



OPC UA **Simulation Server**

User Manual
Version 5.4.0

Table of Contents

Introduction	1
Installing the Application	1
Windows	1
Linux	1
macOS	2
Uninstalling the Application	2
Windows	2
Linux	2
macOS	2
Editions	3
User Interface - Introduction	4
Expert Mode	4
Open Industry 4.0 Mode	4
Preferences Window	4
Status View	6
Simulation Views	7
Objects View	7
Context Menu	8
Configuring a Variable or Object	9
Value Simulation	10
Defining Values	11
Event Simulation (Professional Edition only)	12
Simulation Control	14
Types View	15
Defining Value Simulation for Types	15
Namespaces View	17
Exporting a namespace (Professional Edition only)	19
Importing an information model (Professional Edition only)	19
Address Space View	21
Endpoints View	22
UA TCP Transport Protocol	22
UA HTTPS Transport Protocol	23
Security Policies	23
Bind Addresses	23
Registering to Local Discovery Server	24
Reverse Connections	24
Applying Changes	24
Certificates View	25
Users View	26
Adding and Removing Users	27
Sessions View	28
Connection Log View	29
Req/Res Log View	30
PubSub View	31
Publisher Connection View	32
Adding custom DataSets (Professional Edition only)	34
Variable DataSets View	35

Adding and removing Variable DataSets (Professional Edition only)	36
Adding Fields to a Variable DataSet	36
Event DataSets View	37
Adding and removing Event DataSets (Professional Edition only)	39
Device View	40
OI4 View	41
OPC UA Server Explained	42
Address Space	42
Objects	42
Types	43
Views	43
File Locations	44
Previous Versions	44
Version 3.x.x and earlier	44
Version 4.x.x	44
Application Logs	44
Certificate Stores	44
Troubleshooting Common Problems	46
Common Problems	46
My Simulation Server won't start	46
Trying to Connect to the Simulation Server with Secure Settings Produces Errors	46
Finding the Log File	46
Contact Us	46
Free License Users	46
Professional License Users	46

Introduction

Prosys OPC UA Simulation Server is an [OPC UA server](#) application, which provides simulated data. You can use it in place of OPC UA servers that provide online production data, for example, to test connections from different OPC UA client applications or help you with your OPC UA system or application development. The latest version of Simulation Server can also be used as an [OPC UA Publisher](#). The Publisher can publish the simulated data to a PubSub Network.

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 Simulation Server is not automatically removed but can be uninstalled manually.



If upgrading from version 4.x.x, you should note that any Nodes you have created in the *Simulation View* will be lost.



If upgrading from version 5.0.0 or 5.0.2, you should note that any existing settings in the *Users View* and the *Bind Addresses* section of the *Endpoints View* will be reset.

The installation includes a complete “embedded” Java Runtime Environment (JRE). This ensures that you don’t need to install Java on your computer, although the application is running in a Java environment, and you don’t need to 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 request. You should get the correct package, depending on your target environment.

Windows

On Windows, run the installer executable and follow the instructions. By default, the application is installed in the folder *Program Files/ProsysOPC/Prosys OPC UA Simulation Server*.

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 add file permission to make the installation shell script executable with the command:

```
sudo chmod u+x prosys-opc-ua-simulation-server-linux-x.x.x-x.sh
```

Then run the installation shell script with the command:

```
sudo ./prosys-opc-ua-simulation-server-linux-x.x.x-x.sh
```

This will open the installer, where you can follow the steps to complete the installation. The application is installed in the folder *opt/prosys-opc-ua-simulation-server* by default.

macOS

On macOS, run the installer application and follow the instructions. The application is installed in the folder */Applications* by default.

Uninstalling the Application



Uninstalling the application does not remove any existing application settings or project configuration. They can be manually removed by deleting the folder described in the chapter [File Locations](#).

Windows

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

Linux

On Linux, open the terminal and navigate to the installation folder (default folder is */opt/prosys-opc-ua-simulation-server*) and use the command:

```
sudo ./uninstall
```

macOS

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

Editions

Prosyst OPC UA Simulation Server 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 Simulation Server:

- Adding or using imported information models. In the Free Edition, the **Import NodeSet file** functionality in the Namespaces view will be disabled along with any previously imported information models.
- Allows for adding custom DataSets in the **PubSub View**. The limit for Fields in a Variable DataSet is six in the Free Edition whereas Professional Edition has no limit for the amount of Fields.
- Possibility to simulate Events for Objects in **Objects View**
- **Open Industry 4.0 Mode** which allows simulation of a Device and Open Industry 4.0 Open Edge Computation (OEC).

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

User Interface - Introduction

The user interface of the Simulation Server consists of several views. By default, the application starts in the *Basic Mode* that shows only the essential views used to edit and simulate the server's address space.

The views available in the *Basic Mode* are:

- [Status](#)
- [Objects](#)
- [Types](#)
- [Namespaces](#)

Expert Mode

You can enable more configuration and monitoring options by switching to the *Expert Mode* through the [Options](#) menu.

The *Expert Mode* contains the following additional views:

- [Address Space](#)
- [Endpoints](#)
- [Certificates](#)
- [Users](#)
- [Sessions](#)
- [Connection Log](#)
- [Req/Res Log](#)
- [PubSub](#)

Open Industry 4.0 Mode

You can enable two additional views by selecting *Show Open Industry 4.0 Mode* through the [Options](#) menu.

The *Open Industry 4.0 Mode* contains the following views:

- [Device](#)
- [OI4](#)

Preferences Window

You can configure the application functionality and the UI in the *Preferences* window accessible through the [Options](#) menu. The available preference options are briefly described below:

Autosave project backup file

enables or disables the autosaving of the simulation configuration to a separate backup file (happens every 5 min). This can be manually used to recover the previous simulation configuration.

Link Objects and Types views

links the [Types View](#) to automatically select the type Node of the selected instance in the [Objects](#)

View.

Start simulation with default values

resets the simulation values to their defaults when the application starts.

Show value plot

shows or hides the view that shows the shape and range of the simulated value in a graph with the current value.

Show type name

enables or disables the type name postfix '::' for instance Nodes in the *Objects* view.

Show type name tooltip

enables or disables the hovering type name tooltip, for instance, Nodes in the *Objects* view.

Simulate methods

allows methods of instances of imported types to return a dummy value (0 or null) when called.

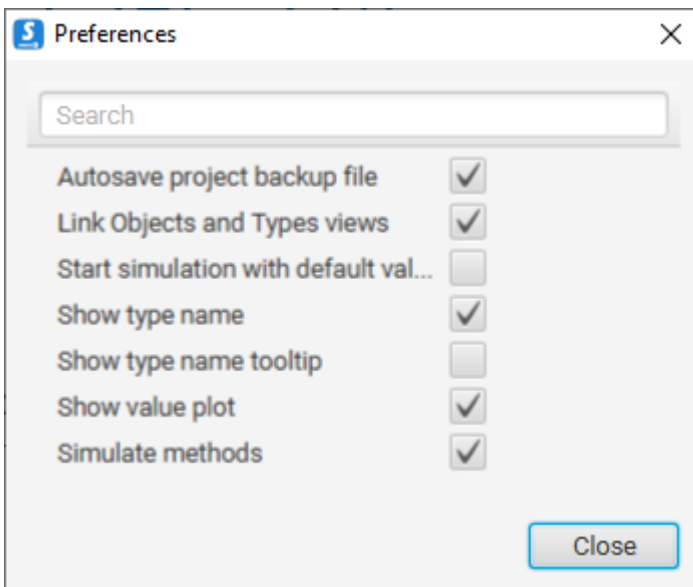


Figure 1. The UI and the application's functionality can be customized in the Preferences window.

Status View

The Status View (Figure 1) displays information about the current server status and available connection addresses. *PubSub Status* can also be seen in the Status View along with the possible *PubSub Connection Address*.

If everything is fine, Server Status displays *Running*. The status will show a message displaying the current startup phase during startup. In case of errors, it may turn to *Error* with additional information about the exact problem.



You can have only one instance of the Prosys OPC UA Simulation Server running at one time.



Figure 2. The Status View displays some basic information about the server and PubSub connection.

The Connection Addresses show the OPC UA addresses, which the OPC UA client applications can use to connect to the OPC UA Server. Simulation Server supports both UA TCP (*opc.tcp*) and UA HTTPS (*opc.https*) protocols. You can define the available addresses in the [Endpoints View](#). You can easily copy a connection address to the clipboard by clicking the button next to it. Then, you can paste the address to the OPC UA client of your choice.

The Status View also displays the *Current Server Time* and *Server Starting Time*. If remote connections are used, you should ensure that the computers run with synchronized clocks. Secure connections, especially, will not work if the clocks between the client and server are too much out of sync. Server Status, including *CurrentTime* and *StartTime*, are also available for OPC UA Client applications. You will find it as part of the Server Object (see [OPC UA Server Explained](#)).

The bottom of the Status View also displays the current edition. Depending on the license, information on the licensee and the expiration date might be shown.

Simulation Views

The application contains three Simulation Views that can be used to create a customized [OPC UA Server Address Space](#) with user-created Nodes with simulated values. These views include the *Objects*, *Types* and *Namespaces* views.

Objects View

The Objects View enables the creation of custom Objects and Variables with live data. This enables simulation of different server configurations and testing client applications against a custom address space and data. The Objects View shows a filtered view of the *Objects* folder in the server's address space. The *Server* object, sample Nodes provided by Simulation Server and all simulation configuration information has been filtered out to give you a clean slate to work on.

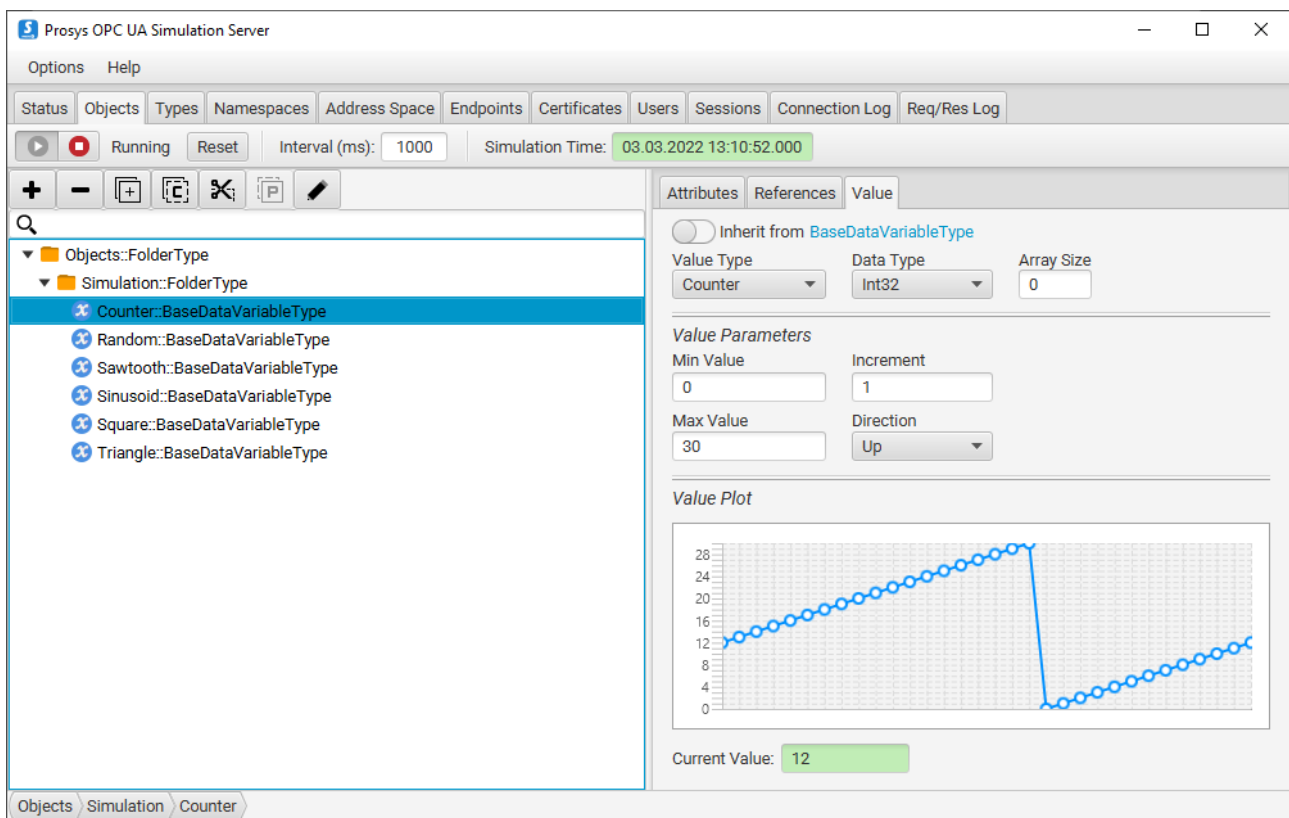


Figure 3. The Objects View enables the configuration of custom Objects and Variables.

The *Objects* tab is divided into two parts (see [Figure 3](#)). The left-hand side of the view contains a tree view of the filtered *Objects* folder. On top of the Node tree, you can find a toolbar of buttons that can be used to edit the contents of the *Objects* folder. The right-hand side of the view contains three tabs: *Attributes*, *References* and *Value*. The first two are for viewing information of a selected Node, and the last one is for defining value simulations for Variables.



Variables with DataTypes that cannot be simulated are grayed out to signify that they cannot have simulated values.

You can search for Nodes with the search box above the Node tree on the left. The input is used to search for Nodes with matching DisplayNames. You can cycle through multiple Nodes matching the search term by pressing **F3**.

Context Menu

When you right-click a Node in the Node tree, you get a context menu that offers you the same editing options as the toolbar above the tree. Every action on the menu is context-sensitive, meaning that the behavior of the action depends on which Node is selected in the tree view.

The following list briefly describes the available actions:

Add Node

contains a list of actions that enable adding new Nodes (Objects or Variables) to the server.

Add Folder

adds a new FolderType Object under the selected Node with an Organizes reference and a user-selected namespace and name.

Add Variable

adds a new Variable under the selected Node with a user-selected namespace, name, VariableType, optional members and ReferenceType.

Add Object

adds a new Object under the selected Node with a user-selected namespace, name, ObjectType, optional members and ReferenceType.

Add Property

adds a new PropertyType Variable under the selected Node with a HasProperty reference and a user-selected namespace and name.

Remove Node(s)

removes all selected Nodes and their child Nodes. It is the only action that handles selections of multiple Nodes.

Duplicate Node

adds identical copies of the selected Node. The duplicates have the same TypeDefinition, and they are placed under the same Node as the selected Node with the same ReferenceType. For Variables, the value configuration is also copied from the original Variable. The names of the duplicates have the suffix (X), where X is an incrementing number.

Copy Node

creates a duplicate of the selected Node on the clipboard, similar to the Duplicate Node action.

Cut Node

removes the selected Node and places it on the clipboard.

Paste To Node

adds the Node from the clipboard under the selected Node.

Edit Node

allows for making changes to the Node, such as renaming it or changing the ReferenceType used to connect the Node to its parent. You can also select which optional members should be added to or removed from the Node.

In a range of supported cases, if the type of the Node defines any InstanceDeclarations with ModellingRules Optional, OptionalPlaceholder or MandatoryPlaceholder, then those will also appear in

the *Add Node* menu and can be quickly created through the shortcut.

Configuring a Variable or Object

When adding a new Variable or Object or when editing an existing one, you will be prompted to fill all or some of the following information:

Namespace

defines the namespace that the Node belongs to.

NodeId Type

defines the type of the NodeId. The options are: Numeric, String, Guid and Opaque.

NodeId Value

defines the value of the NodeId.

Name

defines the DisplayName and the BrowseName of the Node.

Type

defines which VariableType or ObjectType the Node will implement. Type defines the structure of the Node.

Select Optional Members

dialog can be used to include any of the available members with an Optional ModellingRule defined by the selected type.

ReferenceType

defines the type of the Reference that connects the Node with its parent Node (when adding a new Node, the selected Node is the parent Node).

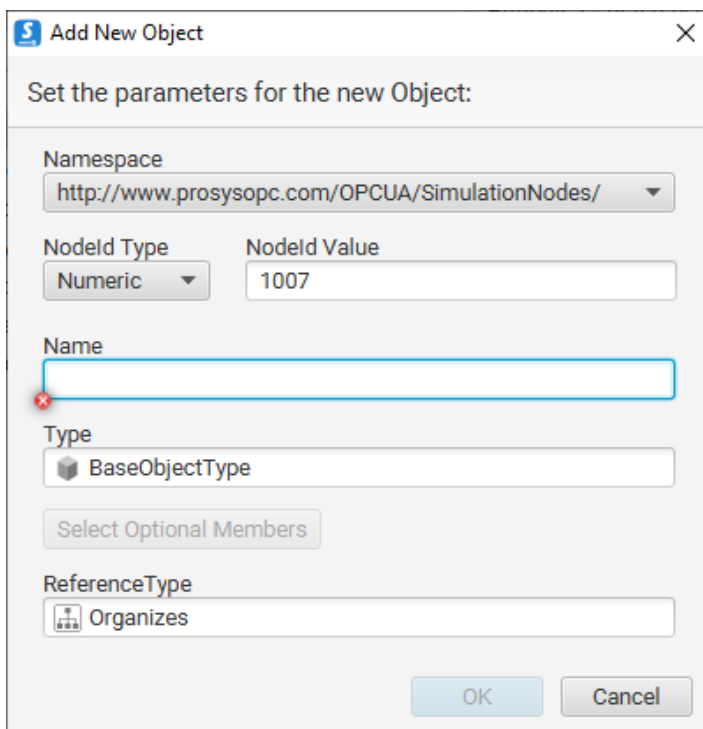


Figure 4. You can configure the parameters of new or existing Variables and Objects.

Value Simulation

The current values of Variables can be simulated with various simulation parameters. The values are simulated with mathematical functions that dictate how the value of the Variable changes over time when the simulation is running. The current value properties of a Variable, VariableType or an InstanceDeclaration Node can be defined with the controls in the *Value* tab (see Figure 5) on the right-hand side of Objects View.

A Node can either have an attached value or inherit a value. Variables can inherit a value from InstanceDeclaration whereas VariableTypes can inherit from their parent types. You can toggle between an attached value or an inherited value by clicking the switch at the top of the Value tab. Enabling value inheritance will disable the simulation settings (apart from the data type and array size of the Node) since the inherited Node defines these settings. You can easily access the inherited Node by clicking its name at the top of the Value tab. By default, added instances inherit their values from their type definition or InstanceDeclaration (if the data type is compatible for simulation).

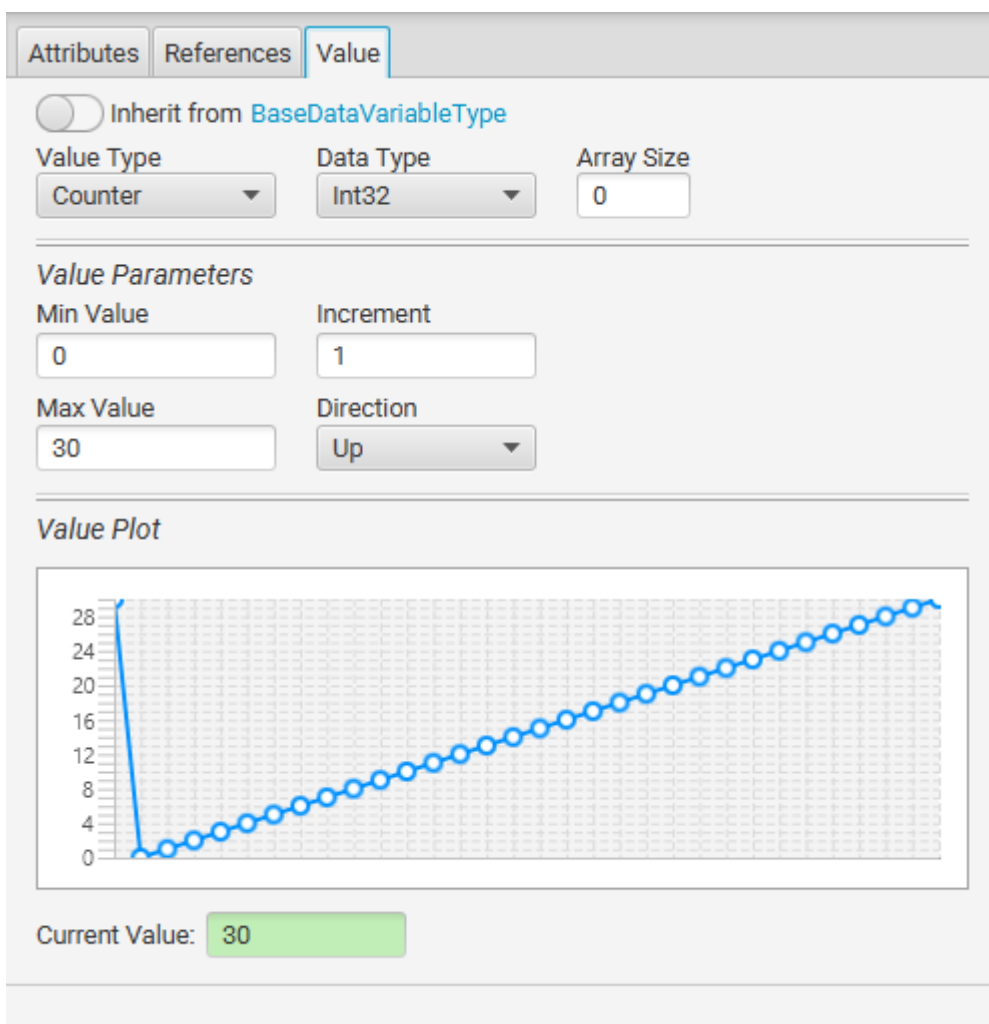


Figure 5. Value tab for defining value simulation in Objects view.

Value Type defines the type of the simulation function. Each alternative has a varying set of additional parameters that enable the exact configuration of the details. Value Type can be one of the following:

Constant

means that the value will not change but will remain constant. The initial constant value is defined in the *Initial Value* field. This determines the initial value for the Variable when the server is started.

The value can be changed during runtime by client applications, but when the server is restarted the value is set according to the initial value. Constant value is the default option for new Variables.

Counter

has parameters *Min Value*, *Max Value*, *Increment* and *Direction*. The value increases or decreases on each simulation interval by *Increment* until it reaches *Max Value* or *Min Value*. When *Direction* is *Up*, the value increases until it reaches *Max Value* and then resets to *Min Value*. When *Direction* is *Down*, the value decreases until it reaches *Min Value* and then resets to *Max Value*. When *Direction* is *Up & Down*, the value increments until it reaches *Max Value* and then decreases until it reaches *Min Value*. If *Data Type* limits the value more than the limits, it will be clamped at the high or low limit, respectively.

Random

produces a value that changes randomly between the uniform range defined by its two parameters *Min Value* and *Max Value*.

Waveform functions

are a group of functions that share a common set of parameters: *Min Value*, *Max Value*, *Period* and *Time Offset*. *Min Value* and *Max Value* define the high and low values of the waveform function. *Period* defines the time to complete one complete wave. *Time Offset* can be used to move the phase of the wave in relation to the current time. The following waveforms are available:

- *Sawtooth*
- *Sinusoid*
- *Square*
- *Triangle*

Data Type defines the OPC UA DataType used for the Variable. The simulated value is mapped to the variable's value. Therefore, in practice, the data type may limit the value assigned to the Variable more than the Parameters because the value is truncated to match the variable's data type.

Array Size defines the size of the one-dimensional array of values that will be created by duplicating this value. If you do not want an array but a scalar value, you can leave the *Array Size* as 0.

Value Plot section displays the range and shape of the value simulation with the selected parameters visualized in a graph. The graph shows one entire cycle for the function with each data point marked. The section also contains a field showing the current simulated value.

Defining Values

When you have selected a Variable, follow these steps to define the new value simulation:

1. Disable value inheritance at the top of *Value* tab.
2. Select *Value Type* (see [Value Simulation](#) for explanations).
3. Select *Data Type* and *Array Size* (see [Value Simulation](#) for explanations).
4. Fill in required parameters. The *Value Plot* at the bottom of the tab shows how the value will change with the given parameters.

Event Simulation (Professional Edition only)

Similarly to simulating Values for Variables, the *Events* tab allows you to simulate Events for Objects. In [Figure 6](#), you can see that Objects have the *Events* tab where Variables have the *Value* tab.

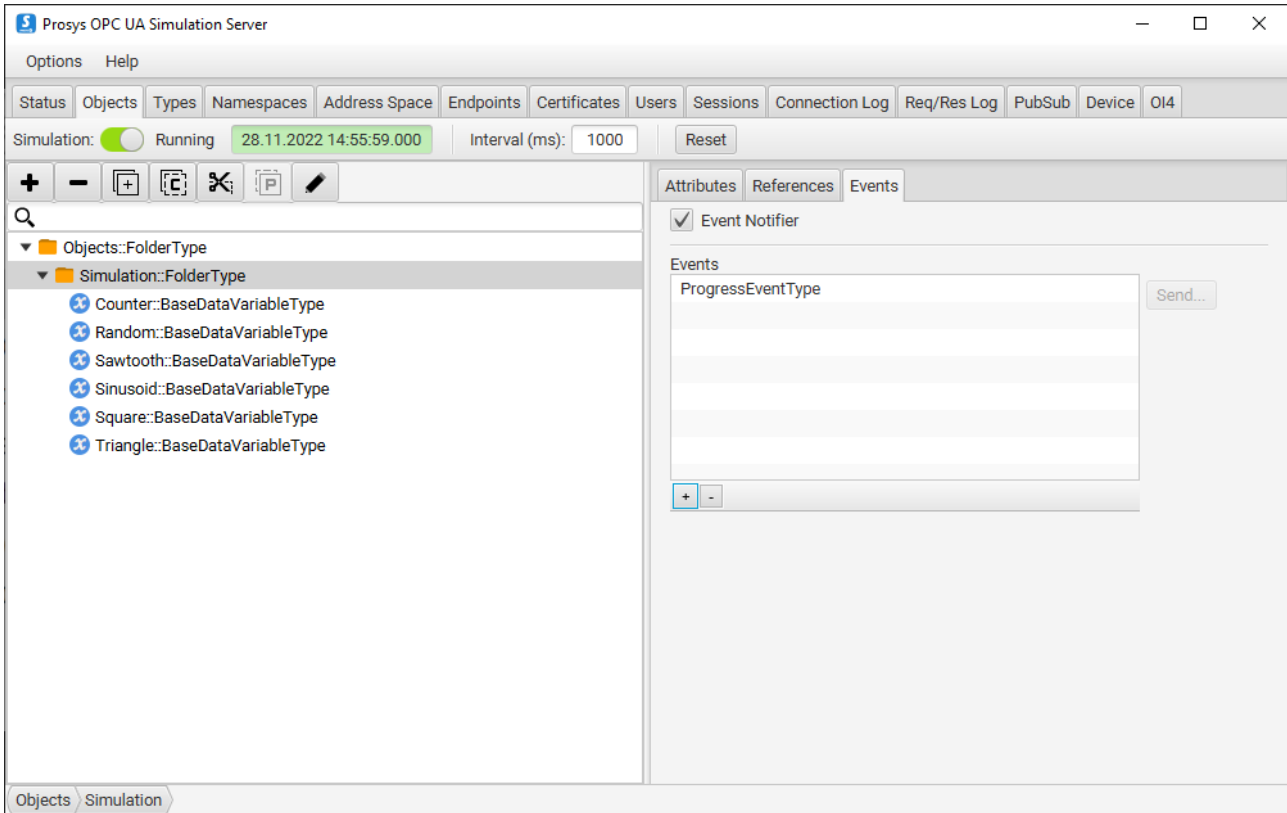


Figure 6. Events tab for simulating events for Objects in Objects view.

To simulate Events for a selected Object, you must first enable an Event Notifier for it ([Figure 6](#), 1.). Next, you can click the '+' button ([Figure 6](#), 2.) to add a new Event for the Object. The dialog shown in [Figure 7](#) will let you select the type of Event you want to add.

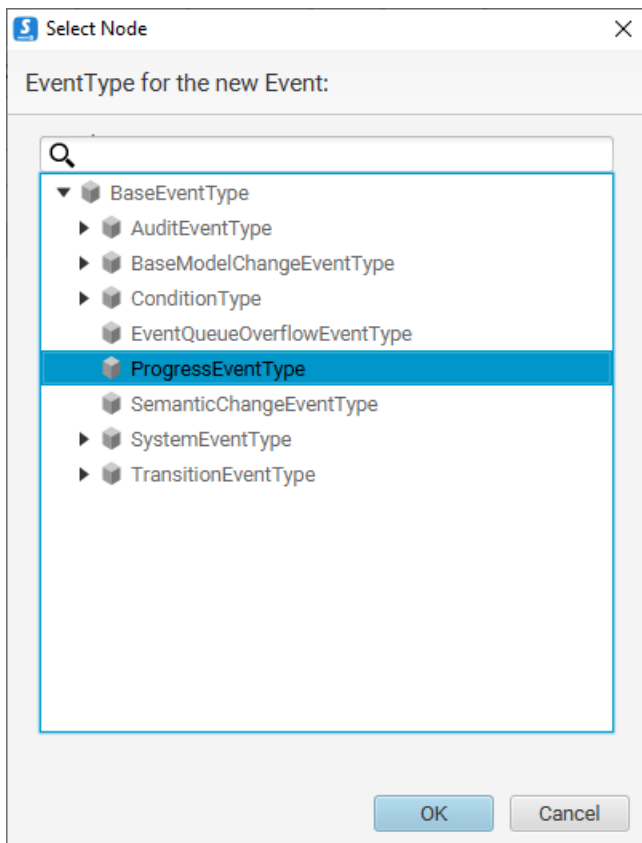


Figure 7. You can select the Event Type for your new Event.

When you want the server to send an Event Notification for one of the simulated Events, you can select the Event and click 'Send'. This will open a new dialog (see [Figure 8](#)), where you can define the details of that Event. After filling the necessary information, you can click 'OK', and the server will send the Event Notification similarly to "real" Event Notifications.

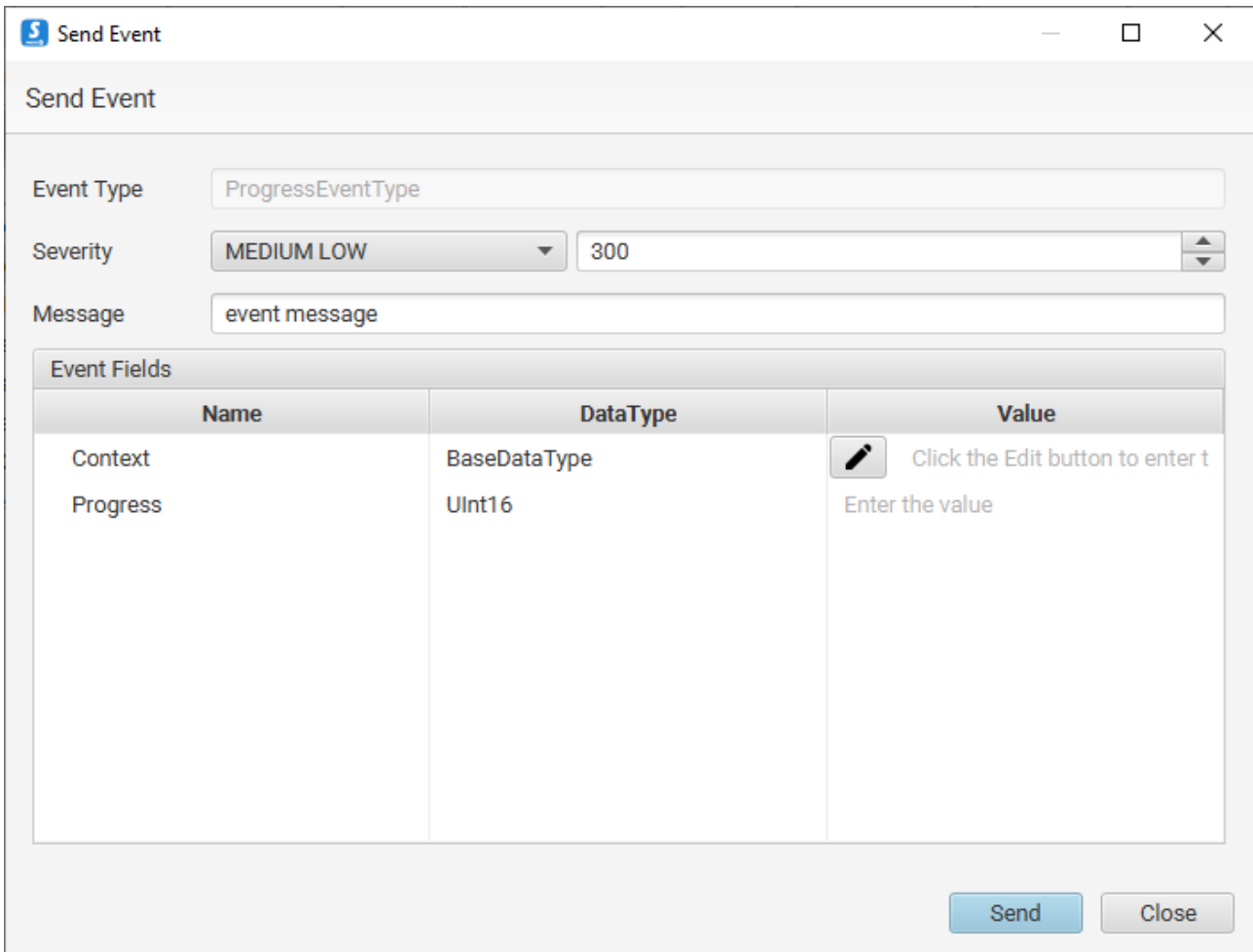


Figure 8. Define the details for your simulated Event.

Simulation Control

The top bar of the Objects View contains simulation controls. You can start or stop the simulation with the associated switch. When the simulation is stopped, none of the values is updated.

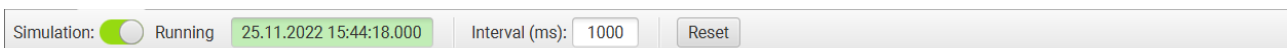


Figure 9. Toolbar for simulation control.

The *Interval*, displayed in milliseconds, defines how often the values are updated. The allowed range is between 100 and 60000 milliseconds. *Simulation Time* shows the timestamp corresponding to the latest value calculation. Simulation can be run in the current time only. The *SourceTimestamp* of the values is locked to the interval increments, which means that even if scheduling for the simulation would be off by a few milliseconds, the *SourceTimestamps* are exactly one interval away from the previous simulation time. If simulation calculation takes more than one simulation interval, intervals are skipped as necessary to keep up with current time.

You can also reset the simulation using the 'Reset' button. Resetting the simulation means that all the Variable values are set to their default values starting from the next simulated value. The default value of a Variable depends on the parameters set for the simulation and the value type.

Types View

In the Types view, you can define simulated values to different VariableTypes and InstanceDeclarations of both ObjectTypes and VariableTypes. These values are inherited by all instances created from the specific types in the [Objects View](#). Types can also inherit their values from parent types.

From [Figure 10](#) we can see that the view is divided into two parts. The left-hand side contains two tabs: *ObjectTypes* and *VariableTypes*, for going through all types contained by the server. This includes all types from imported namespaces as well. The types are represented in a tree view. On the right-hand side you can find three tabs: *Attributes*, *References* and *Value*. The first two tabs can be used to view attributes and references of a selected Node. The last tab is for defining value simulation.



Abstract types in the Types View are greyed out to signify that they cannot be instantiated. All InstanceDeclarations that have incompatible data types for simulation are also greyed out.

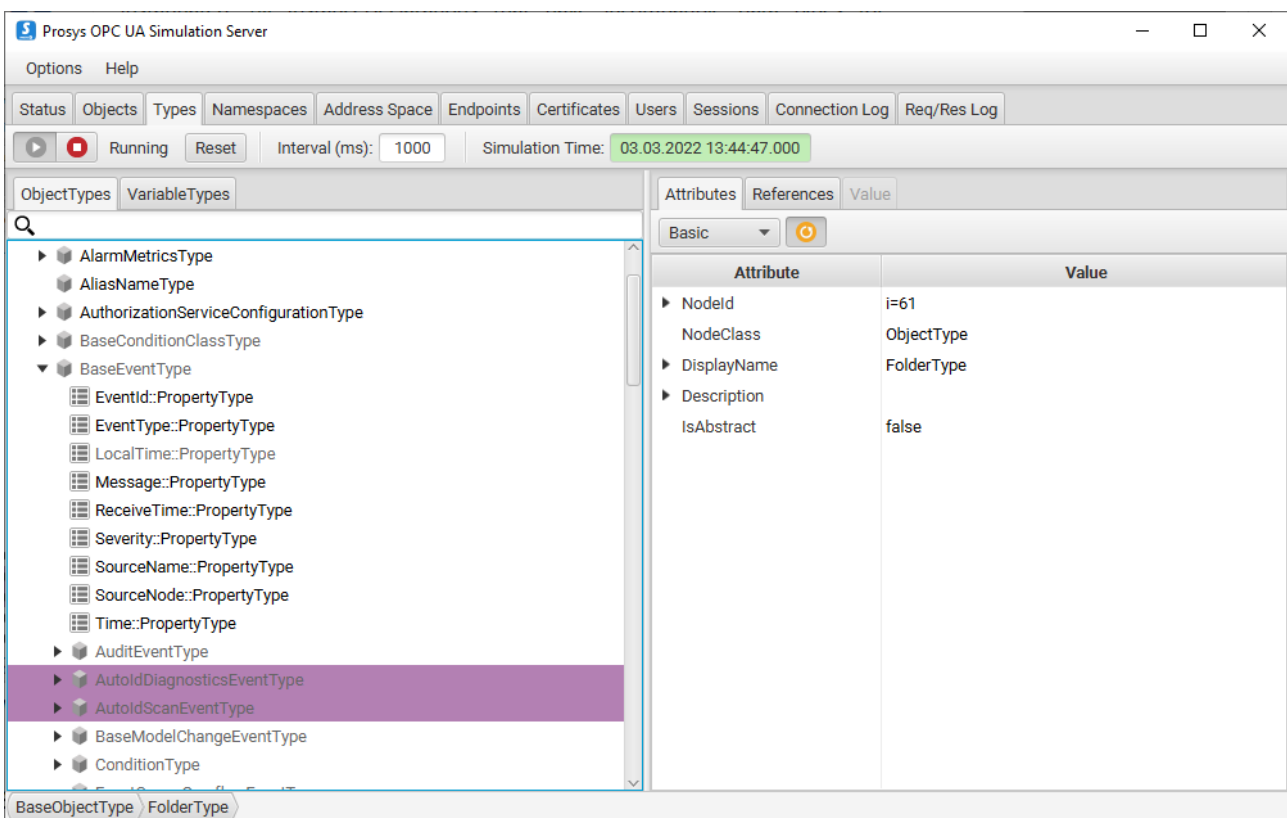


Figure 10. Types view with added AutoID ObjectTypes highlighted.

Defining Value Simulation for Types

First, you need to select a tab: *ObjectTypes* or *VariableTypes*. Next, go through the list of types and select the VariableType or InstanceDeclaration to which you want to define a simulated value. When the *Value* tab is enabled, you know the type can have a simulated value.

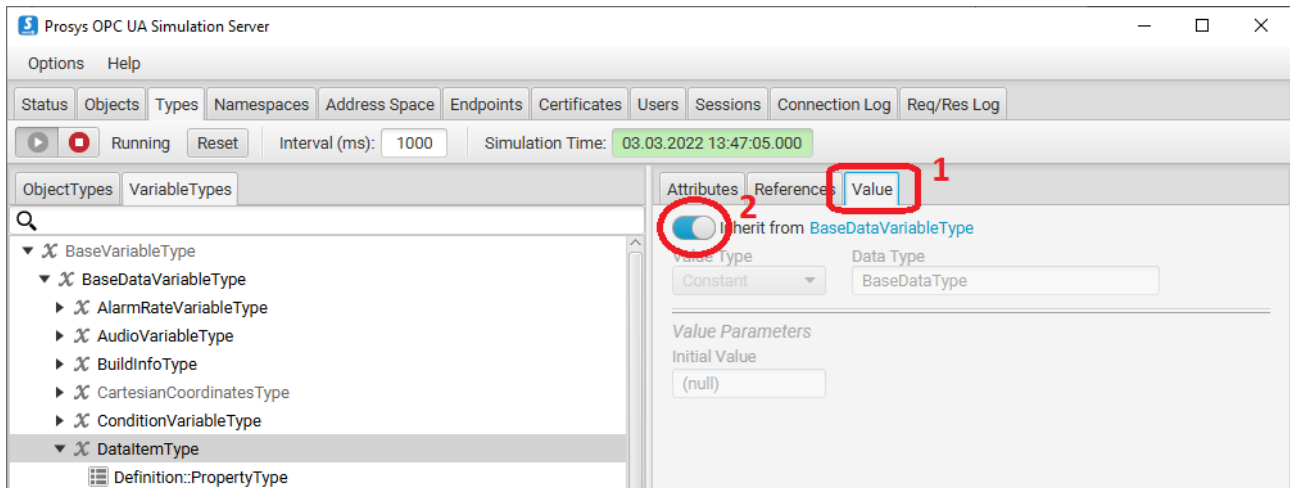


Figure 11. To define a value simulation, select Value tab (1) and disable inheritance of values (2).

Defining the value simulation works the same way as in [Objects View](#). To define a value yourself, you need to disable inheritance from the parent type of the current type. Now you can define your value simulation. See [Value Simulation](#) and [Defining Values](#) for further information about the options in the Value tab.

Namespaces View

The Namespaces View is used to display information about and edit the namespaces on the server. Namespaces can be edited by clicking the toolbar buttons above, through the right-click menu that appears when clicking a namespace or keyboard shortcuts (Ctrl+E). Quick access to namespace editing is also achieved by double-clicking a namespace.

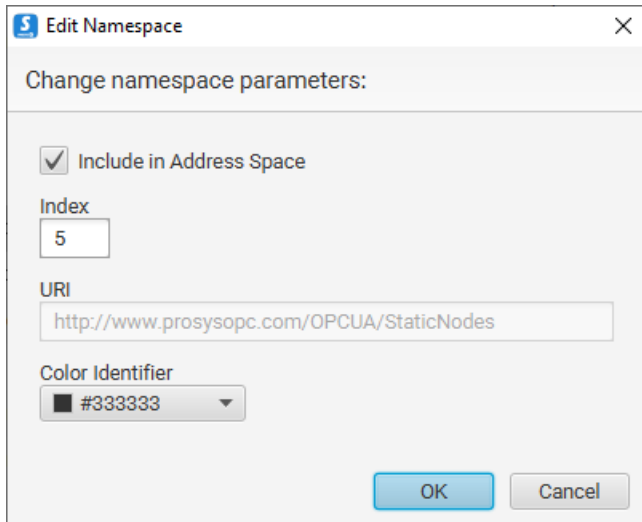


Figure 12. Dialog for editing a namespace. Editable details are enabled.

The information and options available for each namespace are listed below. Some of the details are editable (see [Figure 12](#)) and rest are purely informational and can be seen in the table shown in [Figure 13](#).

Include in address space

the checkbox can be used to enable or disable the namespace. Disabling a namespace means that the entire namespace and all its Nodes are removed completely from the server, but the namespace is still visible (but greyed out) in the Namespaces View so it can be enabled again later.

Lock icon

If the namespace is marked with a lock icon, its contents cannot be modified or exported.

Index

refers to the position of the namespace in the NamespaceArray of the OPC UA server.

URI

a unique identifier for the namespace.

Version

for imported companion specifications that provide the information, displays the version number and date.

Types

displays the number of types provided by the namespace.

Instances

displays the number of instances provided by the namespace (note that this doesn't include absolutely all instances but only those under the Objects folder and outside the Server object).

Color

can be applied to a namespace, and it will change the background color of all Nodes belonging to the namespace in the Objects and Types views.

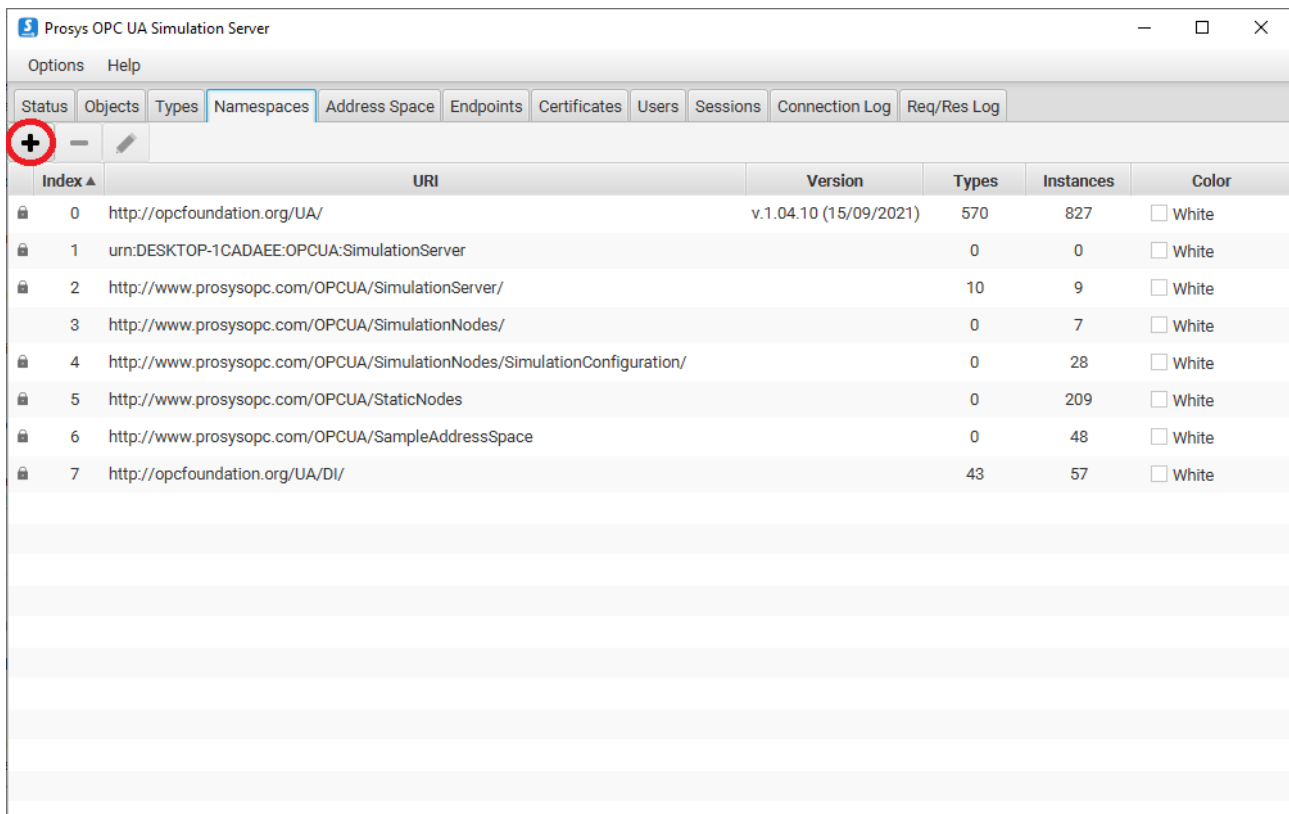


Figure 13. Namespaces View shows the information on the namespaces present in the server.



Enabling/disabling a namespace or changing the index or URI of a namespace all require the application to be restarted for the changes to take effect.

You can use the + and - buttons at the top of the view to add or remove namespaces. Adding a new namespace has two options: you can either add a new empty namespace by clicking **Add Namespace** or import an information model in the NodeSet format by clicking **Import NodeSet** (see section [Importing an information model \(Professional Edition only\)](#)). Removal of a namespace is an action that immediately and permanently removes the namespace and all its Nodes (versus disabling a namespace, which allows for enabling the namespace again later).



A new empty namespace does not add anything to the server, but you can add new objects and variables to it in the *Objects* view.

Namespaces can have dependencies to each other. For example, Variables and Objects in one namespace can use types from another namespace, or a type in one namespace can be a subtype of a type in another namespace. These dependencies are displayed in the Namespaces View visually when you select a namespace:

- *Blue* color highlights any present namespaces that the selected namespace depends on.
- *Yellow* color highlights any present namespaces that depend on the selected namespace.



You cannot remove a NodeSet that is a dependency for another NodeSet present on the server.



Namespaces can have dependencies to certain **versions** of other namespaces. So make sure that all version numbers are correct.

Exporting a namespace (Professional Edition only)

The server allows you to export any available imported or created instance namespaces (namespaces that do not contain any types) as a NodeSet file. If you have an instance namespace, you can export it by right-clicking it and selecting *Export Namespace*. You can then choose where the NodeSet file will be saved.



See [OPC UA specification, Part 6, Annex B](#) for more information on the NodeSet file format for exchanging information models.

Importing an information model (Professional Edition only)

You can import any OPC UA information model to Simulation Server by importing a file in the NodeSet format. First, start by clicking **+** (circled with red in [Figure 13](#)) and selecting the option to **Import NodeSet** (see [Figure 14](#)). Next, you will need to find the NodeSet file from your computer.

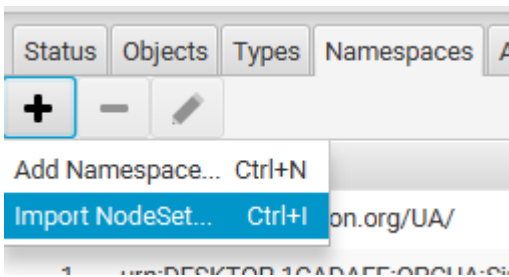


Figure 14. Click on the **+** and then select **Import NodeSet**.

NodeSets for all of the OPC UA companion specifications are available to download at [OPC Foundation's GitHub page](#). You can easily download a .zip file containing all of the models. It is a good idea to download all models because some models require other models to already exist in the list of namespaces before they can be imported.

When importing a NodeSet, it first goes through rigorous validation to ensure that it matches the OPC UA Specification and forms a correct and coherent model with the other namespaces on the server. All invalid models are rejected, and an error dialog is shown.



For a continually updated status on the compatibility of companion specification models, see [our blog post](#).

When importing a NodeSet, you do not need to know all prerequisite models to import beforehand. If you try to import a model that requires other models, an error message will pop up to let you know which models you need to import first. Import those models and then try again with the model you want to import.

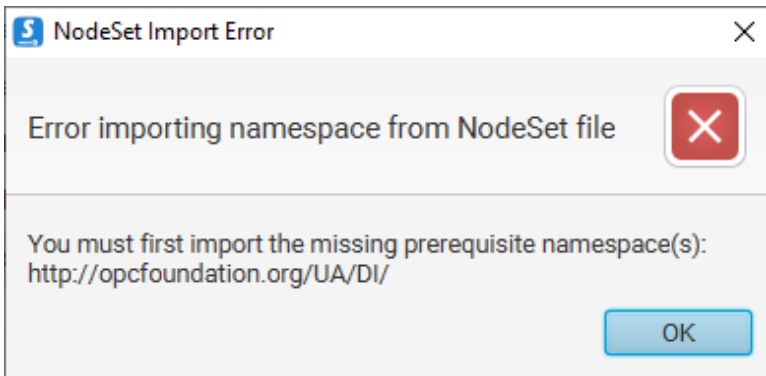


Figure 15. An error message listing all required NodeSets.

Figure 15 represents an example of an importing attempt that ended with an error. The model in question is OPC Foundation's AutoID model. After importing the required DI model, the AutoID model can be imported successfully. If you are unsure of what the imported NodeSet adds to the server, you can change the color of that namespace. This way every node added by the namespace is highlighted with the selected color, as can be seen in Figure 10. Now you can move to [Types View](#) to define value simulations with the newly added VariableTypes and ObjectTypes.

Address Space View

The Address Space View is accessible in the [Expert Mode](#).

The *Address Space View* (Figure 16) shows the OPC UA Server Address Space as it will be available for client applications. The view is also similar to the one used by [Prosyst OPC UA Client](#). The Nodes are shown in the tree view on the left, and the Attributes and References of the currently selected Node are shown on the right.

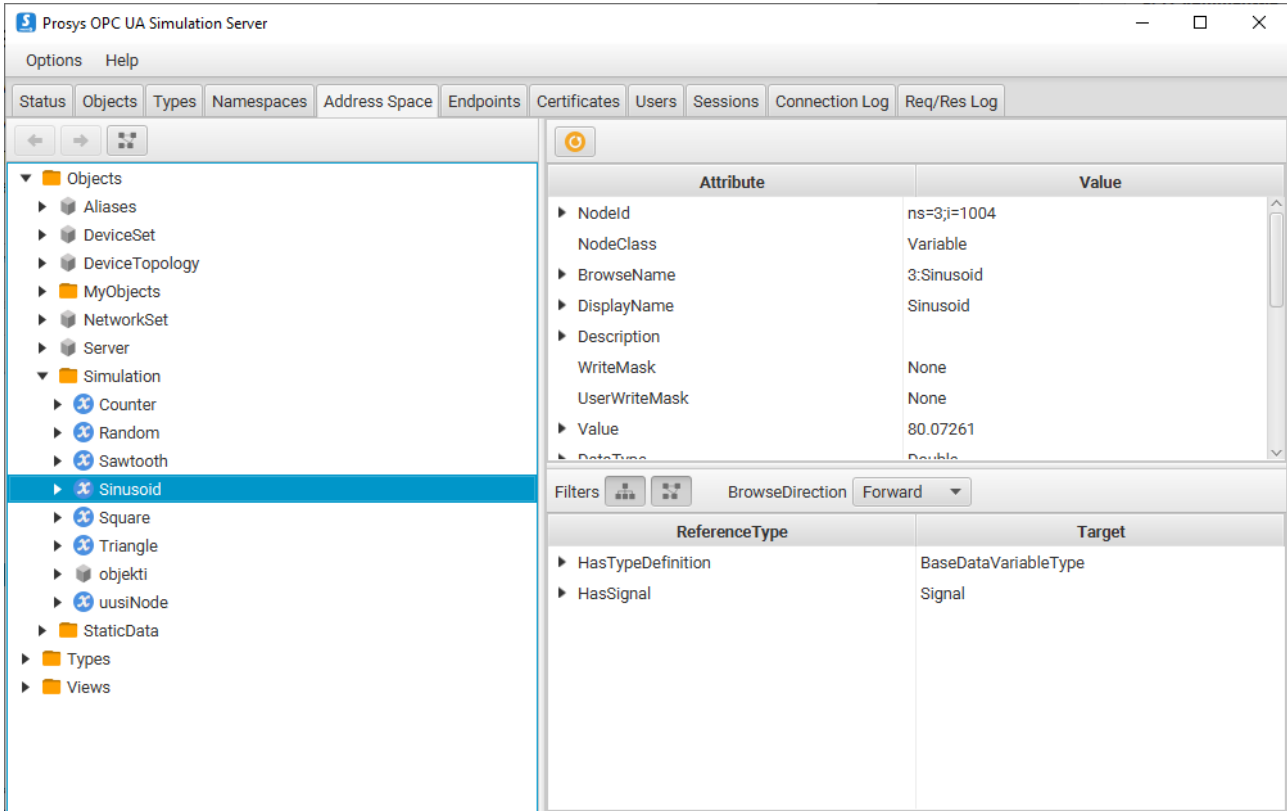


Figure 16. Address Space of the server. The Attributes and References of the selected Node are displayed on the right.

Read more about the contents of the Address Space at [OPC UA Server Address Space](#).

Endpoints View

The Endpoints View is accessible in the [Expert Mode](#).

Endpoints define the OPC UA connection addresses and security modes that the OPC UA clients may use to connect to the OPC UA server. If you don't need to limit the security options, you can usually leave the Endpoints to the default settings.

Should you consider opening communication to any publicly available network, you may need to harden the server configuration via the security settings. As a minimum, disable **None** from the Security Modes.

The *Endpoints View* (Figure 17) allows you to configure these settings and verify which endpoints the server is exposing.

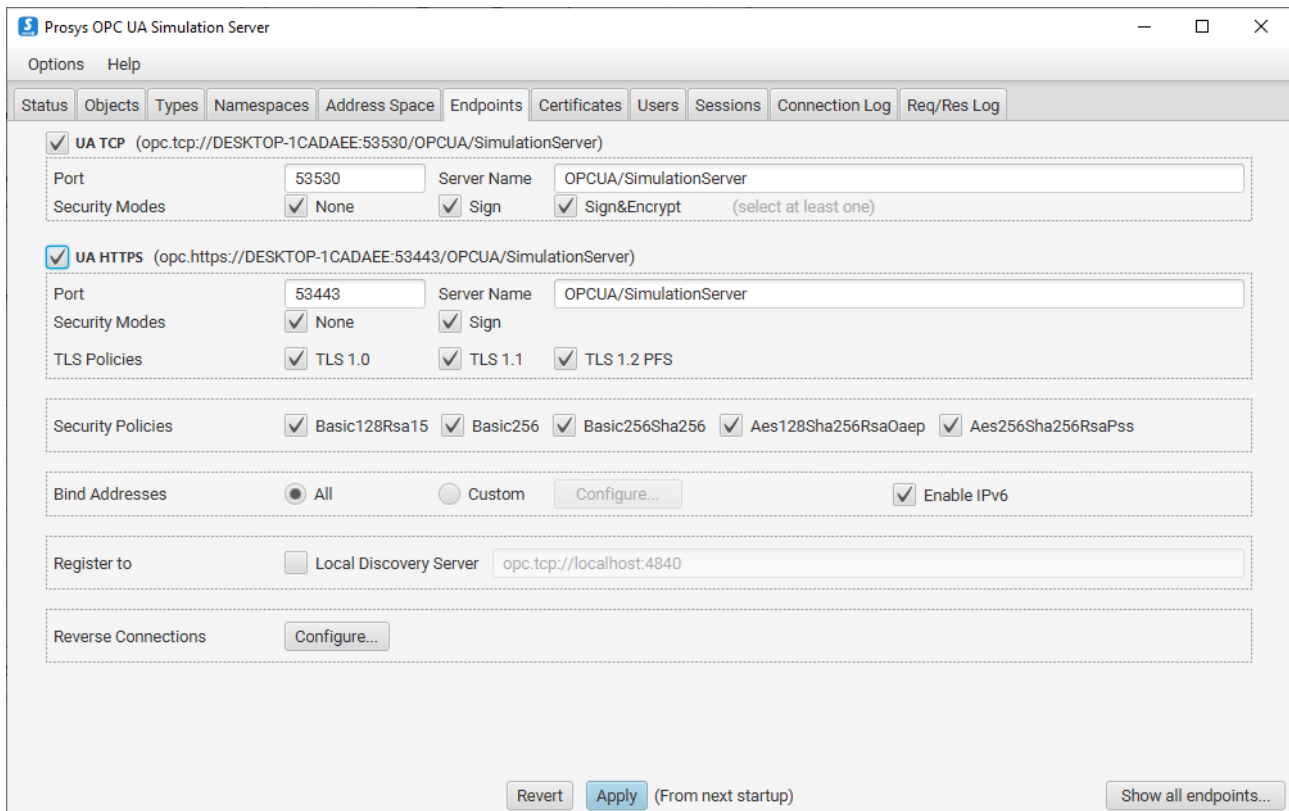


Figure 17. The Endpoints View allows you to configure the connection addresses and security options available for OPC UA clients to connect to the OPC UA server.

By default, Prosys OPC UA Simulation Server enables two transport protocols:

- UA TCP
- UA HTTPS

UA TCP is the usual transport protocol supported by all client applications. UA HTTPS is an alternative transport protocol that is not commonly used.

UA TCP Transport Protocol

UA TCP is an OPC UA specific binary communication, including full OPC UA specific security implementation. The Port and ServerName define the exact connection address, which is displayed at the top (opc.tcp://<hostname>:53530/SimulationServer).

In addition to the connection address, you can define the security modes that the server accepts. The client applications will decide which mode they wish to use, so the server can only configure which options are available. If you want to disable insecure connections, you can deselect the **None** option from Security Modes.

Security Mode **Sign** will ensure that all traffic can be validated by the client and server application and may not be modified during the transfer. Security Mode **Sign&Encrypt** will also make all communication between the client and server encrypted, which means that it cannot be seen by any third party that might be monitoring the network traffic.

If the client decides to use one of the secure modes, the client and server application will also use *Application Instance Certificates* to define which applications they trust to be allowed to make a connection. Please refer to the [Certificates View](#) for details about creating trust between the applications.

UA HTTPS Transport Protocol

UA HTTPS is an alternative transport protocol, which is not required by OPC UA, but which can enable an alternate communication pathway for some installations. It should not be confused with normal HTTPS, which web servers use. OPC UA servers are not web servers, but they can use HTTPS for the transport of OPC UA messages.

Security in UA HTTPS is based on TLS. There are different versions of TLS, and the client and server applications will negotiate the version that they use based on the ones they support. The OPC UA applications that support UA HTTPS may define different TLS versions, and you will need to make sure that there is at least one common TLS version that both of them support.



The applications will also need separate HTTPS certificates for TLS authentication to use UA HTTPS. The HTTPS certificates are usually signed by a CA certificate, and to trust each other, the applications may need to trust the CA certificates as well. The HTTPS certificates of the client applications are not validated at the moment.

The Application Instance Certificates are used to authenticate applications in UA HTTPS when the clients connect with **Sign** mode.

Security Policies

Security Policies define alternative algorithms that the client applications may choose from. It is important to enable algorithms that the client applications support. Some policies (Basic128Rsa15 and Basic256) are already deprecated in OPC UA version 1.04, but to enable interoperability with all client applications, it may be necessary to keep them enabled.

Bind Addresses

The server is bound to listen to connections from all network interfaces by default. If necessary, you can limit the network addresses that it listens to with the *Bind Addresses* setting. Select *Custom* and define the details behind the *Configure* button.

IPv6 addresses are also enabled by default, but you can choose to disable them as well.

Registering to Local Discovery Server

The Endpoints View ([Figure 17](#)) also has controls for registering the server to a Local Discovery Server. See the OPC Unified Architecture Specification Part 12 for more information about the Local Discovery Server. The view allows enabling and disabling the registration and changing the connection address for the Local Discovery Server. Note that the registration requires a secure connection. Therefore the discovery server needs to trust the Simulation Server's certificate.

Reverse Connections

Since OPC UA version 1.04, the connections can also support the so-called Reverse Connection option. In this case, the server application is responsible for opening the connection to the client, which will then take over and establish the connection normally.

By configuring the *Reverse Connections*, you can define the addresses of the client applications listening to connection attempts from the server. The server will then start actively trying to connect to these addresses.

Applying Changes

Once you have changed any endpoint settings, you must click **Apply** to save the settings. You must then **restart the server** before they come to effect. If you don't apply the settings, you can use **Revert** to restore the previously stored settings.

Certificates View

The Certificates View is accessible in the [Expert Mode](#).

The *Certificates View* (Figure 18) allows you to define which OPC UA Client applications are allowed to connect to the OPC UA Server. This is the first layer of validation available in OPC UA technology, prior to user authentication that is managed in the [Users View](#).

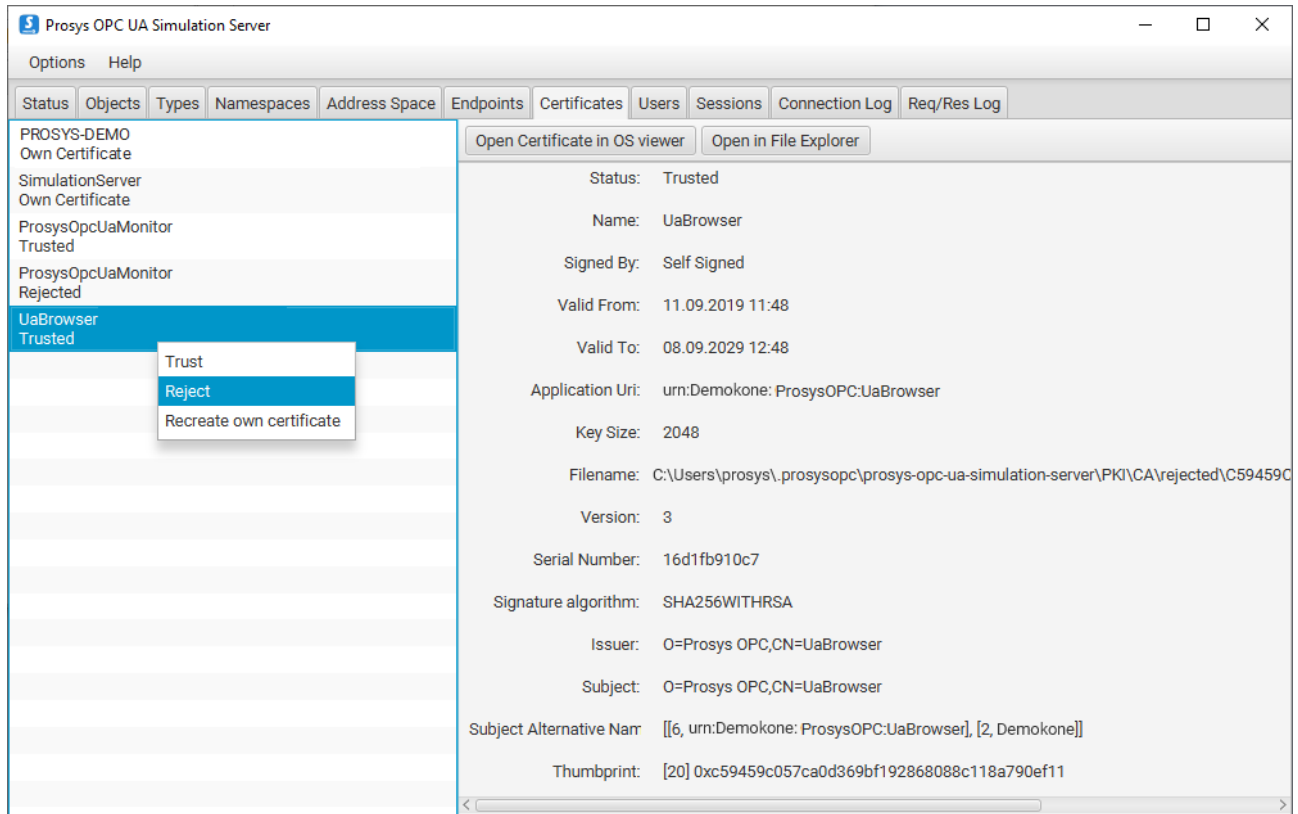


Figure 18. The Certificates View allows you to define which certificates you trust.

OPC UA applications use *Application Instance Certificates* to identify and authenticate other OPC UA applications that they communicate with.

When a new OPC UA client application connects, its certificate will be added to the Certificate List as **Rejected**. A certificate is trusted by right-clicking the Certificate in the list and selecting **Trust** from the context menu. Likewise, you can reject a certificate from the same menu.

If your operating system is capable of displaying the contents of certificate files, you can use the **Open Certificates in OS viewer** function to launch that for the active certificate. The certificates are stored in a Certificate Store as described in [Certificate Stores](#).

If you are issuing certificates with a Certificate Authority (CA), you can copy the certificate of the CA to the Certificate Store as a trusted certificate: this will make Prosys OPC UA Simulation Server trust automatically all certificates that are signed by the CA.



Application Instance Certificates are only used with *secure* OPC UA communications when the client connects with **Sign** or **Sign&Encrypt** mode. If you wish to always require application authentication, you will need to disable the **None** level of security in the Endpoints View.

Users View

The Users View is accessible in the [Expert Mode](#).

The *Users View* ([Figure 19](#)) lets you configure the *User Authentication Methods* as well as the user accounts that can be used to access the OPC UA Server.

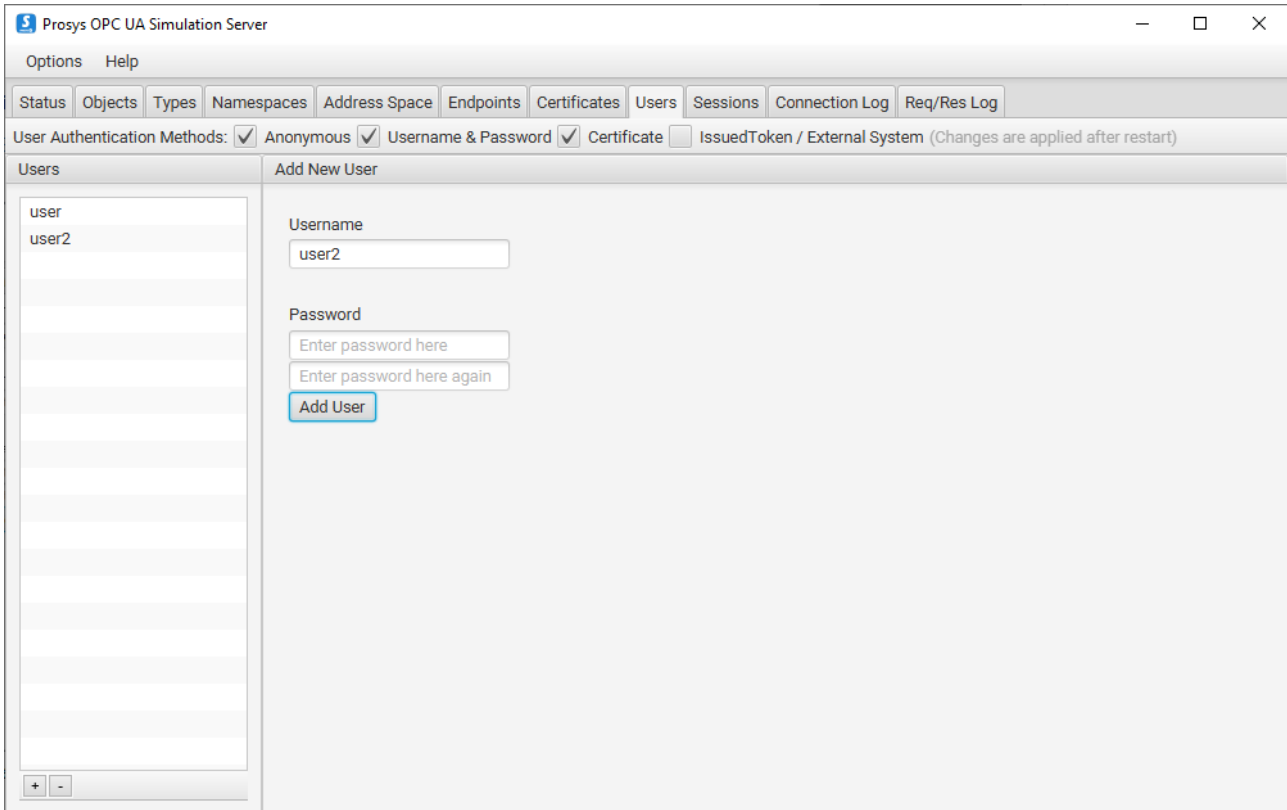


Figure 19. You can use the Users View to add or remove users that may access the OPC UA Server.

The alternative User Authentication Methods, which you can define at the top of the view, are:

Anonymous

which enables connection without any specific user account.

Username & Password

enables the traditional username and password combinations to be defined.

Certificate

enables the server to accept users based on X.509 certificates.



If you change the User Authentication Methods, you will have to restart the application before they take effect.

Anonymous access is enabled by default, but if you wish to increase access control to your server, you can deselect Anonymous from the User Authentication Methods.

If you have enabled the Username & Password authentication, you can define the users that may access the OPC UA Server. The currently defined users are visible on the left in the *Users*. See [Adding and Removing Users](#) for information on how to define more users.

If you have enabled the Certificate-based authentication, you can define the trusted user certificates by

adding them to the respective location in the USERS_PKI folder as described in [Certificate Stores](#).



Prosyst OPC UA Simulation Server does not provide the means to generate the user certificates, so you must create them with some other tool.

Adding and Removing Users

When you see the form on [Figure 20](#) next to the *Users* list, you can set a name and password for a new user. If this form is not visible, you can click the + button on the bottom of the *Users* list. Add the user by clicking on **Add User**. When a user has been added on the server, it will appear on the *Users* list.

The 'Add New User' form contains the following elements:

- Add New User** (Title)
- Username** label above a text input field with placeholder text "Enter new user name here".
- Password** label above two text input fields, both with placeholder text "Enter password here".
- Add User** button.

Figure 20. Add New User

In some cases, you might need to change a password for an existing user. This can be achieved by selecting the user from the *Users* list and filling in the form (see [Figure 21](#)) that appears next to the list. You do not need to know the previous password to set a new one.

The interface shows a 'Users' list on the left and a 'Settings For User: user2' panel on the right.

Users List:

user
user2

Settings For User: user2

- Username: user2
- Change password** label above two text input fields, both with placeholder text "Enter new password".
- Change password** button.

Figure 21. Change user's password

If you want to remove a user from the server, you can simply click on the user on the *Users* lists and then click the - button on the bottom of the list. This will permanently remove the selected user.

Sessions View

The Sessions View is accessible in the [Expert Mode](#).

The *Sessions View* ([Figure 22](#)) contains information about the currently open OPC UA Client Sessions.

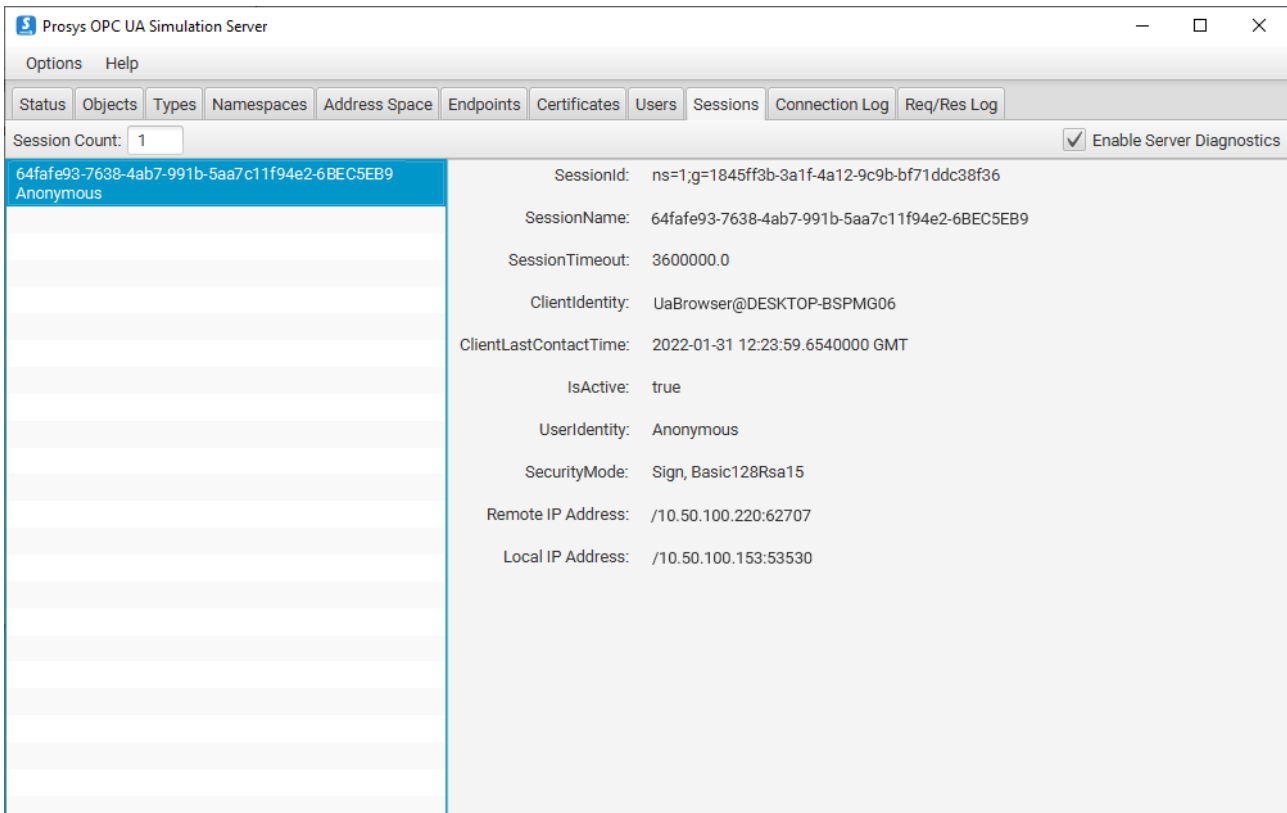


Figure 22. The Sessions View displays all current OPC UA Sessions in the OPC UA Server.

Session Count shows the total number of open sessions at the moment.

The individual sessions are visible on the left, including the session names. When you select a session, more information will be shown on the right side of the view.

Connection Log View

The Connection Log View is accessible in the [Expert Mode](#).

The Connection Log View ([Figure 23](#)) displays a history of client connections. The log entries have color codes that group all entries related to the same OPC UA session together. The list is reset when the server is closed.

Status	Objects	Types	Namespaces	Address Space	Endpoints	Certificates	Users	Sessions	Connection Log	Req/Res Log
Timestamp	Event Type	SessionName	SessionId	ClientIdentity	UserType	UserName	Security *			
20.06.2019 15:07:22....	Session activated	session1	ns=1,g=a6b9b42...	Unified Automation Ua...	Anonymous		SignAndE...			
20.06.2019 15:07:22....	Activating Session	session1	ns=1,g=a6b9b42...	Unified Automation Ua...			SignAndE...			
20.06.2019 15:07:22....	Session created	session1	ns=1,g=a6b9b42...	Unified Automation Ua...			SignAndE...			
20.06.2019 15:06:39....	Session closed	urn:Demokone:U...	ns=1,g=5e1b3c2...	Unified Automation Ua...	Anonymous		SignAndE...			
20.06.2019 15:01:31....	Session activated	urn:Demokone:U...	ns=1,g=5e1b3c2...	Unified Automation Ua...	Anonymous		SignAndE...			
20.06.2019 15:01:31....	Activating Session	urn:Demokone:U...	ns=1,g=5e1b3c2...	Unified Automation Ua...			SignAndE...			
20.06.2019 15:01:31....	Session created	urn:Demokone:U...	ns=1,g=5e1b3c2...	Unified Automation Ua...			SignAndE...			
20.06.2019 14:45:19....	Session closed	ProsystOpcUaCli...	ns=1,g=85528a4...	ProsystOpcUaClient	Anonymous		None, None			
20.06.2019 14:45:11....	Session activated	ProsystOpcUaCli...	ns=1,g=85528a4...	ProsystOpcUaClient	Anonymous		None, None			
20.06.2019 14:45:11....	Activating Session	ProsystOpcUaCli...	ns=1,g=85528a4...	ProsystOpcUaClient			None, None			
20.06.2019 14:45:10....	Session created	ProsystOpcUaCli...	ns=1,g=85528a4...	ProsystOpcUaClient			None, None			

Figure 23. Connection Log View shows all established client connections to the server.

Req/Res Log View

The Req/Res Log View is accessible in the [Expert Mode](#).

The Req/Res Log View can be used to record every request and response transferred to/from the server over the OPC UA communication protocol (except opening and closing of the secure channel). By default, this functionality is off; check the **Active** checkbox to start recording.



The recording is kept in memory until cleared by pressing the **Clear** button or shutting down the application; therefore, it is possible to run out of memory if kept on for a long time.

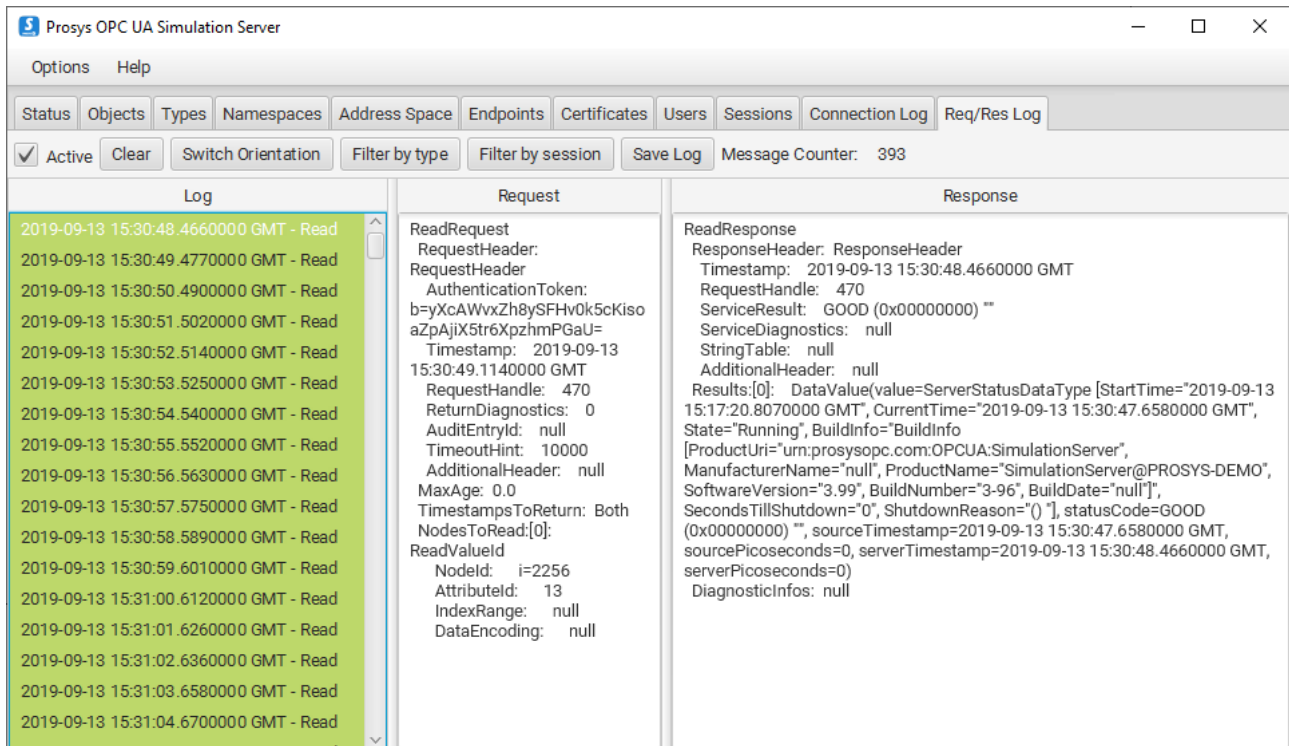


Figure 24. Request-Response Log View.

The view is divided into three parts under the toolbar. The toolbar contains controls for filtering the recording and exporting them to a file.

The three parts of the view show a log list on the left-hand side, the request details of a selected log item in the middle, and the response details of the selected log item on the right-hand side. All sessions are color-coded in the log list to differentiate between separate sessions.

PubSub View

The PubSub View is accessible in the [Expert Mode](#).

The PubSub View can be used to configure and run multiple OPC UA Publishers. Configuring a Publisher includes setting the Network type and Connection Address for that Publisher as well as selecting which DataSets to publish to the PubSub Network. In the case of PubSub, DataSets are containers for the data that has been selected to be published to the Network. One DataSet can contain many Fields storing data from different Variables or Events selected from the OPC UA server.

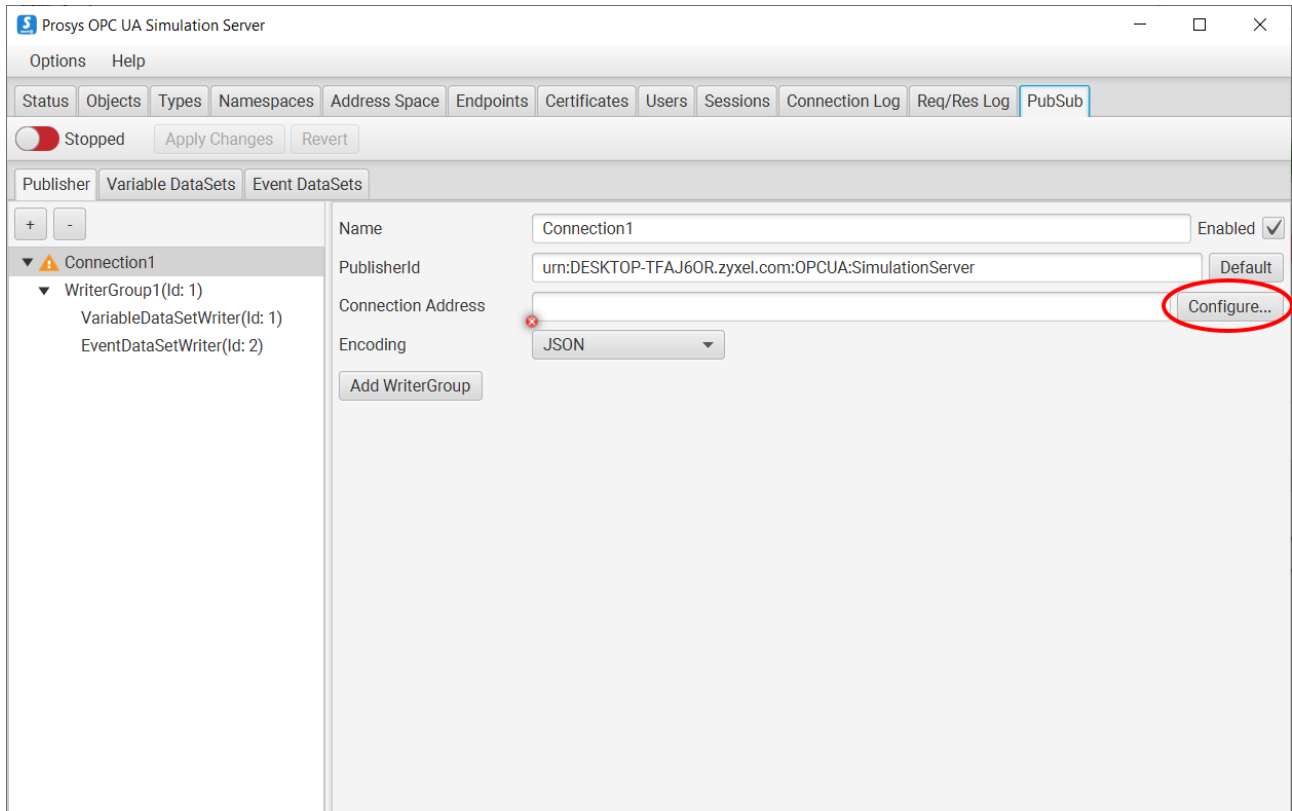


Figure 25. PubSub View on startup.

As can be seen from [Figure 25](#), the PubSub View contains three control buttons and three tabs:

Switch button

Starts or stops publishing.

Apply Changes

Updates all changes to the *Publisher* so that the published data is up to date.

Revert

Reverts all *Publisher* settings to the state they were in at the time of the latest [Apply Changes](#).

Publisher Connection View

In this tab you can configure Publishers (connection and DataSetWriters).

Variable DataSets View

In this tab you can add, remove and configure different Variable DataSets.

Event DataSets View

In this tab you can add, remove and configure different Event DataSets.

Publisher Connection View

The tree view on the left lists all Publisher connections and their WriterGroups. You can add new Connections by clicking the '+' button and remove them by clicking the '-' button.

To define a Publisher, you first need to configure the Network in the *Publisher Connection View*. To do that, you need to define the Connection Address. Click the **Configure** button to open the *Publisher Connection Settings* dialog (see [Figure 26](#)). Select the Network type for your Publisher from the drop-down selector. The Simulation Server currently offers you three options:

mqtt

A Message Queue Broker using MQTT.

mqttts

A secured MQTT network.

opc.udp

A local network using UDP. For this network type, you need to select which network card the data will be sent to. This is called *Network Interface* in the dialog.

After selecting the Network type, you can set the rest of the Connection Address. The MQTT Network Connection Address is in format

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

or

```
mqttts://<hostname>:<port>
```

where the hostname comes from the broker. The default setting "localhost" requires you to have a broker running on the same machine.



For MQTT connections, you will need a broker in order to configure a working connection. If you do not have your own broker, you can find broker options from <http://mosquitto.org/> and <https://www.hivemq.com/>. You can also test an online broker at <http://test.mosquitto.org/>.

UDP Connection Addresses should look like this for multicast or unicast addresses

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

Standard multicast addresses can be chosen from 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.

For unicast the address specifies the target computer and it can be either an IP address or a hostname that converts to a valid IP address.

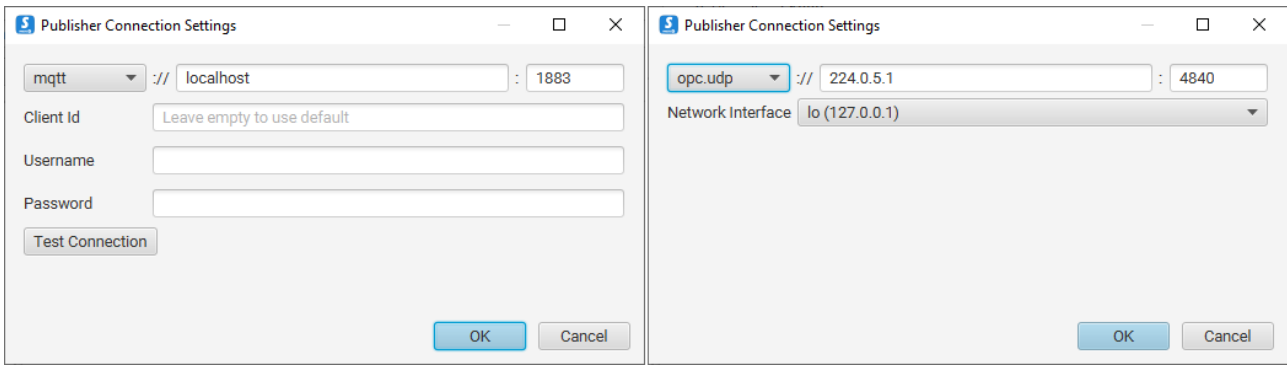


Figure 26. The dialog for configuring the connection settings for the Publisher.

After testing that the connection is working, you can click **OK** and move on to add and configure *WriterGroups*. These *WriterGroups* are used to define the *DataSetWriters* that write the *DataSetMessages* which will be published. To get you started, the *Publisher* already contains one *WriterGroup* that defines one *DataSetWriter* for Events and one for Variables. The same **+** and **-** buttons can be used to add and remove *WriterGroups* as well as *DataSetWriters*.

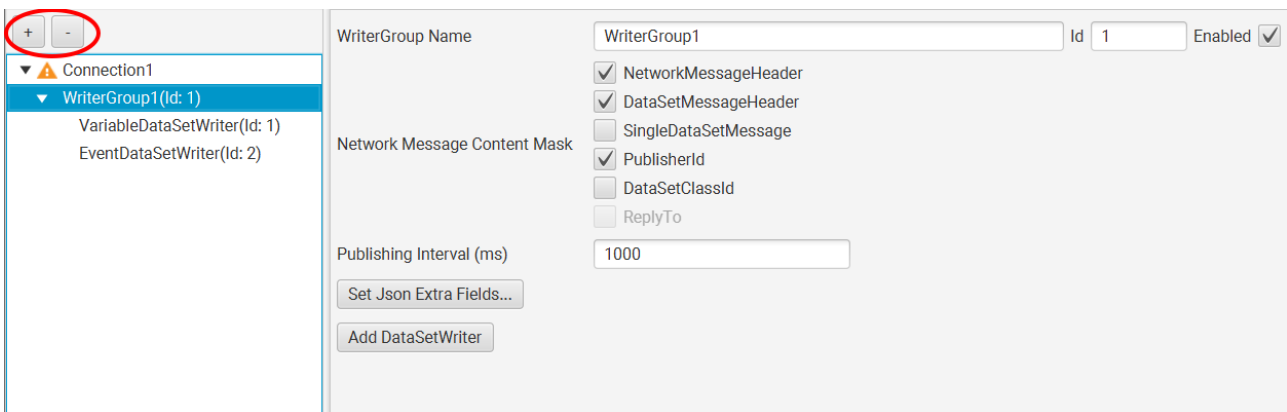


Figure 27. You can add, configure and remove *WriterGroups* that write the *DataSetMessages* that will be sent to the PubSub Network.

When you are satisfied with the generic settings of the *WriterGroup*, you can choose to **Add DataSetWriter** by clicking the corresponding button. New *DataSetWriters* will be automatically added under the selected *WriterGroup* and you can configure them by selecting them from the list on the left.

As shown in [Figure 28](#), there is a warning sign indicating that the configuration is not sufficient yet. The configuration view on the right shows that the *DataSet* still needs to be selected. Other important things to configure for a new *DataSetWriter* are *Queue Name* and *Metadata Queue Name*. The *Queue* corresponds to the MQTT Topic that you wish to publish the *DataSet* to. It is not used for UDP connections. The applications will automatically generate the queues in a way that follows the guidelines for default values found in the specification. If you wish to write your own queue names select **Custom Names**.

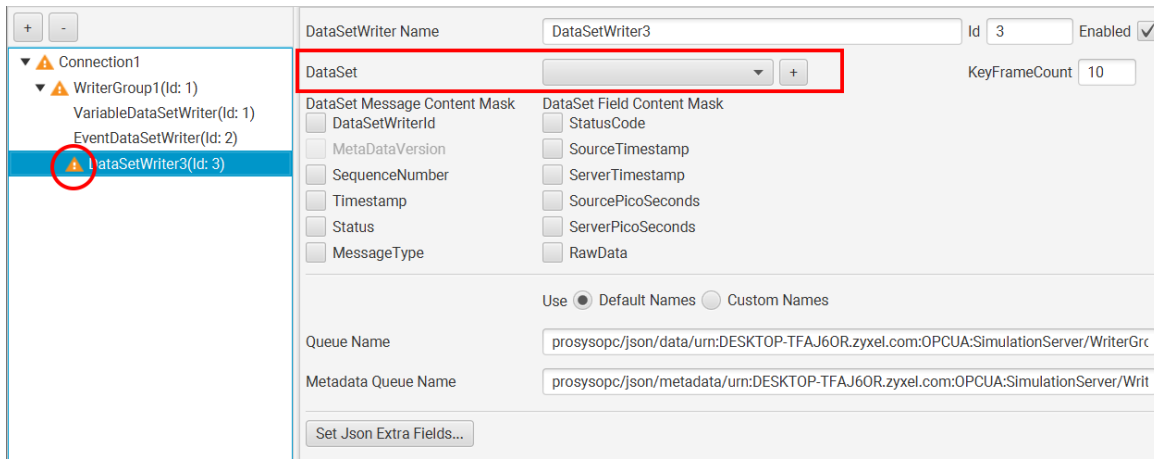


Figure 28. Freshly added DataSetWriter.

You can select the DataSet for your *DataSetWriter* from the drop-down selector highlighted in Figure 29. Simulation Server offers you two options by default: *SampleEventDataSet* and *SampleVariableDataSet*.

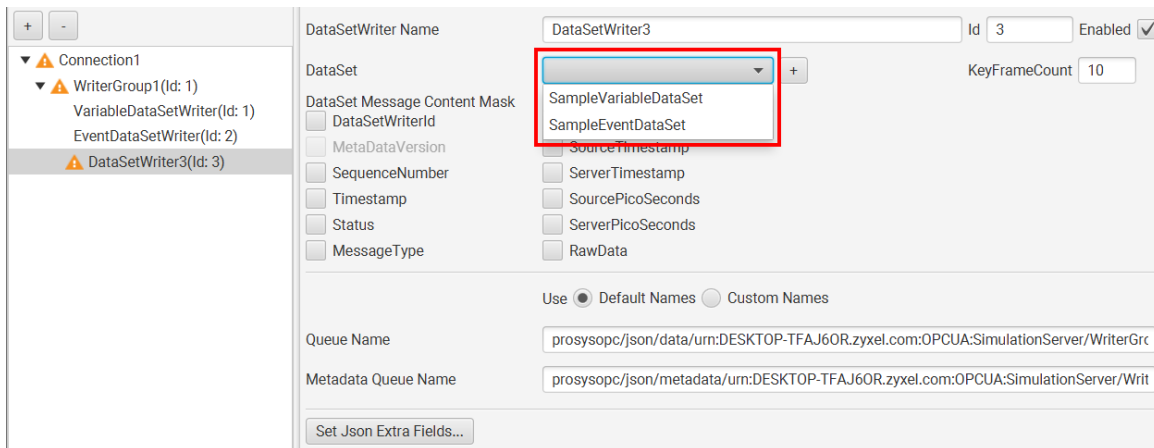


Figure 29. Two DataSet options.

Adding custom DataSets (Professional Edition only)

If you do not find the DataSet you require from the drop-down selector, you can add your own DataSet by clicking the + button next to the selector and selecting the type of DataSet you want to add. This action will bring you to either *Variable DataSets* Tab or *Event DataSets* Tab, depending on which DataSet type you selected. See [Variable DataSets](#) or [Event DataSets](#) for more information on the tabs.

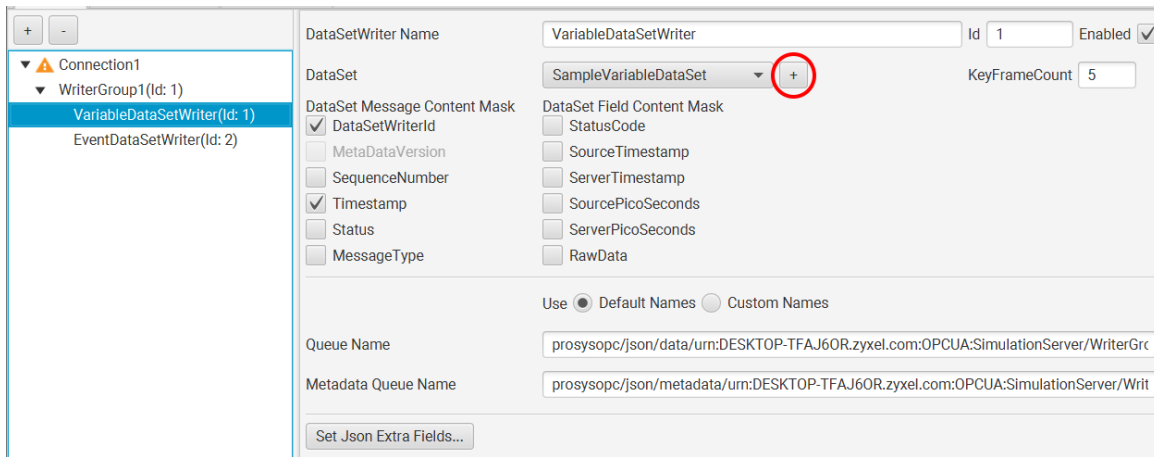


Figure 30. You can add a custom DataSet by clicking the + button.

Variable DataSets View

In the *Variable DataSets View* (Figure 31) you can add and remove DataSets as well as configure their Fields whose values will be written to the DataSetMessages. Those messages are then published to the PubSub Network. In the Free Edition, you only have the default *SampleVariableDataSet* that can contain up to six Fields. The Professional Edition allows adding new Variable DataSets as well as more Fields per DataSet. To make room for new Fields, you can click the **Remove** button to remove Fields that you do not wish to be published.

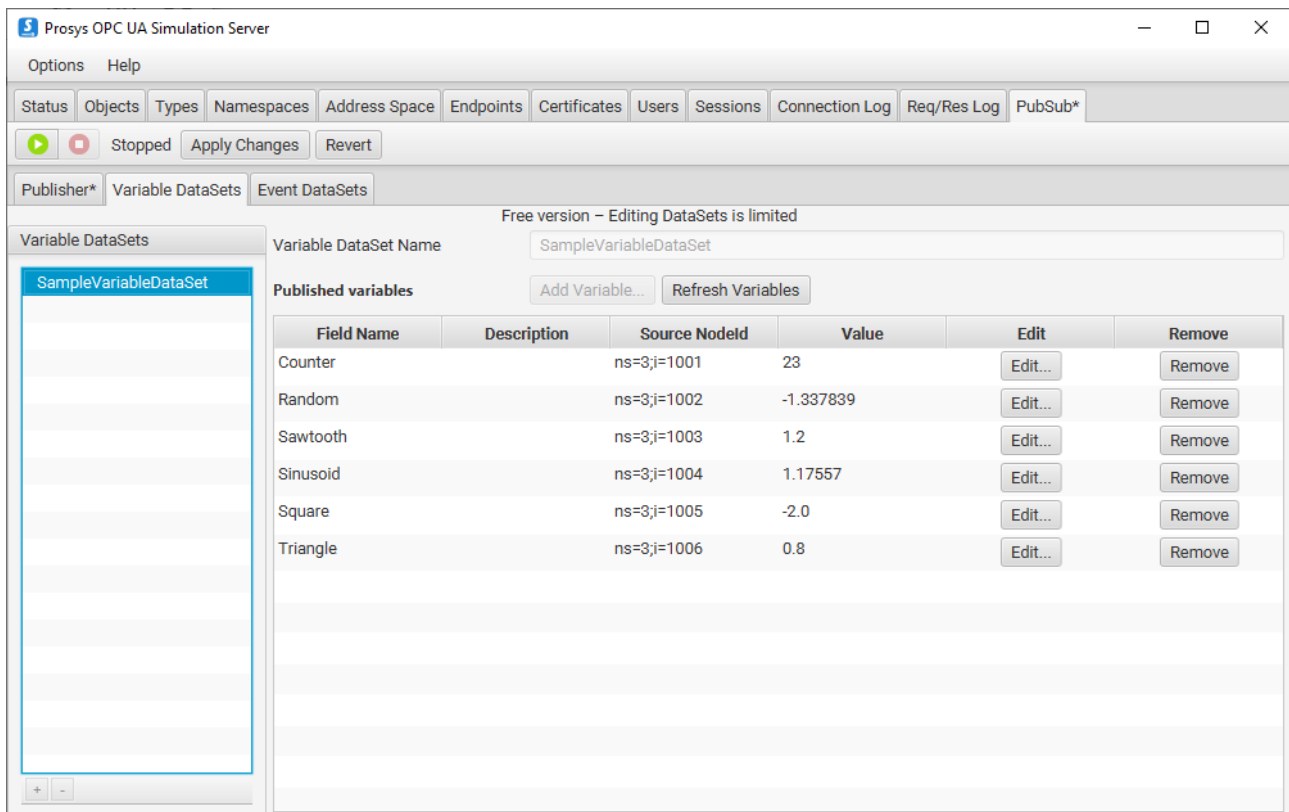


Figure 31. Variable DataSets View.

Adding and removing Variable DataSets (Professional Edition only)

You can add and remove Variable DataSets using the + and - buttons. When + is clicked, a new Variable DataSet is added to the DataSet list on the left. You can change the name of the DataSet as well as other settings of that DataSet by selecting it in the list. Clicking the - button will remove the selected DataSet from the list.

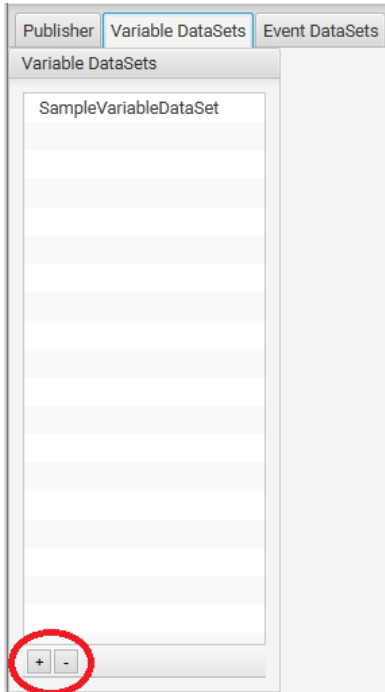


Figure 32. Add or remove a Variable DataSet.

Adding Fields to a Variable DataSet

By clicking the **Add** button, you can open a dialog (see [Figure 33](#)) that lets you browse the server and select the Nodes whose data you want the *Publisher* to publish. Selecting a Node, you can observe its attributes in the middle of this dialog. You can add the Node to the DataSet by either dragging it to the *Nodes to be added* table on the right or right-clicking the Node and selecting **Add**. Selections can be removed from the table by right-clicking and selecting **Remove**.

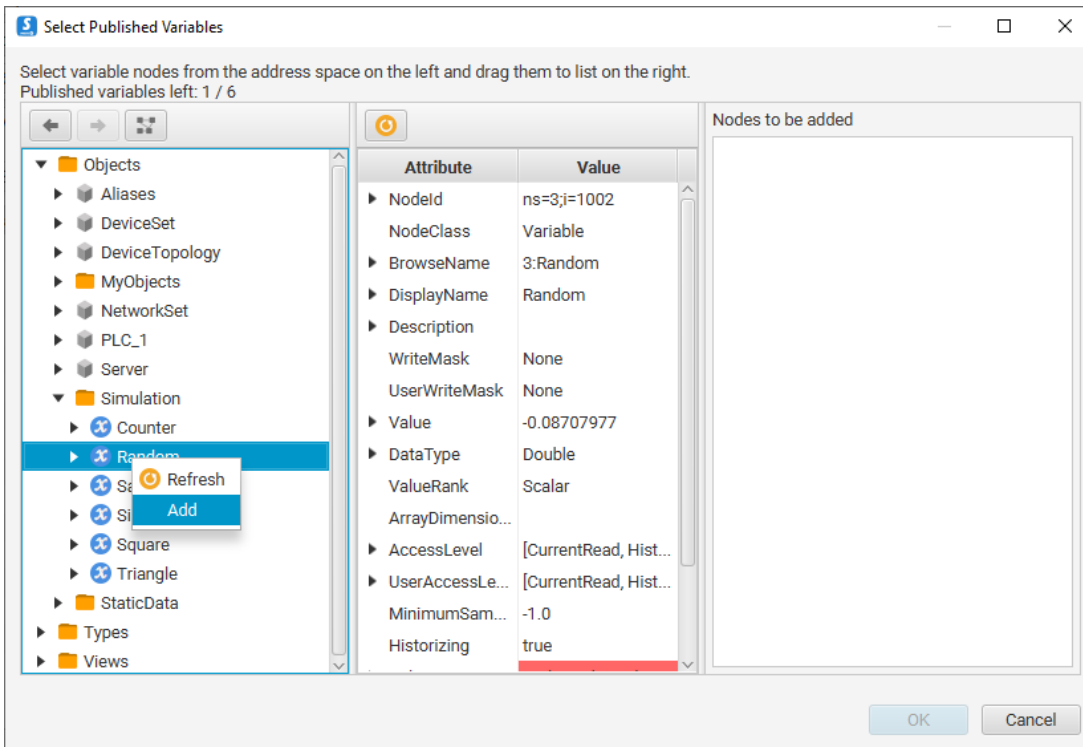


Figure 33. The dialog for selecting Variables to publish.

When you are satisfied with your selections, you can click **OK** and the selected Nodes will be added to the DataSet. If you do not want to add the selected Nodes to the DataSet, you can click **Cancel** and nothing will be added.

Event DataSets View

In the *Event DataSets View* shown in [Figure 34](#) you can select which Node will be monitored for EventNotifications, and which Fields will be included in the DataSetMessage.

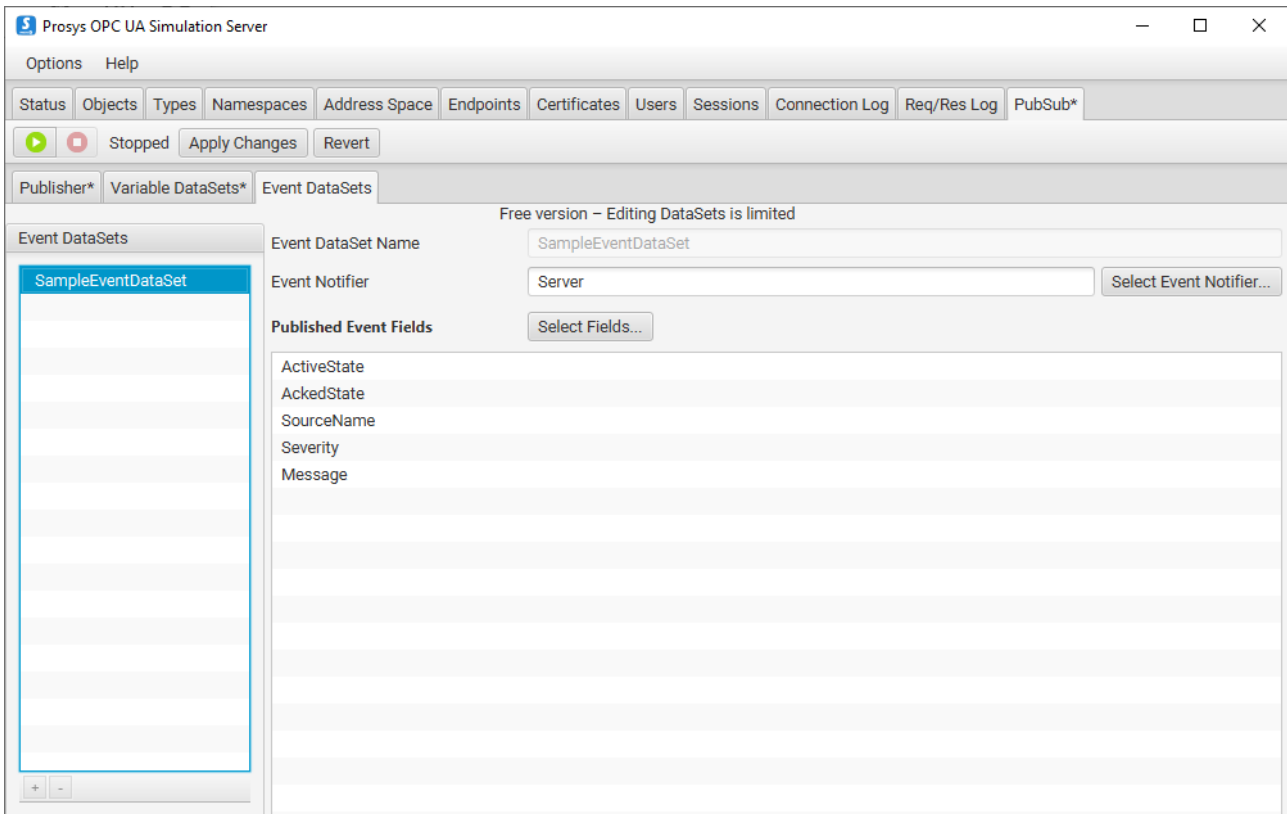


Figure 34. Event DataSets View.

By clicking **Select Event Notifier**, you can open a dialog that lets you browse the Address Space for Event Notifiers. By default, the Server Object is selected, which means that every Event triggered inside the Server will be published. If you wish to get EventNotifications only from a smaller subset of Nodes, you can select an Event Notifier that is under Server in the Address Space. When you click **OK**, the selected Node will be set as the EventDataSet’s Event Notifier.

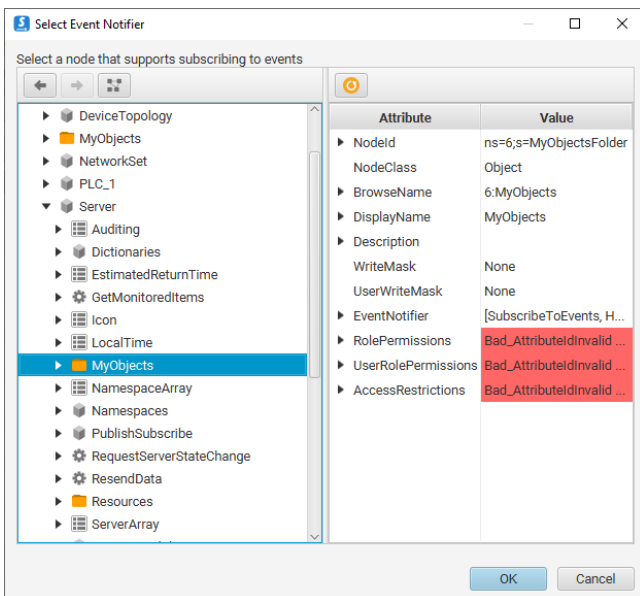


Figure 35. Event Notifier selection dialog.

After selecting the Event Notifier, you can select which Fields the DataSetMessage should contain. This can be done by clicking **Select Fields** and selecting all appropriate Fields in the dialog shown in [Figure 36](#). The selected Fields will be added to the list of Fields when you click **OK**. To remove Fields from the list,

you need to click **Select Fields** again and un-select the Fields that are to be removed.

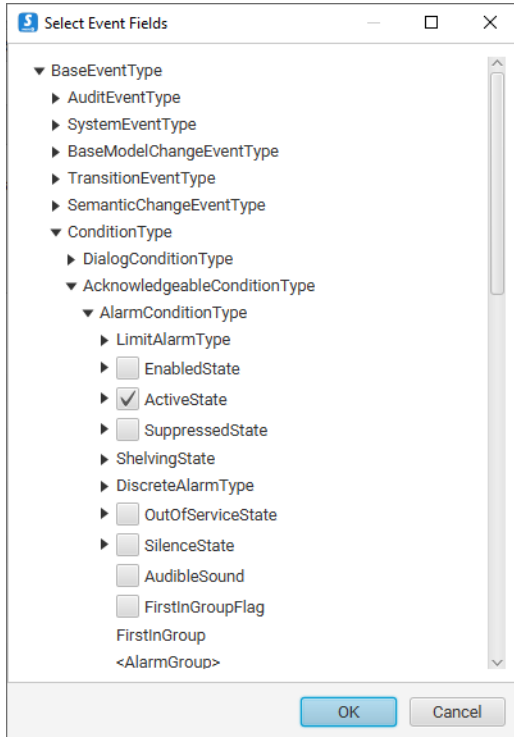


Figure 36. Event Fields selection dialog.

Adding and removing Event DataSets (Professional Edition only)

You can add and remove Event DataSets with the buttons highlighted in [Figure 37](#). Adding a new DataSet automatically adds it to the *Event DataSets* list, and you can start editing it by selecting it from the list. Removing a DataSet immediately removes it from the list.

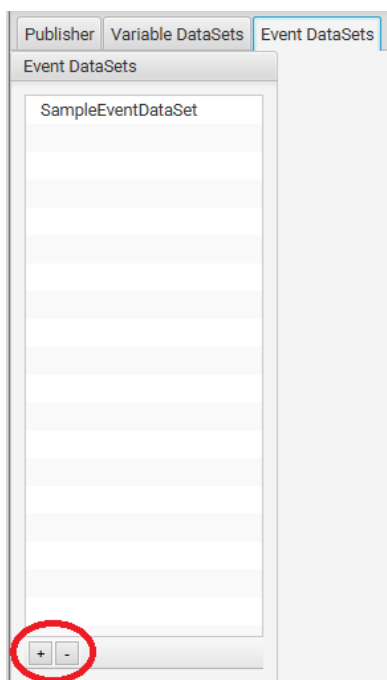


Figure 37. Add and remove Event DataSets.

Device View

The *Device View* is accessible in the [Open Industry 4.0 Mode](#), and can only be used in the Professional Edition.

In this view, you can define and simulate a device that can then be used to simulate Open Industry 4.0 Open Edge Computation (OEC). As you can see in [Figure 38](#), you can define a device that contains Vendor Nameplate and Device Health information, as defined by the OPC 10000-100 Devices Specification (<https://reference.opcfoundation.org/DI/docs/>).

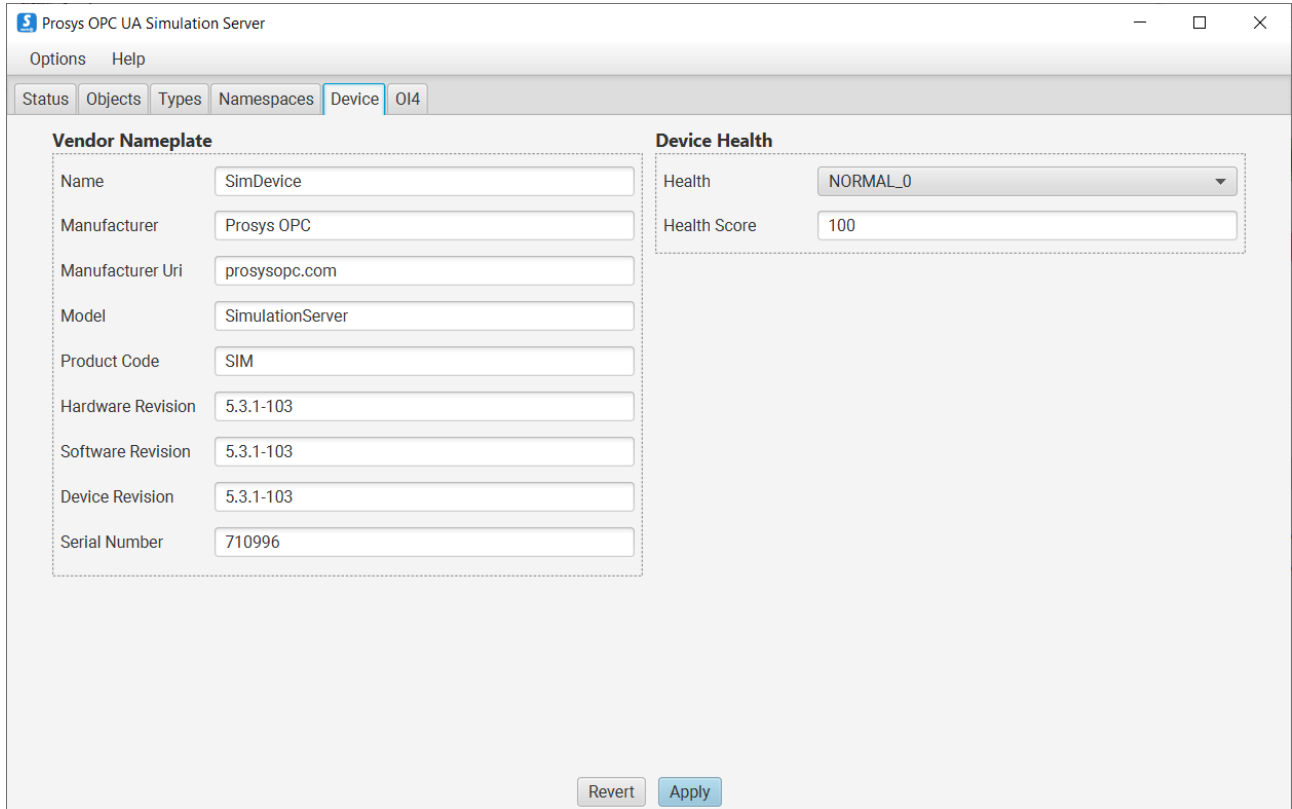


Figure 38. Device View for simulating OEC.

OI4 View

The *OI4* View is accessible in the [Open Industry 4.0 Mode](#), and can only be used in the Professional Edition.

In this view, you can define the MQTT connection and enable message types for the Open Industry 4.0 Open Edge Computation (OEC) connection.

In order to simulate the OEC, the server must contain the Device Information Model. You can add it yourself in the [Namespaces View](#) by selecting the *Add Device Information Model*, or by clicking the 'Import' button in the *OI4* View.

In [Figure 39](#), we can see that this view contains a toolbar for simulation control and settings for the OEC connection. You can define the MQTT connection address similarly to how it is done in the [PubSub View](#). You can also define which Open Industry 4.0 DataSets are published and how often.

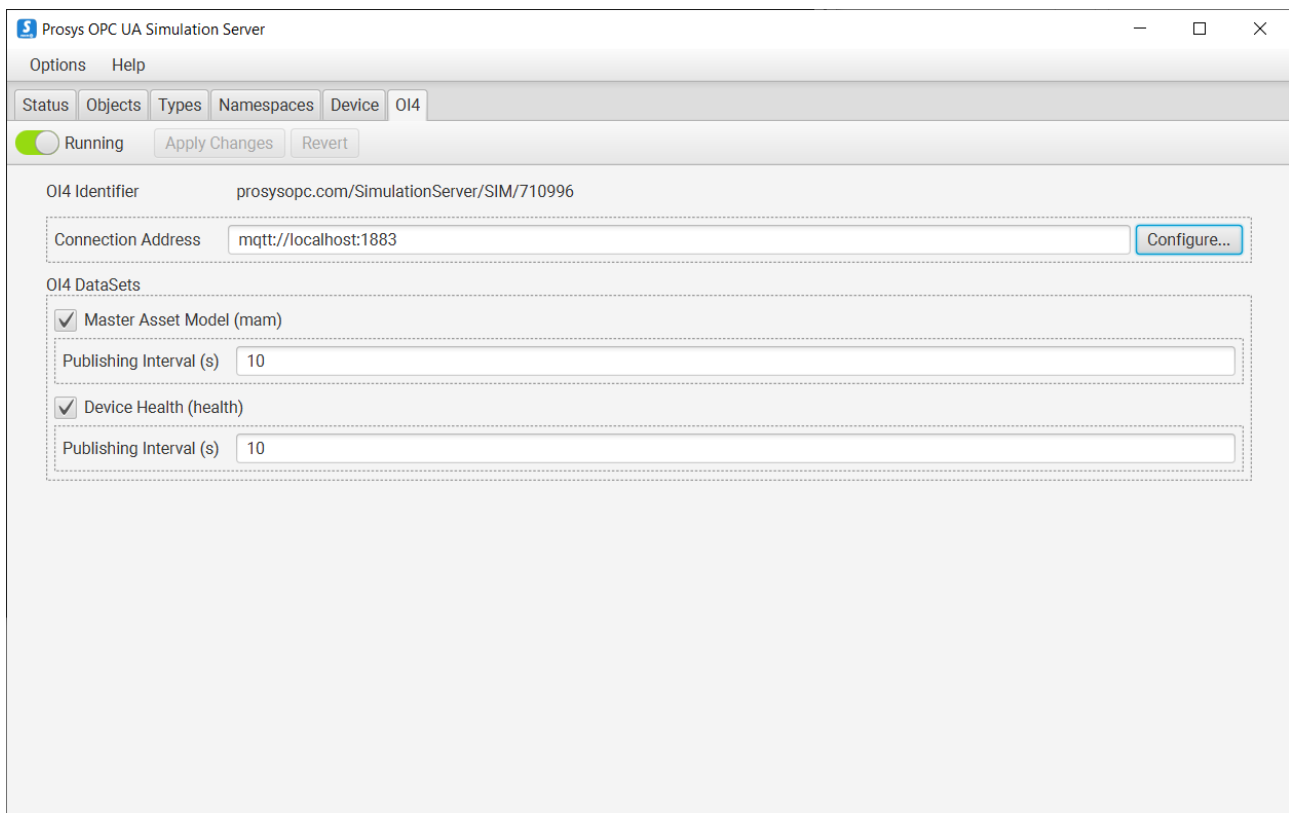


Figure 39. Device View for simulating OEC.

To start publishing to the OEC MQTT Broker, click the switch on the toolbar to change the status to running.

OPC UA Server Explained

The Simulation Server implements an OPC UA server that contains simulated values for demonstration and testing purposes. The following chapters explain the key characteristics of the OPC UA server.

Address Space

OPC UA applications provide the data that they have available for OPC UA client applications from the *OPC UA Server Address Space*. This helps the client applications locate all relevant data from the server when they don't have any prior knowledge about it.

The OPC UA client applications identify data in the OPC UA server using Node Identifiers (NodeIds). NodeIds are used to uniquely identify all information (Nodes, which can be Objects, Variables or various Types) in the server. For example, they are used when the client sends *read* or *write* requests to the server. If the client applications don't have the NodeId of a certain variable available, they can *browse* the server address space to find it.

You can use the [Prosyst OPC UA Client](#) to explore the address space visually and access the information of the OPC UA Server. You can also use the [Address Space View](#) to verify how the address space will look for the client applications.

Prosyst OPC UA Simulation Server follows the standard outline of OPC UA Server Address Space definition.

The address space contains three main parts, available as OPC UA Folders:

- Objects
- Types
- Views

Objects

The Simulation Server defines the following Objects:

Server

a standard Object that provides the status and capability information about the server.

MyObjects

folder, contains a sample Object, *MyDevice* that simulates a simple device with five components:

MyEnumObject

a simple variable containing an enumeration value using the custom enumeration data type named *MyEnumType*.

MyLevel

a Variable (DataType=Double) for simulated level measurement.

MyLevelAlarm

an alarm Object corresponding to *MyLevel*. New events are sent from the server when the value of *MyLevel* exceeds the alarm limits defined in the alarm Object.

MyMethod

a sample method that can be called from the client applications.

MySwitch

a simple switch Variable (DataType=Boolean) that can be turned on and off.

StaticData

folder, includes Variables of various data types that change only when written to.

Simulation

contains default objects and variables.

Types

OPC UA servers provide complete type information for its instances in the address space. These can be found in the *Types* folder. The Simulation Server includes the following types:

DataTypes

defines all the DataTypes that are used in the server address space.

EventTypes

define all the Events that appear in server address space.

ObjectTypes

includes type definitions for all the Objects in the server address space.

ReferenceTypes

defines all the Reference types that appear between Nodes.

VariableTypes

includes type definitions for all the Variables in the server address space.

Views

The Simulation Server does not define any Views, as defined in OPC UA specification. Thus you can find an empty folder there.

File Locations

The application saves its settings to a folder in path *(user.home)/.prosysopc/prosys-opc-ua-simulation-server*. The *(user.home)* is the location of the current user's home folder. If you want to reset to default settings, just delete the settings file in the folder. Settings are saved automatically when the application is closed. The certificates used in OPC UA communication are saved to the *PKI* subfolder, and the certificates used for User Authentication are stored in the *USERS_PKI* subfolder.

Previous Versions

Version 3.x.x and earlier

The earlier versions of the application (3.x.x and earlier) stored the settings in *(user.home)/.prosys/SimulationServer*. This version of the application does not use these settings so they can be removed from the file system.

Version 4.x.x

The 4.x.x version saved its settings in the same folder as the current version, but the file **settings.xml** and the folder **SimConfig** at the root application settings folder are no longer used and can be removed.

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.

Certificate Stores

The main settings folder also contains the certificate store folders used to keep the certificates known by the application. The following directories are used:

- **PKI** for Application Instance Certificates.
- **USERS_PKI** for User Certificates.

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

The certificate stores contain a few sub-folders:

- **certs** for the trusted certificates.
- **rejected** for the rejected certificates.

Prosyst OPC UA Simulation Server will reject all unknown certificates by default, unless they are signed by a Trusted Issuer Certificate that is placed in the **certs** folder.

Although you can modify the stores directly on the disk, it is usually better to use the [Certificates View](#) to define the trusted Application Instance Certificates. You cannot yet use the user interface to trust or reject User Certificates, so the only option is defining the trusted certificates in the folders.

Troubleshooting Common Problems

Common Problems

My Simulation Server won't start

You can only run one instance of Prosys OPC UA Simulation Server at once. Sometimes processes might continue running even after closing the application, which makes it impossible to start the application since it is already running on the background. In a case like this, open your Task Manager and look for *UaSimulationServer.exe*. When you find this task, end the task, and try starting the application again.

Trying to Connect to the Simulation Server with Secure Settings Produces Errors

When trying to establish a secure connection between a client and Prosys OPC UA Simulation Server for the first time, you will run into this problem. Your connection fails. This happens because the server does not trust the client by default. To fix the issue, go to the *Certificates* tab, right-click the certificate of your client application, and select *Trust*. Now, try connecting to the server again.

Finding the Log File

In some cases where the error is not obvious, you might benefit from looking at the log files. The log files contain valuable information of errors and events during runtime of the Simulation Server.

As mentioned in [File Locations](#), you can find the log file from *(user.home)/.prosysopc/prosys-opc-ua-simulation-server/log*. The latest file is called *simulationserver.log*.

Contact Us

When you have tried troubleshooting and fixing the problem yourself, but nothing seems to work, you can contact our support team to get help.

Free License Users

Forum-based support is available at <https://forum.prosysopc.com/forum/opc-ua-simulation-server/>.

Professional License Users

In addition to forum-based support, you can also use our email-based support by sending us an email to: simulation-server-support@prosysopc.com with a description of your problem. We recommend that you attach the log file to the email to make resolving the problem faster.