration. Whenever possible, enable MQTTS in the broker instead. MQTTS uses an encrypted connection over TLS/SSL and the passwords are not visible to third parties. MQTTS also supports client authentication with X.509 certificates.

8.1.2. UDP Network

The Connection Address for OPC UA UDP can be either a multicast or a unicast address. In any case, it should match the address defined in the respective Publisher configuration (in Prosys OPC UA Simulation Server, for example), in order to receive messages from them. Type the address to the address bar at the top and press Enter or click the Connect button (→).

UDP Multicast

Multicast addresses are used to publish the messages to several subscribers in the network and you would typically want to use those. The Connection Address for multicast should be in the format

```
opc.udp://<address>[:<port>]
```

where address is one of the standard multicast addresses in the range 224.0.0.0 ~ 239.255.255.255 (224.0.0.0/4) as defined in RFC 3171.

The OPC UA specification defines a special address, opc.udp://224.0.2.14 for *OPC UA discovery purposes*, so avoid that for normal connections.

UDP Unicast

UDP unicast defines a point-to-point connection between two computers. In this case the connection address should be

```
opc.udp://localhost[:<port>]
```

The default port number for both multicast and unicast communication with OPC UA UDP is 4840.

Network Interface

You must also choose the Network Interface that you want to listen to (see Figure 42).

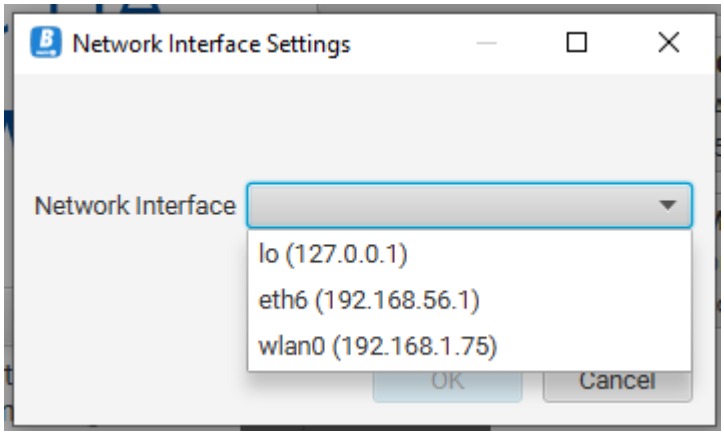

Figure 42. Dialog for setting the Network Interface for the UDP connection.

The Network Interface is currently not used for unicast addresses. Instead, Browser is always listening to all interfaces.

8.2. PubSub Connection View

The Browser will show your subscription data in a PubSub Connection View. From Figure 43 you can see that the title of the tab is named after the Network type and broker (MQTT) or host (UDP). The address bar shows the Connection Address as well as the Topic you have subscribed to.

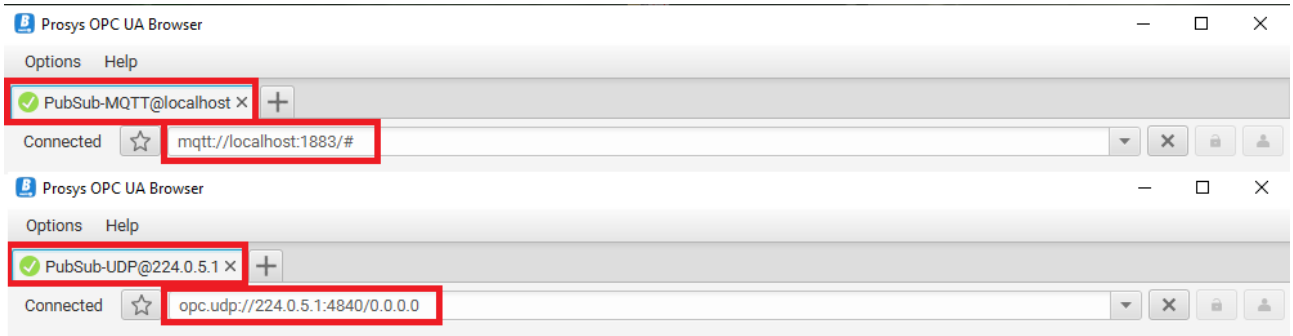

Figure 43. The title and address bar are named to describe the PubSub connection and subscription.

8.2.1. PubSub View

The new PubSub Connection View itself contains two tabs: PubSub and Log. The *PubSub* View is divided into two parts: a list of DataSetWriters received on the left-hand side and two tabs for the data to be presented on the right-hand side.

The Browser contains a logic to determine if the messages contain Variable Data or Events, so it can separate them into the two Views on the right: Data View and Event View. Figure 44 shows how the DataSetWriters are separated in both MQTT and UDP Networks. The different icons are used to identify them. If, however, the Browser guessed wrong, you can right-click the DataSetWriter on the list and select Monitor as events or Monitor as data to have it moved to the correct group.

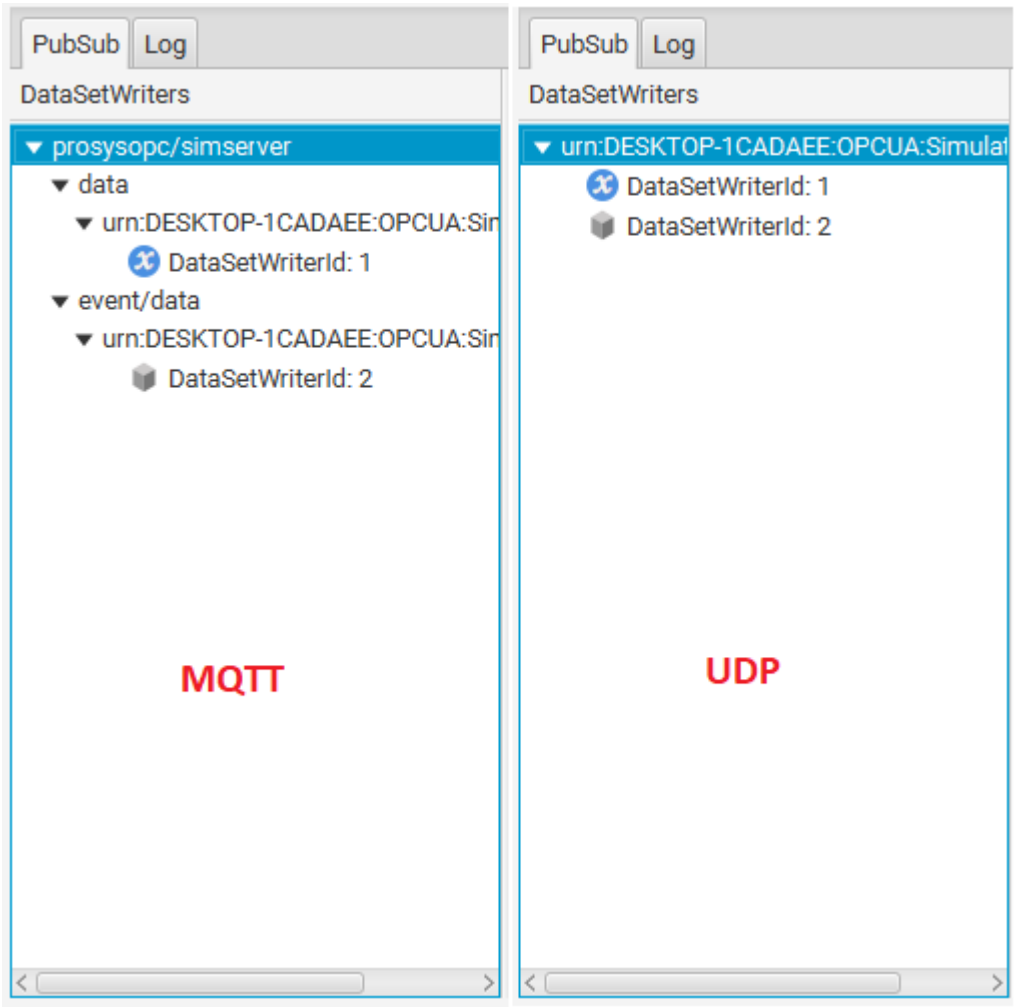

Figure 44. The list of DataSetWriters (MQTT on the left, UDP on the right).

In the MQTT case, the DataSetWriters are also classified into a hierarchy corresponding to the MQTT Topics that they are publishing to.

You can also use the DataSetWriter list to filter the content of the Data View and Event View. If you select only one DataSetWriter, it determines the view on the right, based on the writer type. If you use Clear Selection (on the right click menu) to select all or use the MQTT Topic Tree to filter a part of the tree, you can switch between the views and see both data and Events that are received.

PubSub Data View

The *PubSub Data View* (Figure 45) represents Variable data received by the Subscriber in a similar way as Data View shows the values received by the Client/Server Subscriptions. There is a separate row for each DataSet Field with their latest values and timestamps. If you wish to see how the values are changing over time, you can select the *Graph* option for them, in which case you will see a trend chart and a table of value changes for each.

Note that the actual data that can be shown depends on the configuration of the Publisher. In case you don't see any timestamps, for example, check that the Publisher is configured to send them. Also note that data is identified by Field Names, which are also configured in the Publisher.

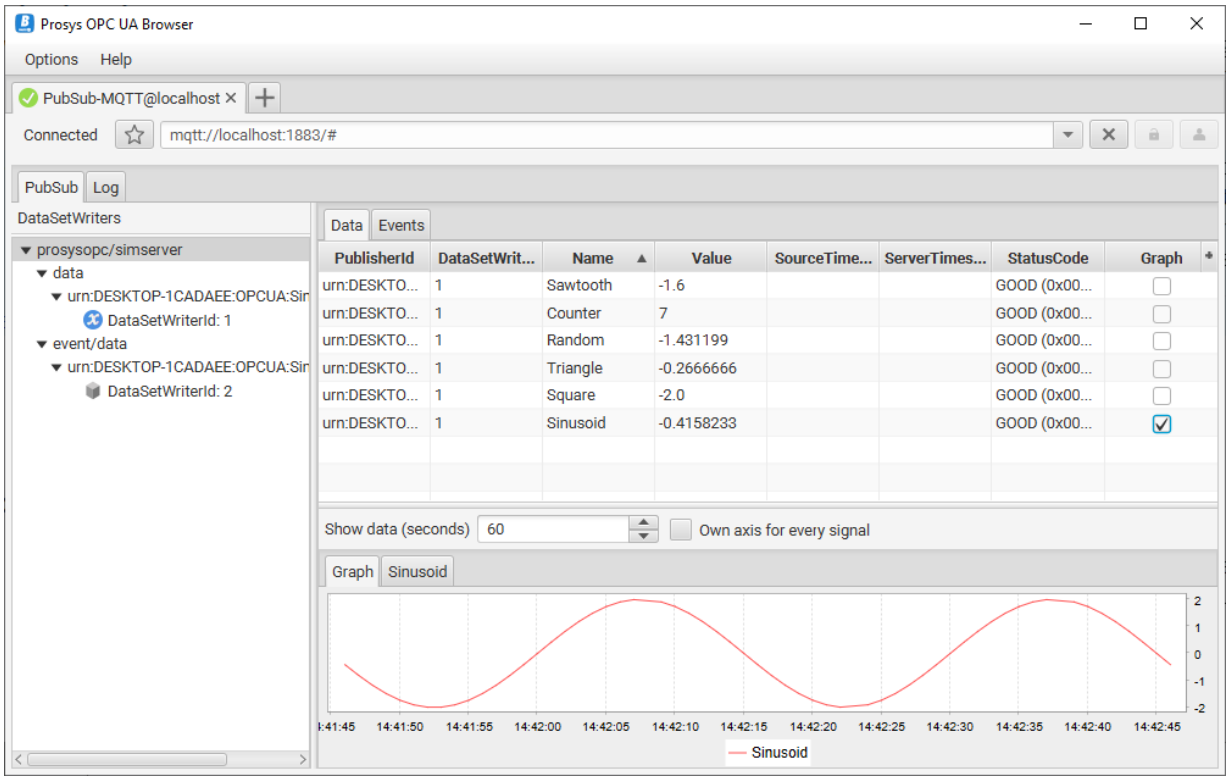

Figure 45. PubSub Data View inside PubSub View.

PubSub Event View

The *PubSub Event View* (Figure 46) represents the Events received by the subscriber in the same way as Events are represented in the Client/Server Event View. It logs all EventMessages including the Event fields that are received from the selected DatasetWriters.

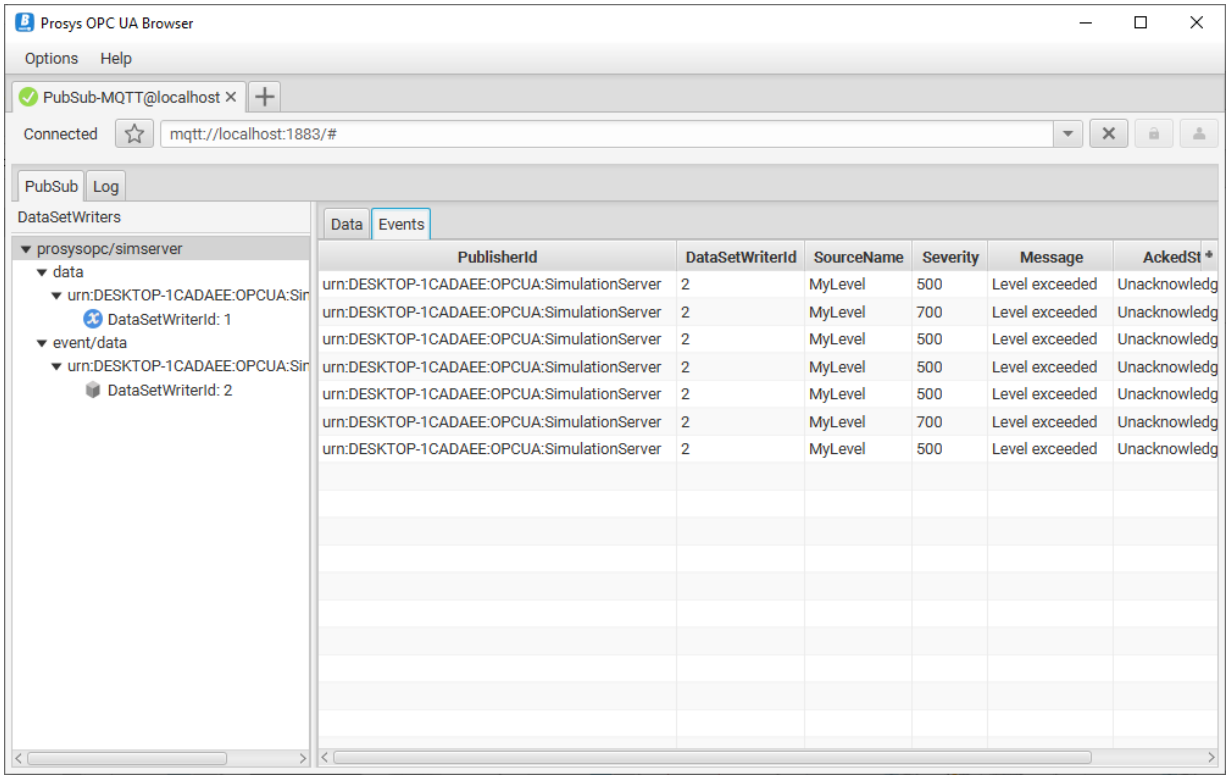

Figure 46. PubSub Event View listing Event Messages.

8.2.2. PubSub Log View

The *PubSub Log View* logs every message that is received from the PubSub Network. You can browse the messages and select those that you want to observe in more detail by clicking them. The whole message will be shown on the right side of the view. This view has a toolbar with two checkboxes and a button:

Active

This selection defines if the updating of the Log is active.

Keep selection in view

This selection prevents the selected message from scrolling from the view when new messages are logged.

Clear

This button clears the log.

Even though the log does not try to interpret the contents of the messages, we can show the topic tree for MQTT messages on the left (see Figure 47). You can also use the topic tree to filter the messages that are displayed.

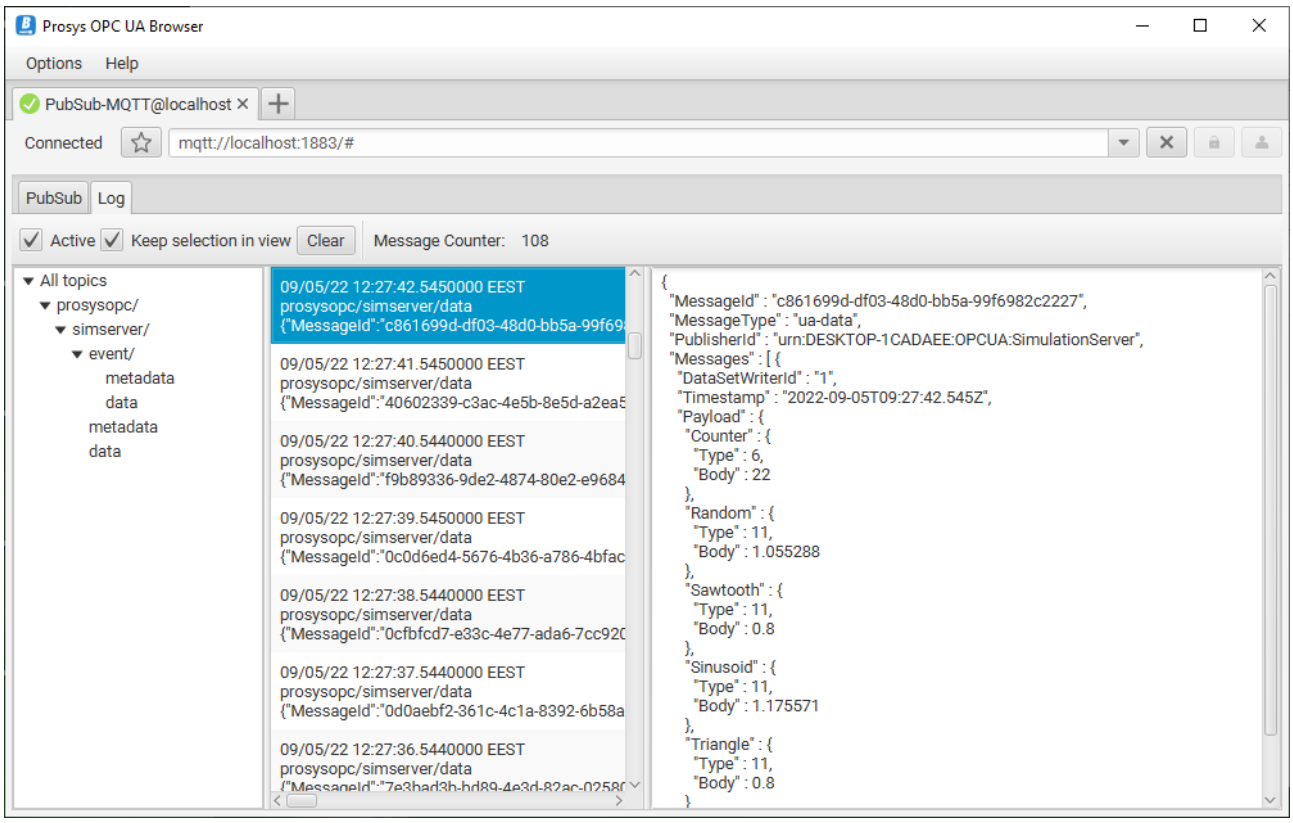

Figure 47. The PubSub Log View that lists all messages from the PubSub Network. MQTT lets you filter messages by Topics in this view.

If the messages are using JSON encoding, you can read the contents of them in the log. UADP messages are binary encoded, and we can only show them in their raw format (see Figure 48).

MQTT messages may be either JSON or UADP, whereas UDP uses only UADP messages.

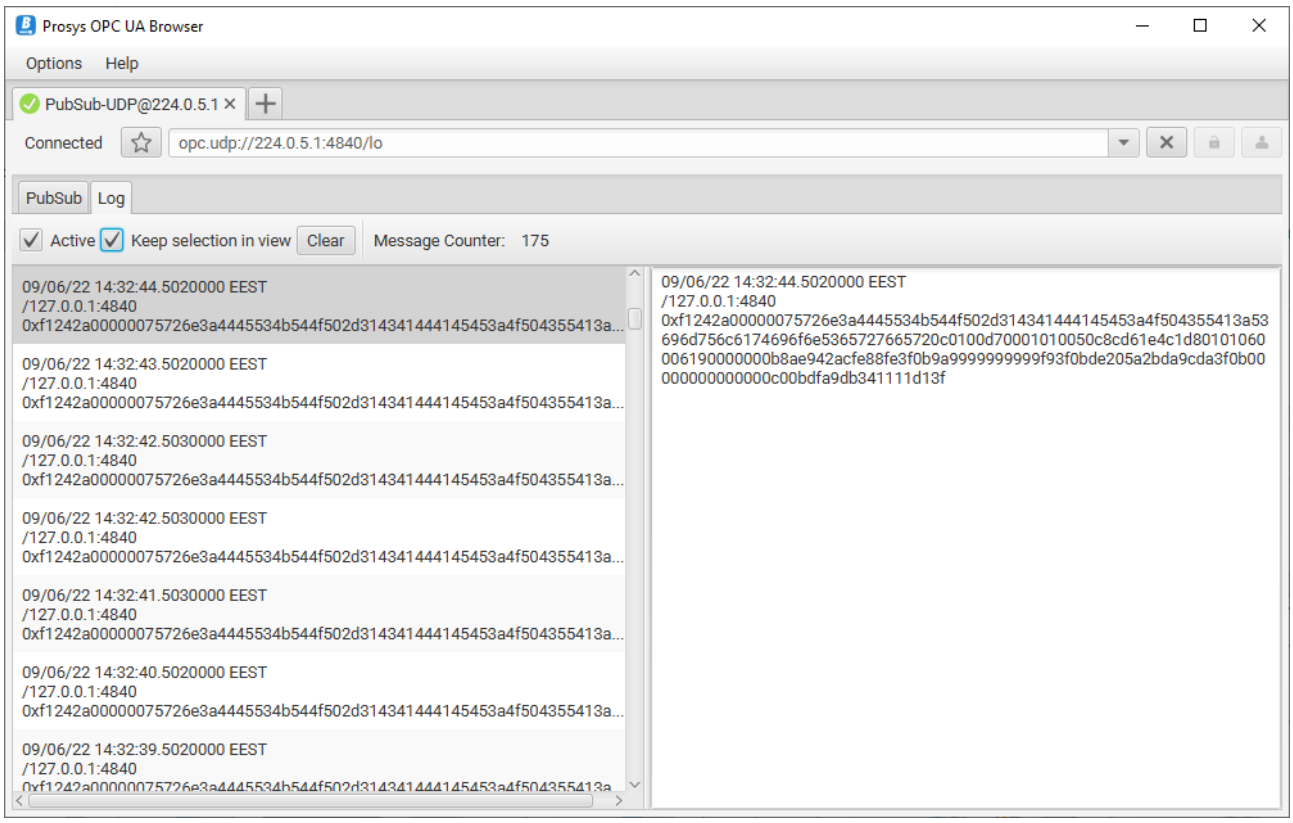

Figure 48. Log for messages from a UDP Network.

If the messages are decoded as OPC UA PubSub messages, you will find their respective data from the PubSub View.

In addition to troubleshooting the details of OPC UA PubSub messages, the Log can be useful for monitoring other messages as well, especially with MQTT/JSON communication.

9. Preferences

The preferences of the OPC UA Browser can be configured in *Preferences Configuration* dialog (see Figure 49) opened by selecting Preferences... in *Options* menu.

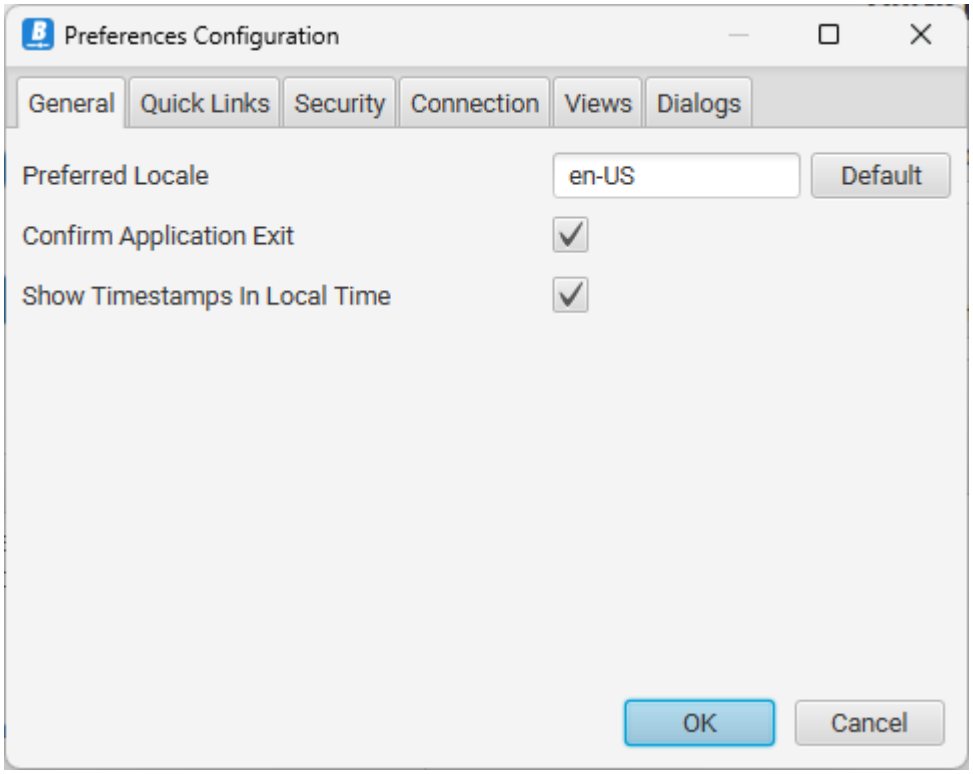

Figure 49. Preferences Configuration dialog.

Hover over the controls on the right side of the dialog with your cursor to show tooltips of preferences to read more information on them. Changes to preferences are applied immediately after accepting them with OK - unless otherwise stated in the tooltips.

10. File Locations

The application saves its settings to a folder in path *(user.home)/.prosysopc/prosys-opc-ua-browser*. The (user.home) is the location of the current user's home folder. If you want to reset to default settings and clear connection settings, just delete the conf.xml settings file in the folder. Settings are saved automatically whenever they are changed and include the connection settings. The certificates used in OPC UA communication are saved to the PKI sub-folder. When saving views is enabled, they're saved in view-configuration.json files per connection.

10.1. Application Logs

The application logs are placed in the log folder under the main settings folder. Logging is configured with the log4j2.xml file. See https://logging.apache.org/log4j/2.x/manual/configuration.html for more information on how to modify the logging configuration.

To limit the disk space required for logging, the log file is rotated:

1. When the log size reaches a configured limit (default maximum size is 50 Mb).
2. When the application is restarted.
3. When the current date no longer matches the log's start date.

When the log file is rotated, it is first archived in compressed .gz format and then reset. The archived log files are organized in folders based on date, and the default maximum amount for archived log files is 50.

10.2. Certificate Store

The main settings folder also contains the certificate store used to keep the certificates known by the application in PKI sub-folder.

The certificate store can also keep Trusted Issuer Certificates that enable the application to automatically trust certificates signed by the respective Issuers.

The certificate store contains the following sub-folders:

- certs for the trusted Application Instance Certificate and trusted issuers.
- crl for certificate revocation lists of trusted issuers.
- issuers/certs for known issuer certificates that are not trusted but are needed for validating certificate chains.
- issuers/crl for certificate revocation lists of known issuer certificates in issuers/certs folder.
- private for Prosys OPC UA Browser's own Application Instance Certificate and its private key.
- rejected for the rejected certificates.

Prosys OPC UA Browser will prompt the user to trust or reject all unknown certificates, unless they are signed by a Trusted Issuer Certificate that is placed in the certs folder. Validating certificate chains may require manually copying intermediate issuer certificates to either certs folder or issuers/certs folder. Providing certificate revocation lists (CRL) is optional in Prosys OPC UA Browser. For trusted issuers,

their CRLs are placed in crl folder. For known issuers, their CRLs are placed in issuers/crl folder.

Although you can modify the certificate store directly on the disk, it is usually better to use the *Certificates* dialog that can be opened from the *Help* menu to define the trusted Application Instance Certificates.