

OPC UA
Simulation Server

User Manual

Version 2026.2.4

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
  - Project persisted . . . 5
  - General . . . 6
  - Appearance . . . 6
- Projects . . . 7

Status View . . . 8
Simulation Views . . . 10
Objects View . . . 10
- Simulation Control . . . 11
- Node Tree . . . 11
- Configuring a Variable or Object . . . 13
- Value Simulation . . . 14
- Defining Values . . . 16
- Event Simulation (Professional Edition only) . . . 17
- Method Simulation (Professional Edition only) . . . 20
  - Arguments . . . 20
  - Actions . . . 21

Playback (Professional Edition only) . . . 23
- Basic use . . . 24
- Loading datalogs . . . 24
- Navigation and positioning . . . 24
- Time handling . . . 25
- Looping . . . 25
- Overview . . . 26

Types View . . . 26
- Defining Value Simulation for Types . . . 27

Namespaces View . . . 28
- Exporting a namepace (Professional Edition only) . . . 30
- Importing an information model (Professional Edition only) . . . 30

Address Space View . . . 32
Endpoints View . . . 33
- UA TCP Transport Protocol . . . 33
- Security Policies . . . 34
- Bind Addresses . . . 34

- Registering to Local Discovery Server . . . 34
- Reverse Connections . . . 34
- Applying Changes . . . 35

Certificates View . . . 36
Users View . . . 37
- Adding and Removing Users . . . 38

Sessions View . . . 39
Connection Log View . . . 40
Req/Res Log View . . . 41
PubSub View . . . 42
- Publisher Connection View . . . 43
  - Adding custom DataSets (Professional Edition only) . . . 45
- Variable DataSets View . . . 46
  - Adding and removing Variable DataSets (Professional Edition only) . . . 47
  - Adding Fields to a Variable DataSet . . . 47
- Event DataSets View . . . 48
  - Adding and removing Event DataSets (Professional Edition only) . . . 50

Device View . . . 51
OI4 View . . . 52
Headless and Container Mode (Professional Edition only) . . . 53
- Starting the server . . . 53
  - Windows . . . 53
  - Linux . . . 53
  - macOS . . . 53
  - Command-line options . . . 54
  - Exit codes . . . 54
- Saving the Project . . . 55
- Running Playback from the command line . . . 55
- Controlling the server over OPC UA . . . 56
- Client certificates in headless mode . . . 57
- Docker Container distribution . . . 57

OPC UA Server Explained . . . 59
- Address Space . . . 59
- Objects . . . 59
- Types . . . 60
- Views . . . 60

File Locations . . . 61
- Previous Versions . . . 61
  - Version 3.x.x and earlier . . . 61
  - Version 4.x.x . . . 61
- Application Logs . . . 61
- Certificate Stores . . . 61

Troubleshooting Common Problems . . . 63
- Common Problems . . . 63
  - My Simulation Server won't start . . . 63
  - Trying to Connect to the Simulation Server with Secure Settings Produces Errors . . . 63
- Finding the Log File . . . 63
- Contact Us . . . 63
  - Free License Users . . . 63

Professional License Users 63

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.

If upgrading from version 2026.1.2 or earlier, the settings that are now stored per computer (*Bind Addresses*, IPv6, Local Discovery Server registration and Reverse Connections) and the Preferences that are now stored in the Project (see Preferences Window) are reset to their default values.

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

Prosys 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:

- Playback logged device data on 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
- Possibility to add and simulate Methods 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 preferences are grouped by where they are stored: the *Project persisted* preferences are saved in the Project and travel with it, while the *General* and *Appearance* preferences belong to this computer and apply to all Projects.

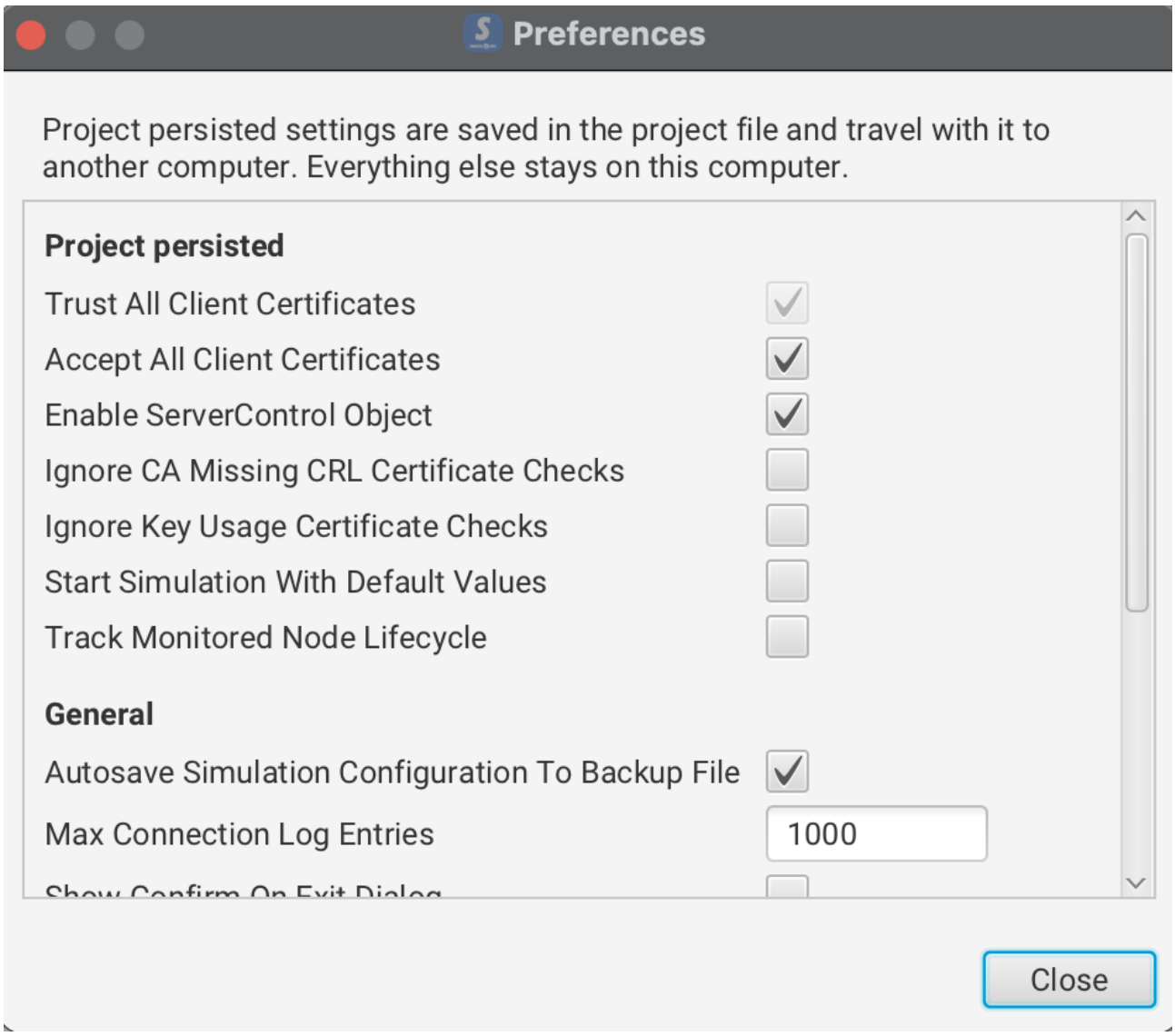

Figure 1. The UI and the application's functionality can be customized in the Preferences window.

The available preference options are briefly described below:

Project persisted

Trust All Client Certificates

Trust any client certificate without manual approval, but still validate it normally, so an expired or otherwise invalid certificate is rejected. This is a security downgrade intended for unattended testing, such as headless mode. Leave it off for normal use.

Accept All Client Certificates

Accept any client certificate without validating it at all, so even an expired, revoked or malformed certificate is accepted. This is a larger security downgrade than *Trust All Client Certificates* and overrides it. Leave it off for normal use.

Enable ServerControl Object

Expose the ServerControl Object, whose Methods let any connected client start and stop Playback and the value simulation. Turn it off on a server that untrusted clients can reach. Takes effect on the next server restart. See Controlling the server over OPC UA.

Ignore CA Missing CRL Certificate Checks

Enable or disable ignoring CA missing CRL checks when validating certificates.

Ignore Key Usage Certificate Checks

Enable or disable ignoring Key Usage checks when validating certificates.

Start Simulation With Default Values

Reset the simulation signals to their default value when the application starts.

Track Monitored Node Lifecycle

Send a Bad_NodeIdUnknown StatusCode when a monitored Node is removed, and resume notifications if the Node reappears. Enabling this option may impact performance.

General

Autosave Simulation Configuration To Backup File

Enable or disable the autosaving of the simulation configuration to a separate backup file (happens every 5 min).

Max Connection Log Entries

Set the maximum amount of entries to show in the Connection Log view.

Show Confirm On Exit Dialog

Enable or disable the Exit confirmation dialog.

Appearance

Link Objects And Types Views

Link Types View to automatically select the type Node of the selected instance in the Objects View.

Show DataType Name Tooltip

Enable or disable the hovering DataType name tooltip for Variables in views (is overridden by the *Show Type Name Tooltip* option).

Show ModellingRule

Show or hide the modelling rule postfix '::ModellingRule' for InstanceDeclarations in the Types View.

Show Type Name

Enable or disable the type name postfix '::TypeName' for instance Nodes in the Objects View.

Show Type Name Tooltip

Enable or disable the hovering type name tooltip for instance Nodes in the Objects View.

Show Value Plot

Show or hide the view that shows the value plot shape and range in a graph with the current value.



Some preferences are applied only after a restart. Hover over a preference to see its description.

Projects

Prosys OPC UA Simulation Server allows you to open and save your configurations as Projects. Projects are managed from the File menu. Projects contain the Address Space, the simulation and Playback configuration, the Endpoint settings (port, server name and security options), the PubSub and User settings, and the *Project persisted* preferences.

Some settings are not stored in the Project, because they would be wrong on another computer: the *Bind Addresses*, IPv6, Local Discovery Server and Reverse Connection settings of the Endpoints View. These are stored per computer and always take precedence over a Project that was created elsewhere. Trusted certificates and the *General* and *Appearance* preferences are also shared across all Projects.

If a Project cannot be loaded because the file is damaged, the desktop application does not fail to start. It falls back to the backup file, or to the default configuration if that is not usable either, and tells you which one was loaded. If a Project was recovered from the backup file, save the Project to keep the recovered changes. A headless server instead refuses to start, so that it never serves the default configuration unnoticed.

A namespace that cannot be loaded is left out of the server, and the rest of the Project is loaded normally. The data of such a namespace stays in the Project file and is not modified when you save, so you can correct the NodeSet, then remove and re-import the namespace.

Status View

The Status View (Figure 2) 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 also set the Server Status yourself by clicking it and selecting a state, for example *Failed*, from the list. This is useful for testing how a client application reacts to a server that is not running normally. It only changes the value of *ServerStatus.State* that the clients read; it does not affect the actual functionality of the server.



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 uses the UA TCP (*opc.tcp*) transport protocol. 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.

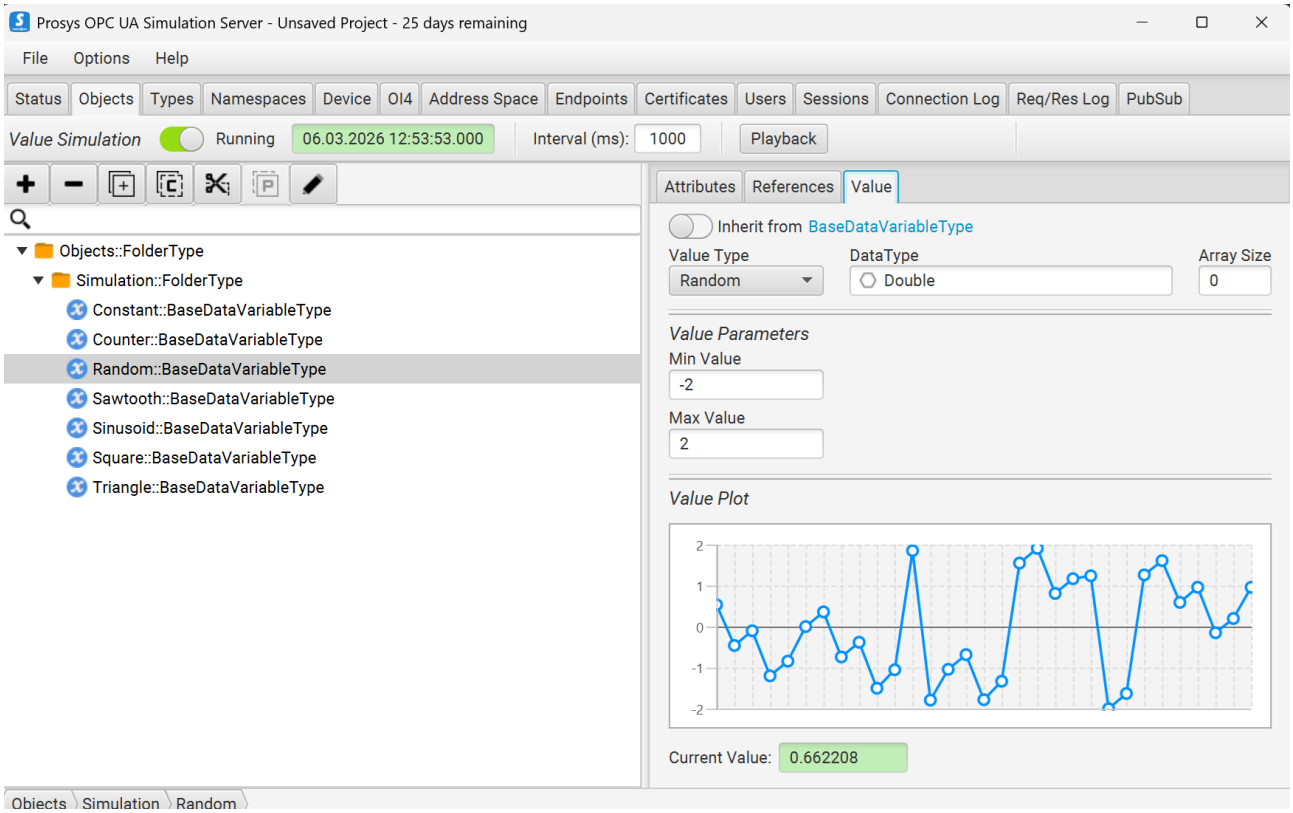

Figure 3. The Objects View enables the configuration of custom Objects and Variables.

The *Objects View* is divided into three parts (see Figure 3).

1. At the top, you will find the Simulation Control.
2. The left-hand side of the view contains the Node Tree.
3. The *Attributes* and *References* of the selected node are displayed in the respective tabs on the right-hand side. Depending on the Node that is selected in the Node Tree, there are also some additional tabs:
    a. Variables have a Value tab for Value Simulation
    b. Objects have an Events tab for Event Simulation (Professional Edition only)
    c. Methods have an Arguments and Actions tab for Method Simulation (Professional Edition only)



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.

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 4. 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.

The simulation control bar also includes a Playback button, that opens the Playback window, which is described in it's own chapter later in this document.

Node Tree

The Node Tree on the left-hand side of the Objects View is a hierarchical tree view of the filtered *Objects* folder. By default, it only shows the contents of the Simulation folder. But you can add your own nodes using the Node Tree Actions, which are available from the tool bar on top of the tree view and also from the context menu that is available with a right click. The actions are context-sensitive, meaning that the behavior of them depend on the selected Node.

The following list briefly describes the available actions:

Add

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

New Folder

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

New Variable

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

New Object

adds a new Object under the selected Node with a user-selected namespace, name, ObjectType,

optional members and ReferenceType.

New Method

add a new Method under the selected Node with a user-selected namespace and Name. This action is available only in the Professional Edition.

New Property

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

Remove

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

Duplicate

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

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

Cut

removes the selected Node and places it on the clipboard.

Paste

adds the Node from the clipboard under the selected Node.

Edit

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:

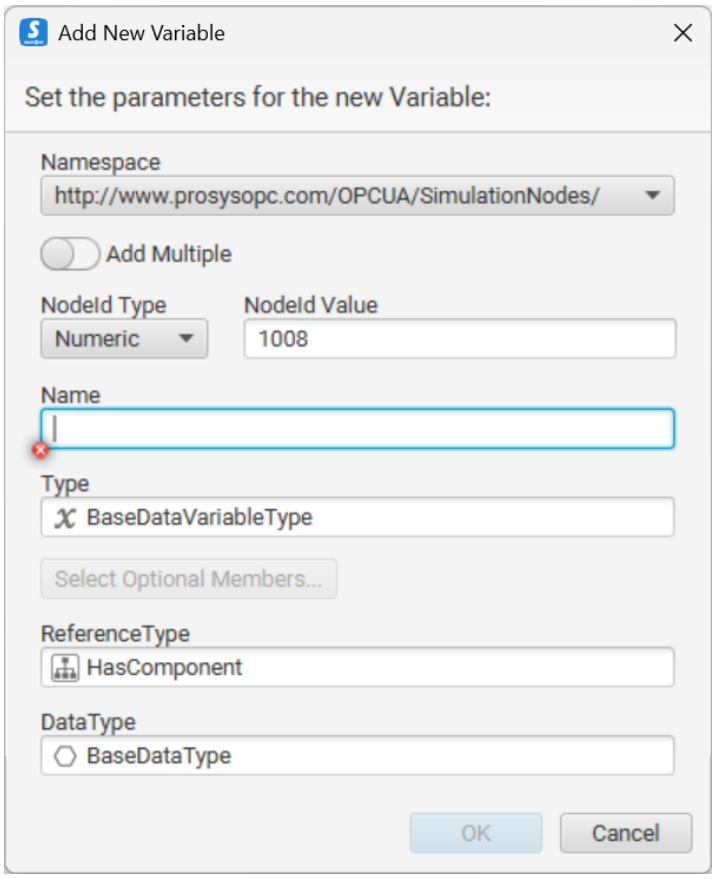

Figure 5. You can configure the parameters of new or existing Variables and Objects.

Namespace

defines the namespace that the Node belongs to.

Add Multiple

select to create multiple instances of the same type.

NodeId Type

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

NodeId Value

defines the value of the NodeId.

Generate numeric NodeIds / Use names as NodeIds (when Add Multiple selected)

selection defines if a numeric NodeId is automatically generated for the added Nodes or if the name is used as a String type NodeId.

Name Separator (when Add Multiple selected)

Defines the separator sign that is used as the delimiter in the list of names that is input in the Names field.

Name

defines the DisplayName and the BrowseName of the Node.

Names (when Add Multiple selected)

list of names that defines the DisplayName, the BrowseName and the NodeId (when Use Names as NodeIds is selected) 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).

DataType

defines the DataType for the value of a Variable.



Some options are not available when editing an existing Node.

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 6) 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).

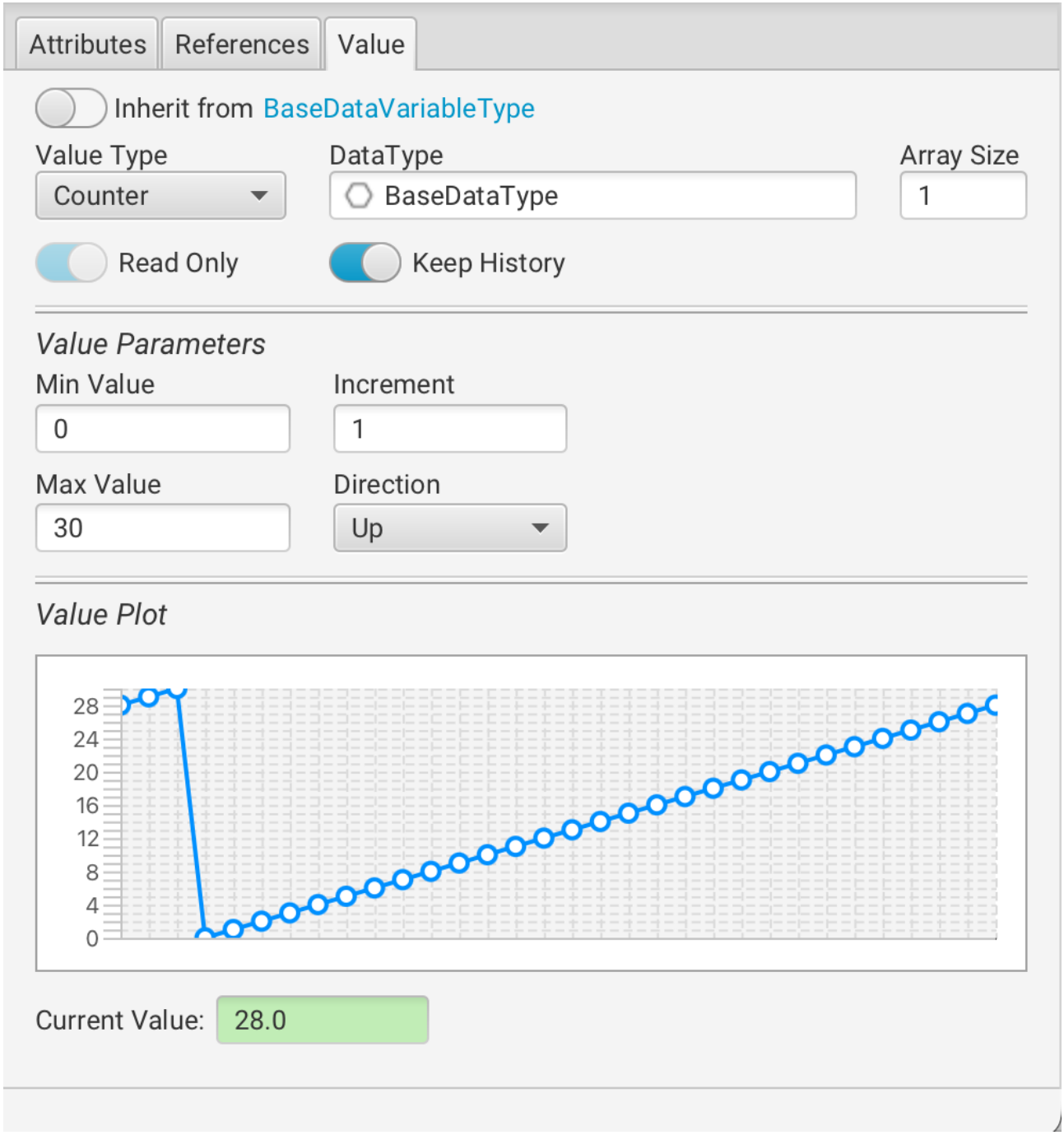

Figure 6. Value tab for defining value simulation in the Objects View.

Keep History and *Read Only* allows you to control the AccessLevel-attribute of a Variable Node. Keep History sets the History Read value while Read Only unsets the CurrentWrite AccessLevel. Inherited nodes also inherit their access level settings and thus cannot be modified. Variables with simulated signals cannot be written into and thus are forced to have Read Only on. History access can also be enabled for multiple nodes at once. Select them in the Objects View and apply “Enable History” or “Disable History” from the context menu.

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*

DataType 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 7, you can see that Objects have the *Events* tab where Variables have the *Value* tab.

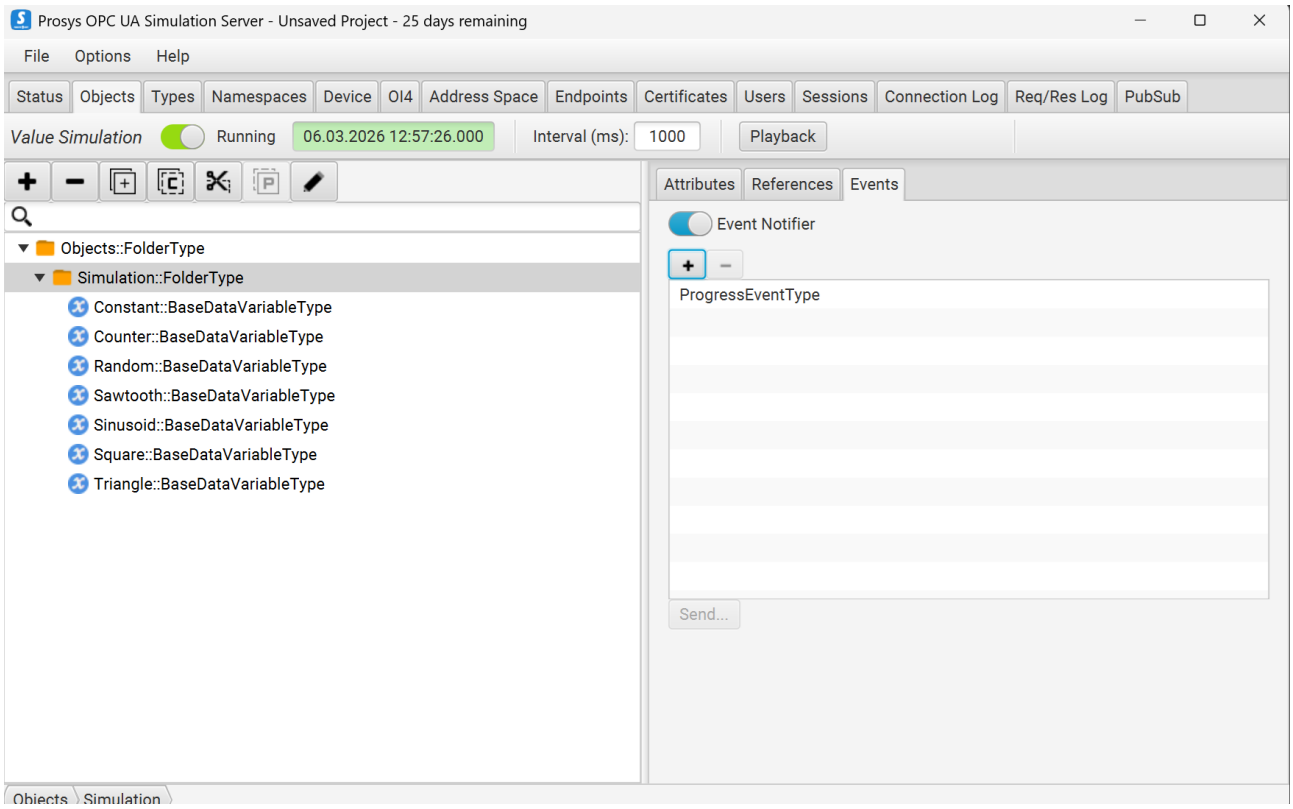

Figure 7. Events tab for simulating events for Objects in the Objects View.

To simulate Events for a selected Object, you must first enable an Event Notifier for it (Figure 7, 1.). Next, you can click the '+' button (Figure 7, 2.) to add a new Event for the Object. The dialog shown in Figure 8 will let you select the type of Event you want to add.

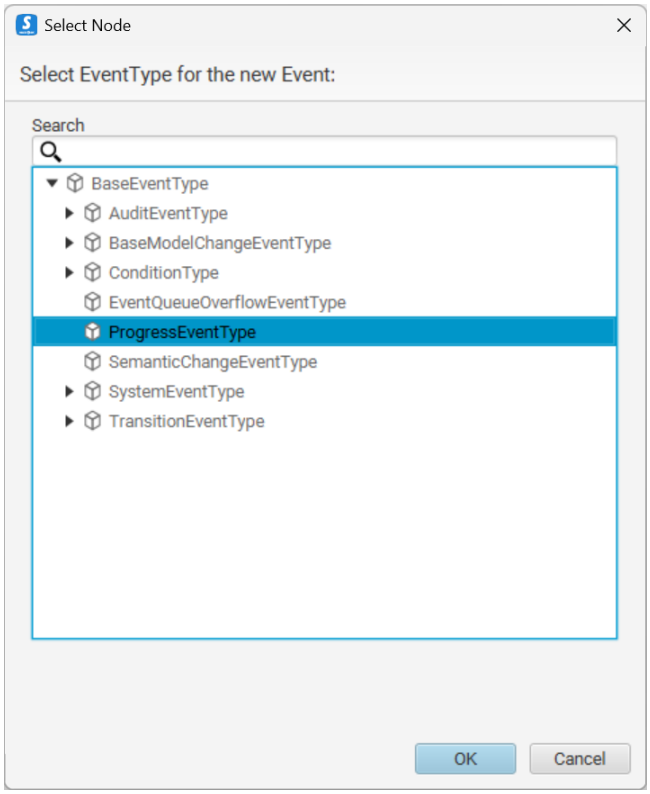

Figure 8. You can select the Event Type for your new Event.

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 9), 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.

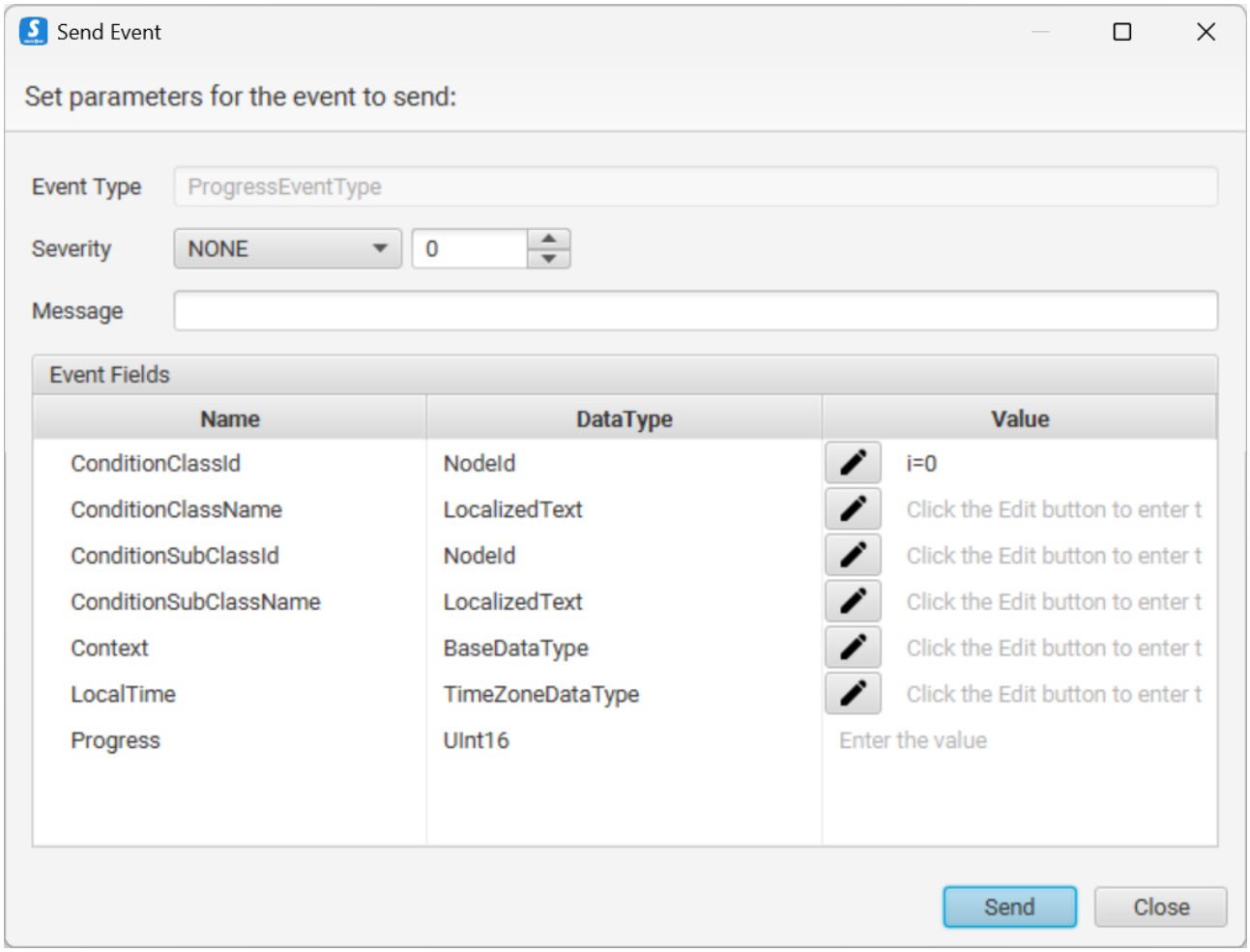

Figure 9. Define the details for your simulated Event.

Method Simulation (Professional Edition only)

In the Professional Edition of the application, you have the ability to add Methods and to simulate the functions of the Methods. When a Method is selected in Objects View, you will find the Arguments and Actions tabs on the right-hand side.

Arguments

The Arguments tab provides an overview of the input and output arguments defined for the Method (see Figure 10). Here, you can see the Name, DataType and ValueRank of each argument.

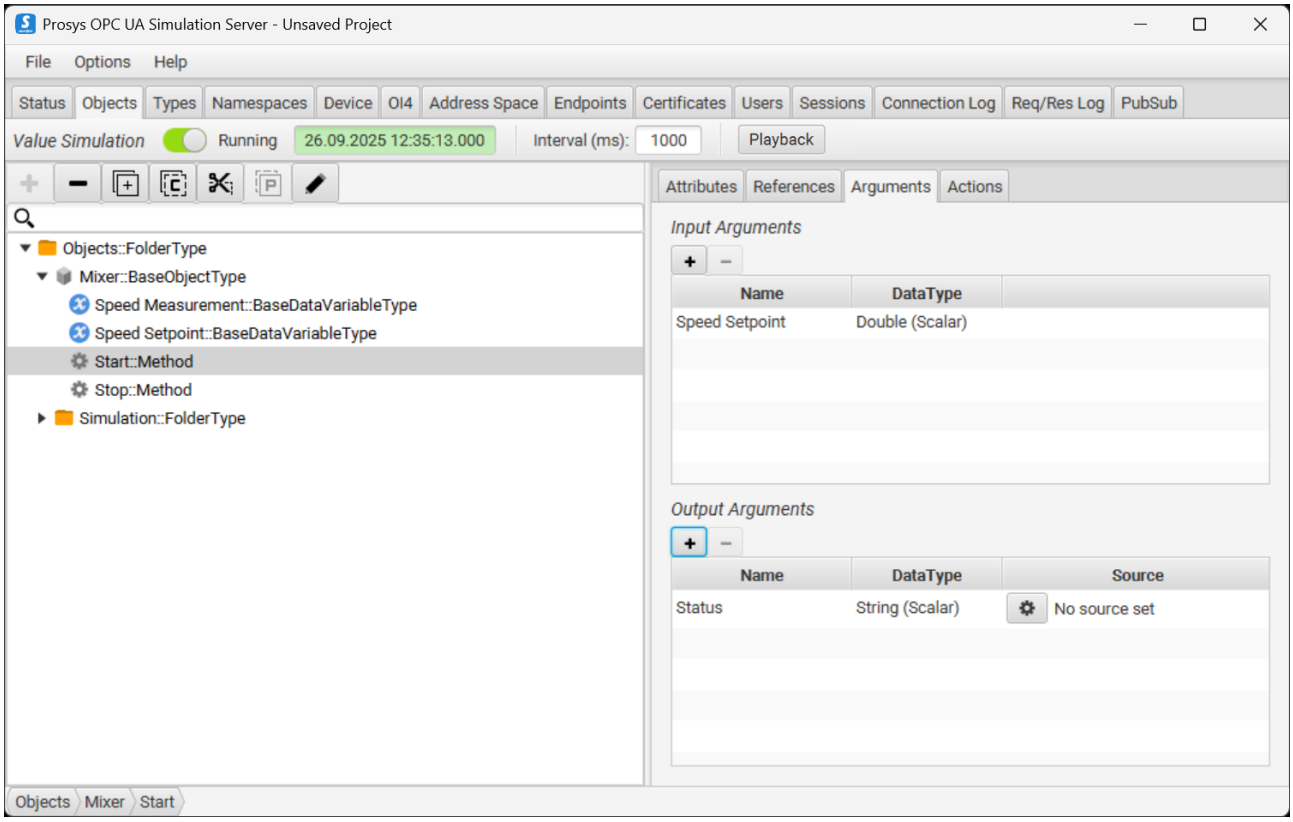

Figure 10. Arguments tab for displaying and simulating arguments for Methods in the Objects View.

For user-created Methods, you can add and remove arguments with the + and - buttons above the input and output argument lists.

To simulate the function of the Method, you can define values for the output arguments that will be received by the client when the Method is called. To do this, simply click the button in the Source column of the Output Arguments table.

The Source option determines the value that will be assigned at the time of the Method call. You have the following choices:

Input

The value is copied from a selected input argument that was provided in the Method call. Inputs are available as sources only when they are compatible with the DataType and ValueRank of the output argument.

Variable

The value is copied from a user-created Variable in the server's address space.

Constant

The value is a given constant value that you provide.

Simulation

The value is the output of a value simulation performed at the time of the Method call. For more information on available value simulations, refer to the Value Simulation section.

Actions

The Actions tab allows defining Actions that will be performed when the Method is called (see Figure 11).

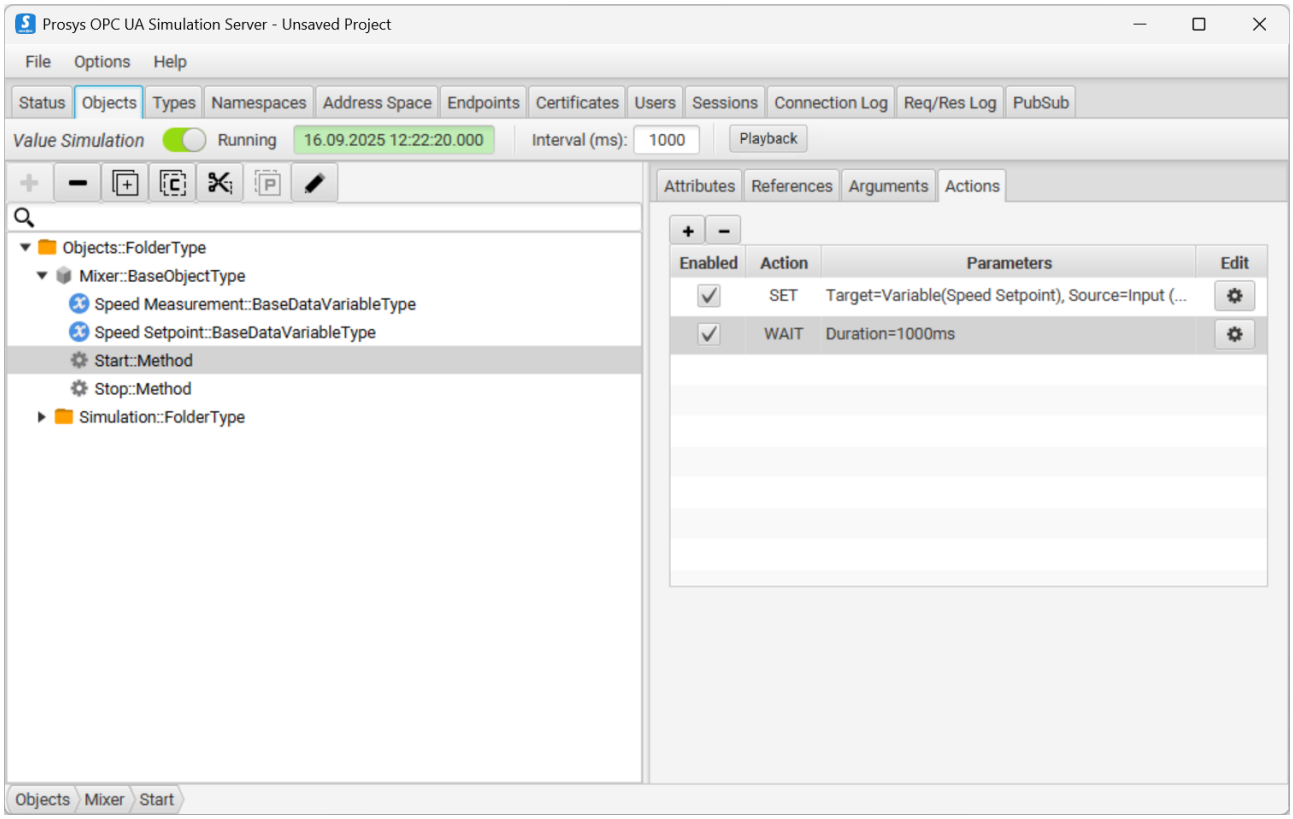

Figure 11. Actions tab is for defining Actions that are performed when the Method is called.

Currently, the available action types are *SET* and *WAIT*.

SET

Allows you to set the value of a Target Variable in the server's address space. The value for the Target Variable is defined by selecting a Source similarly to how Sources of output arguments are selected in the Arguments tab.

WAIT

Allows you to configure the Method simulator to wait for a fixed duration before performing the next action or finishing Method handling and returning its output arguments.

You can *Add*, *Edit* and *Remove* Actions from the context menu, which is opened with a right-click on the Actions list. Actions can also be added and removed with the + and - buttons above the list. *Edit* is also available with double-click.

Actions are performed sequentially in the order they are defined. You can change their order from the

context menu as necessary.

The Enabled column can be used to enable and disable Actions. Only enabled Actions are performed when the Method is called.

Playback (Professional Edition only)

The Playback view is used to replay previously logged device data from CSV files on the Prosys OPC UA Simulation Server. It reads recorded values and Status Codes, writing them back to matching server nodes in chronological order, allowing recorded process behavior to be repeated for testing, demonstrations, and validation.

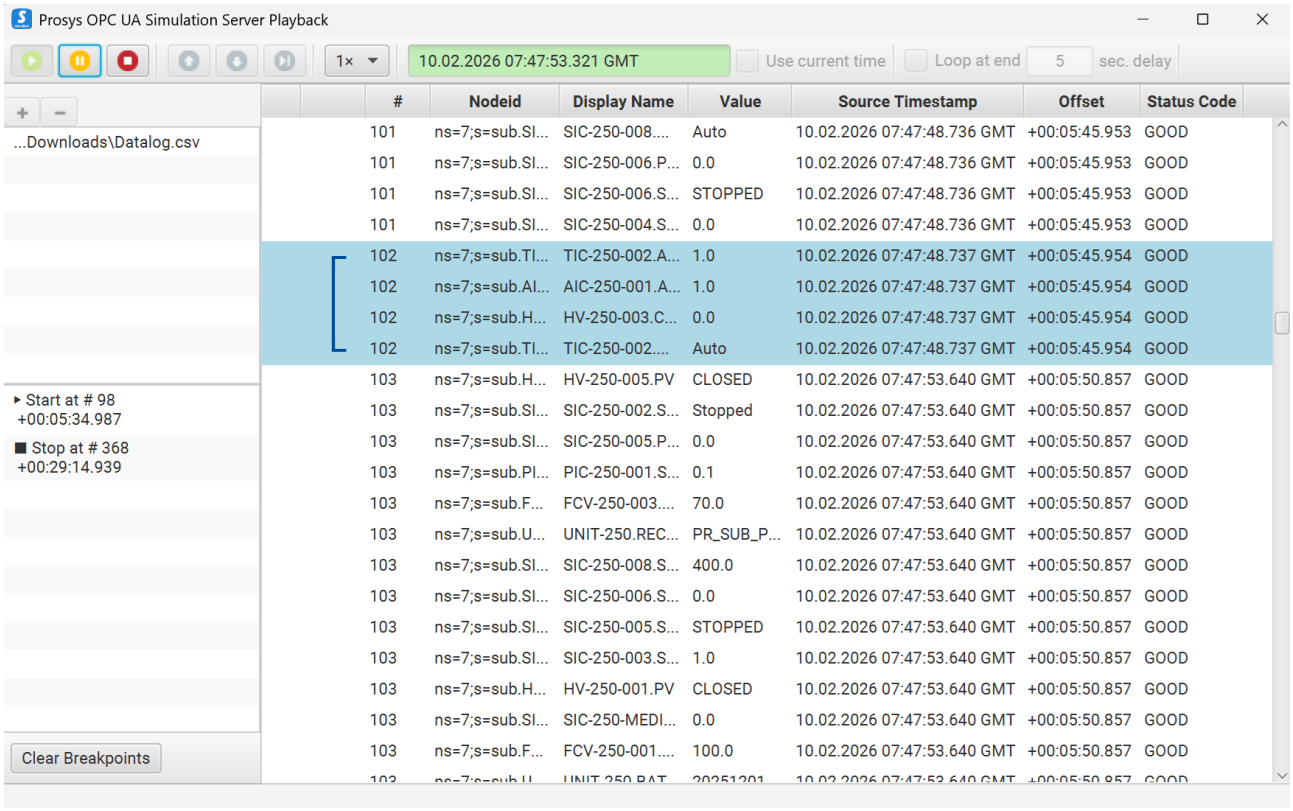

Figure 12. Overview of the Playback window

One or more CSV files can be imported into the Playback view by using the + button. When multiple CSV files are added, their contents are merged into a single chronological playback sequence. The imported files are shown in the file list on the left, and the combined data is displayed in the main table with NodeIds, DisplayNames, Values, SourceTimestamps, Offsets between grouped rows, and StatusCodes.

An imported CSV file is copied into the Project, so the Project contains everything Playback needs and can be moved to another computer. The original CSV file is not needed afterwards, and editing it does not change the imported data. Remove a file from the file list to remove it from the Project as well.

Playback is an alternative to the Value Simulation which is enabled by default in the Objects Tab. When you start Playback, value simulation is disabled. Value Simulation can be re-enabled once playback is stopped.

Because the CSV files are stored in the Project, a very large dataset makes saving the Project slower. The application warns you before importing if the total size grows beyond 50 MB.

Basic use

- Import necessary nodes into the server.
- Import one or more CSV files with the + button.
- Review the loaded data in the table.
- Start playback with the play button.
- Pause or stop playback with the controls in the toolbar.
- Adjust playback speed with the speed selector.

During playback, the table highlights the currently active group of rows so the user can follow the progress of the replayed data. Rows are grouped and applied to the server based on their timestamps, and each such group is a *time block*. Timestamp nanoseconds are truncated before grouping. Time blocks are the unit that the start and end positions, the breakpoints and the block indices used in headless mode all count.

Loading datalogs

Simulation Server Playback currently expects a CSV file with the following format:

- Field Separator: Comma
- Decimal Separator: Period
- Fields to included in the CSV:
  - NodeId (with a Namespace URI, not index)
  - DisplayName
  - Value
  - StatusCode
  - SourceTimestamp (UTC)



Compaptible datalogs and nodesets for Playback can be created using Prosys OPC UA Browser. To learn more on how to achieve this using Browser, please refer to our blogpost on the subject.

Navigation and positioning

The Playback view provides several ways to control where playback starts and stops.

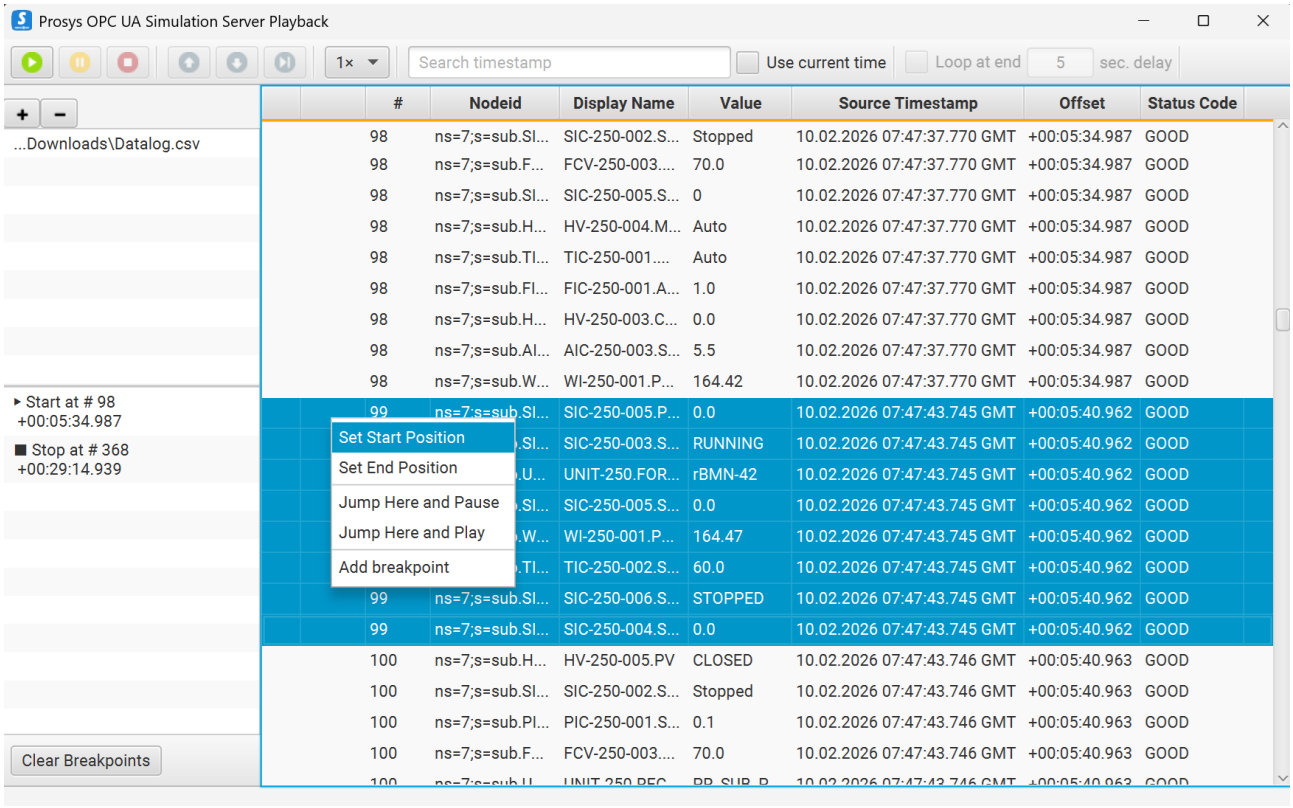

Figure 13. Right-click on a row-group in the table to open the context menu.

- Use the context menu in the table to set a start position.
- Use the context menu in the table to set an end position.
- Add breakpoints to automatically pause playback at selected positions.
- Use the breakpoint list on the left for quick navigation to marked positions.
- When playback is paused, step backward or forward through the data.
- Jump directly to a selected position and continue playback from there.

This makes it possible to focus only on a specific portion of a recording instead of replaying the entire dataset.

Time handling

Playback can be used in two timing modes:

- **Original source timestamps** Data is replayed using the timestamps stored in the recording.
- **Use current time** Data is replayed as if it were happening now, instead of in the past. This is useful for demonstrations and test situations where the data should appear freshly applied.

When **Use current time** is enabled, the displayed timestamps are aligned to the current time at playback start.

Looping

Looping is available when **Use current time** is enabled.

- Enable **Loop at end** to restart playback automatically after the selected range finishes.

- Set the delay value to control how long the application waits before resuming the playback after reaching the end of the log.
- On each loop, the playback starts again from the beginning of the selected range and updates the timestamps to the current time.

This is useful when a repeating live-like data stream is needed for long-running demonstrations or test setups.

Overview

The Playback view also includes tools to make large datasets easier to work with.

- The timestamp field can be used to search for a specific time value in the loaded data.
- The file list shows all currently loaded CSV files.
- The breakpoint panel shows the configured start position, end position, and any breakpoints.

Together, these features allow the user to load recorded data, inspect it, limit the playback range, and replay it in a controlled way on the Simulation Server.

Once the CSV files and the playback settings are saved in a Project, Playback can also be run without the UI: from the command line, or from an OPC UA client application. See Headless and Container Mode.

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. By right-clicking an ObjectType, VariableType or DataTypes it is possible to quickly create a new Variable or Object utilising said type.

From Figure 14 we can see that the view is divided into two parts. The left-hand side contains five tabs: *DataTypes*, *EventTypes*, *ReferenceTypes*, *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 for VariableTypes or InstanceDeclarations.



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.

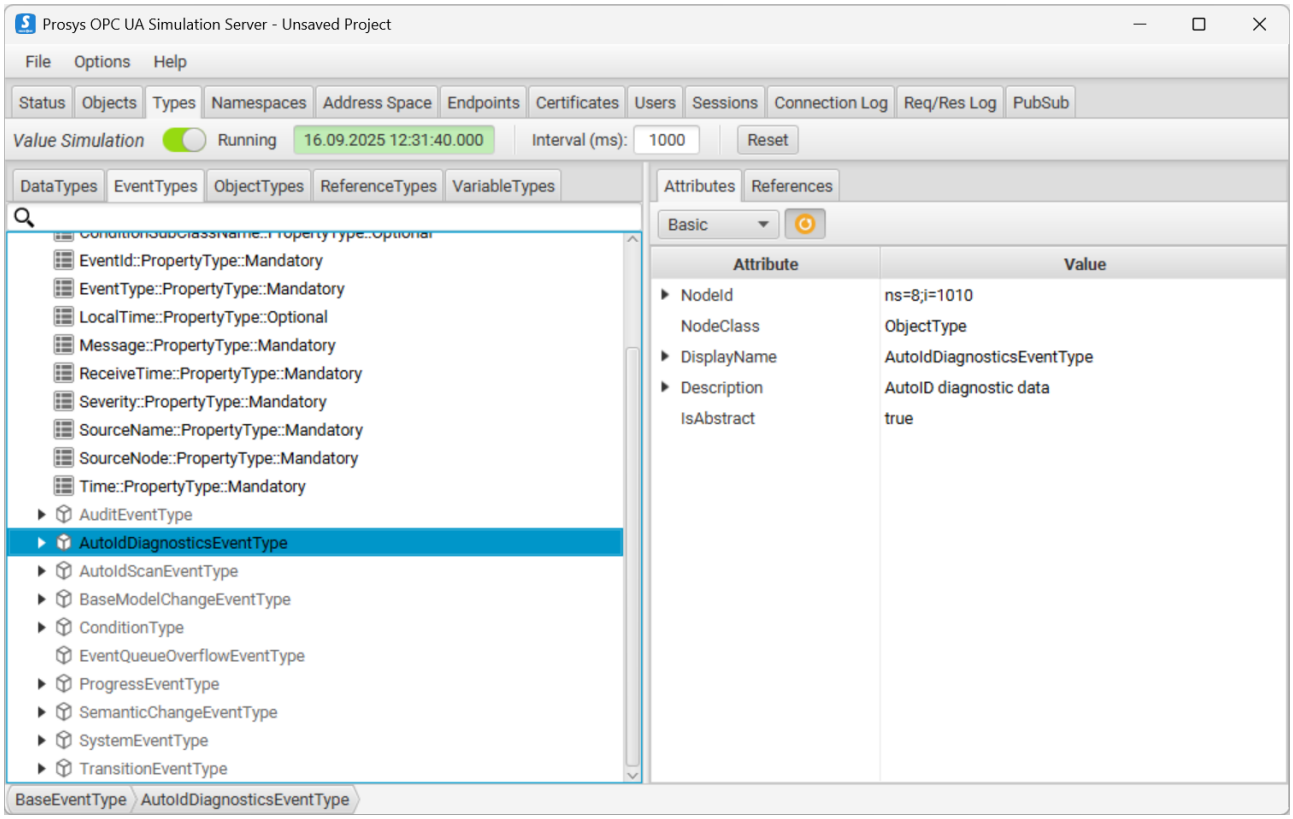

Figure 14. Types view with added AutoIdDiagnosticsEventType highlighted.

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.

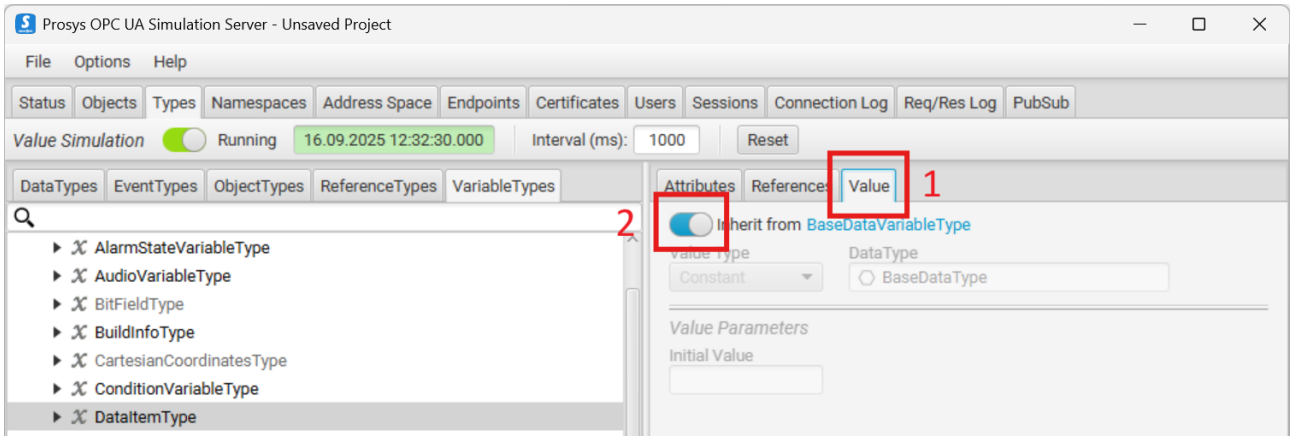

Figure 15. To define a value simulation, select Value tab (1) and disable inheritance of values (2).

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.

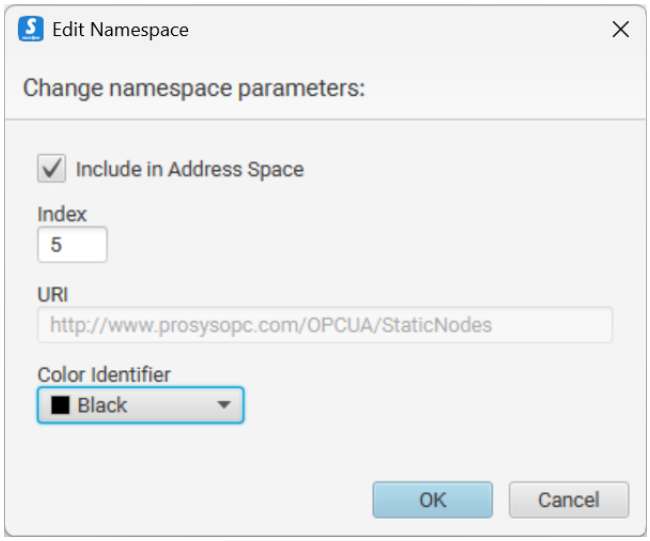

Figure 16. Dialog for editing a namespace. Editable details are enabled.

The information and options available for each namespace are listed below. Some of the details are editable (see Figure 16) and rest are purely informational and can be seen in the table shown in Figure 17.

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.

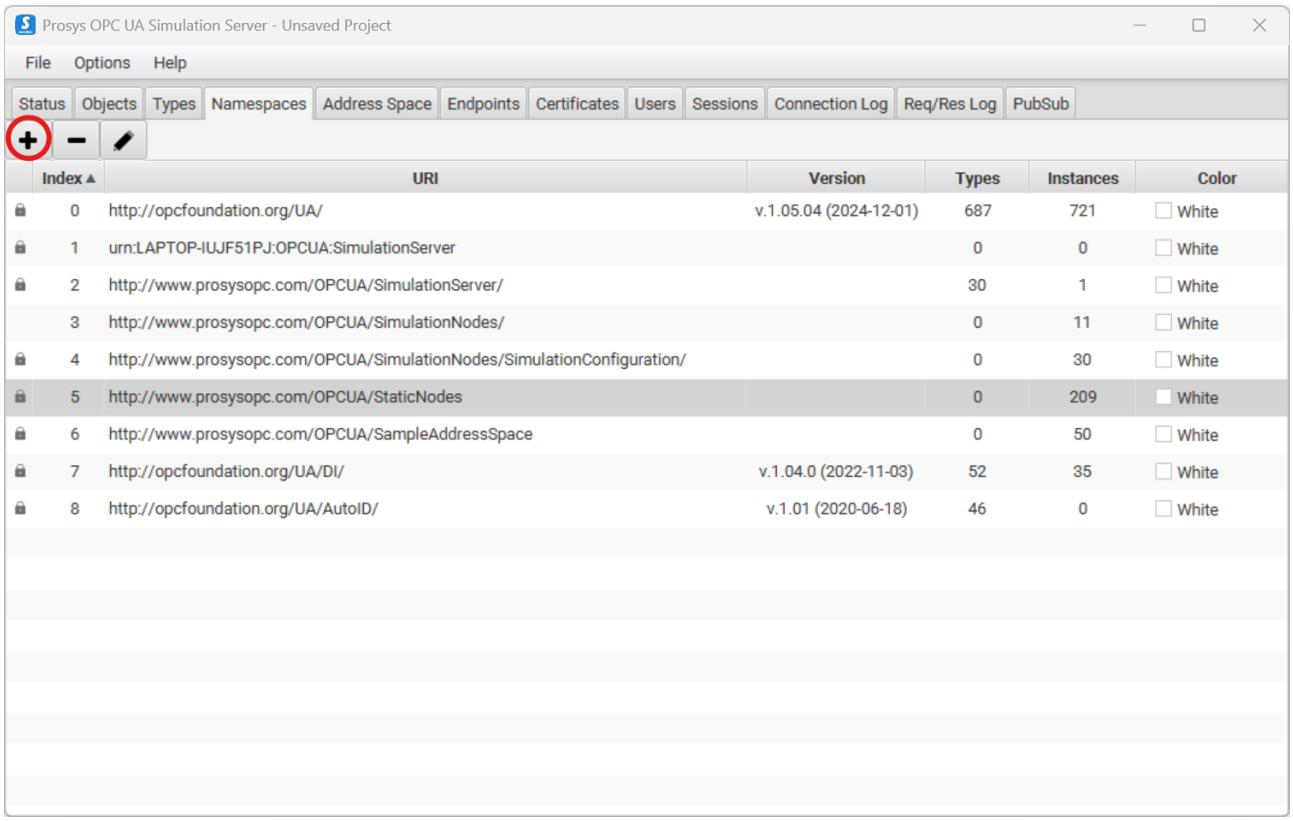

Figure 17. Namespaces View shows the information on the namespaces present in the server.

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 namepace (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 17) and selecting the option to Import NodeSet (see Figure 18). Next, you will need to find the NodeSet file from your computer.

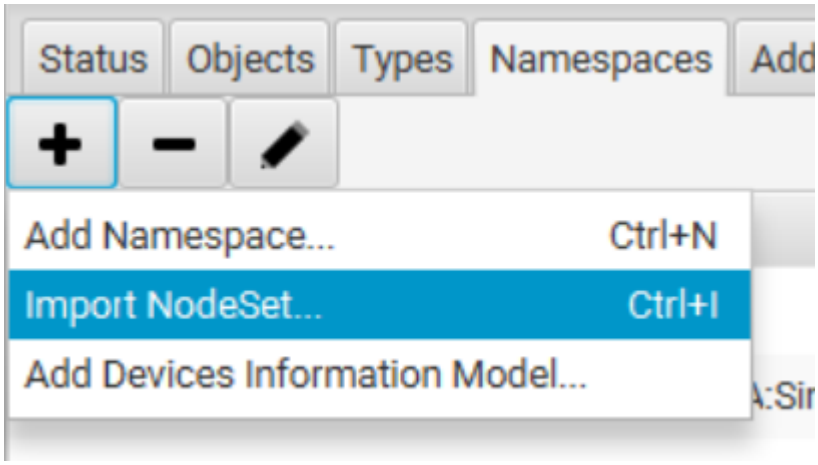

Figure 18. Click on the + and then select Import NodeSet.

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.

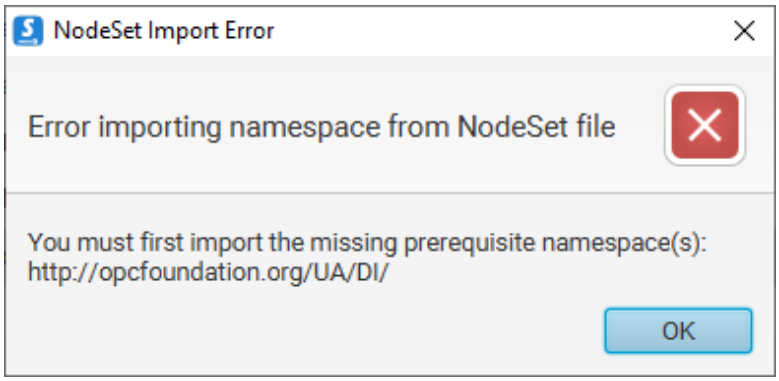

Figure 19. An error message listing all required NodeSets.

Figure 19 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 14. 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 20) 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 Prosys 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.

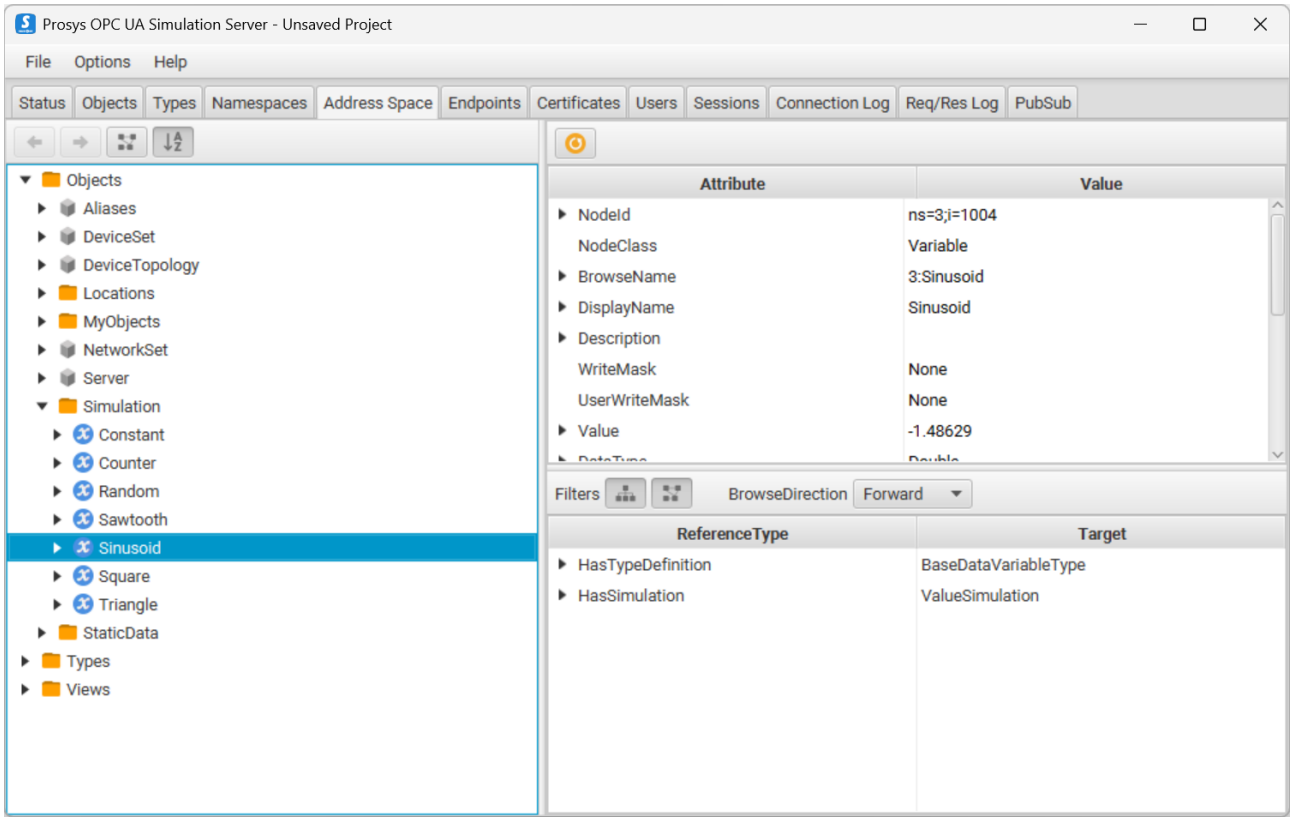

Figure 20. Address Space of the server. The Attributes and References of the selected Node are displayed on the right.

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 21) allows you to configure these settings and verify which endpoints the server is exposing.

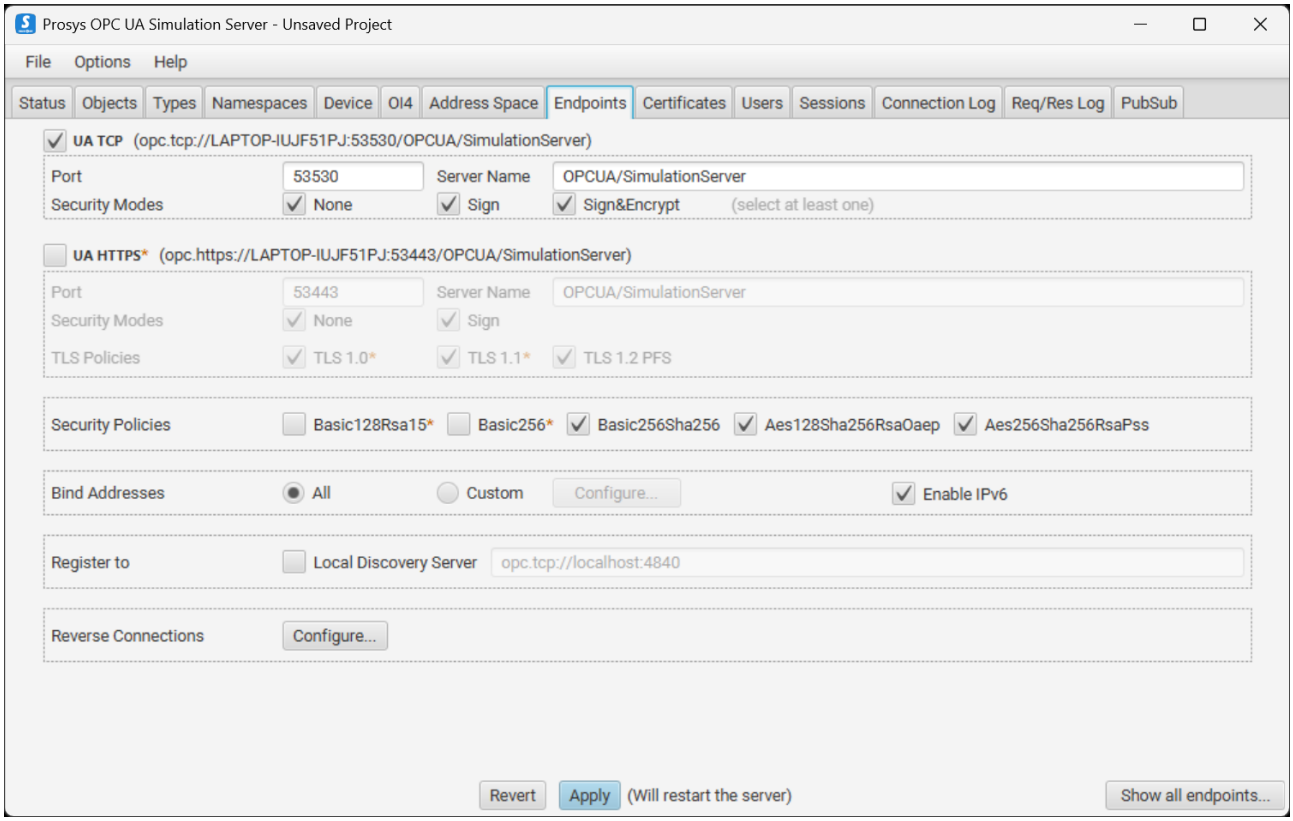

Figure 21. The Endpoints View allows you to configure the connection addresses and security options available for OPC UA clients to connect to the OPC UA server.

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/OPCUA/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.



Support for the UA HTTPS (*opc.https*) transport protocol was removed in version 2026.2.0. UA TCP is the only transport protocol.

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 21) 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 apply the settings. This will restart the server. If you don't apply the settings, you can use Revert to restore the previously stored settings. Saving the Project won't include unapplied settings.



The Port, ServerName, Security Modes and Security Policies are stored in the Project. The *Bind Addresses*, IPv6, Local Discovery Server and Reverse Connection settings are stored per computer instead, because they usually do not apply on another machine. Opening a Project that was created on another computer therefore does not change them. See Projects.

If none of the configured bind addresses exists on the computer, the server logs a warning and binds to all network interfaces instead of binding to none.

Certificates View

The Certificates View is accessible in the Expert Mode.

The *Certificates View* (Figure 22) 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.

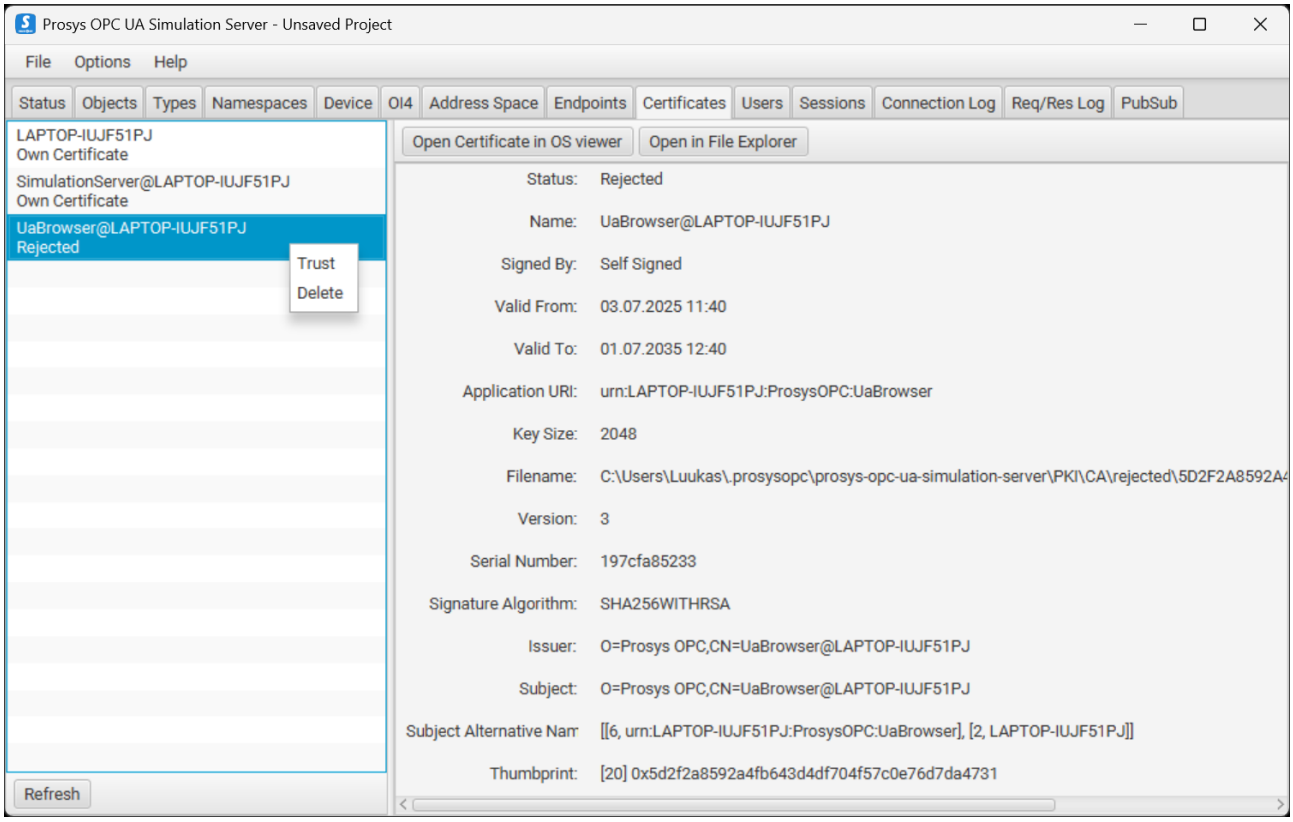

Figure 22. The Certificates View allows you to define which certificates you trust.

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 23) lets you configure the *User Authentication Methods* as well as the user accounts that can be used to access the OPC UA Server.

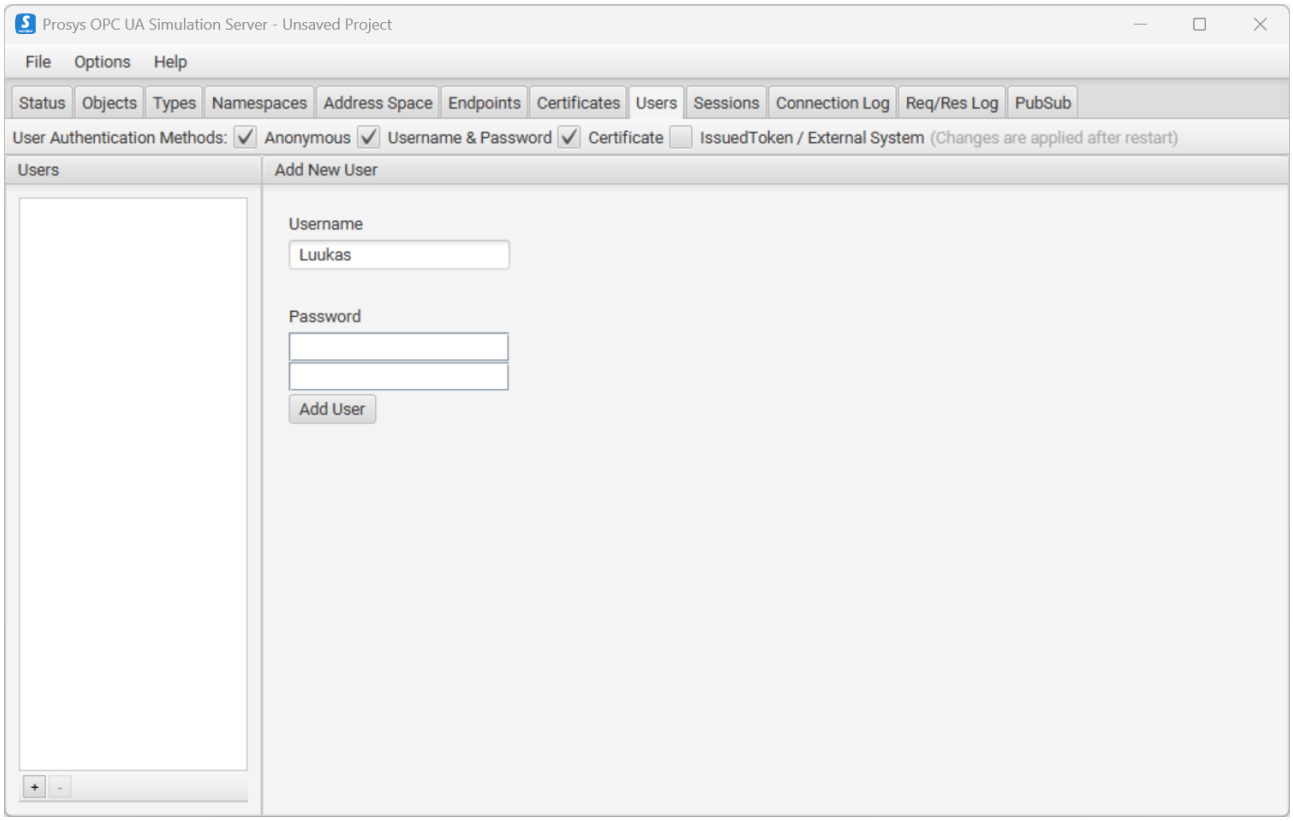

Figure 23. You can use the Users View to add or remove users that may access the OPC UA Server.

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.



Prosys 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 24 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.

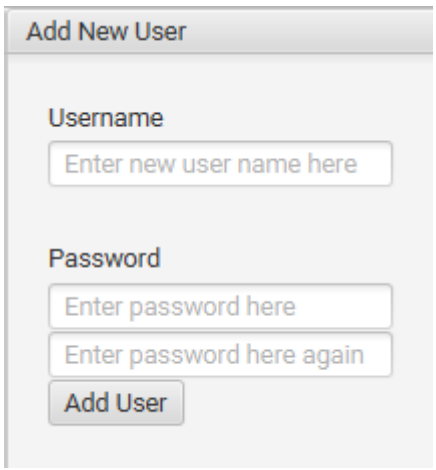

Figure 24. Add New User

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 25) that appears next to the list. You do not need to know the previous password to set a new one.

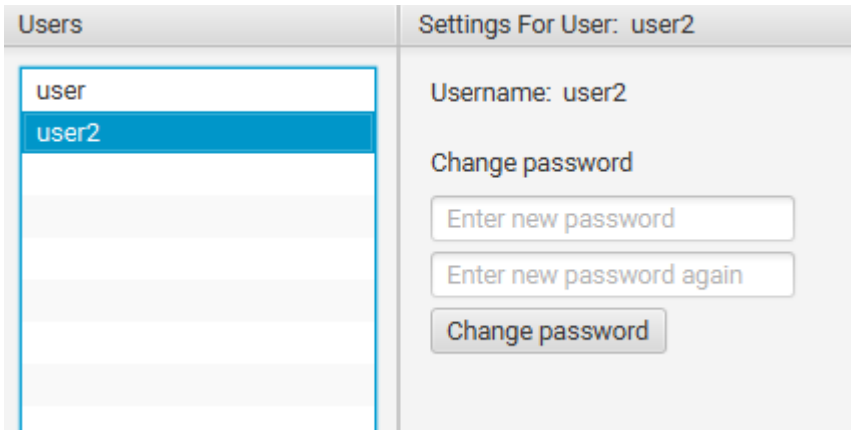

Figure 25. Change user's password

If you wan 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 26) contains information about the currently open OPC UA Client Sessions.

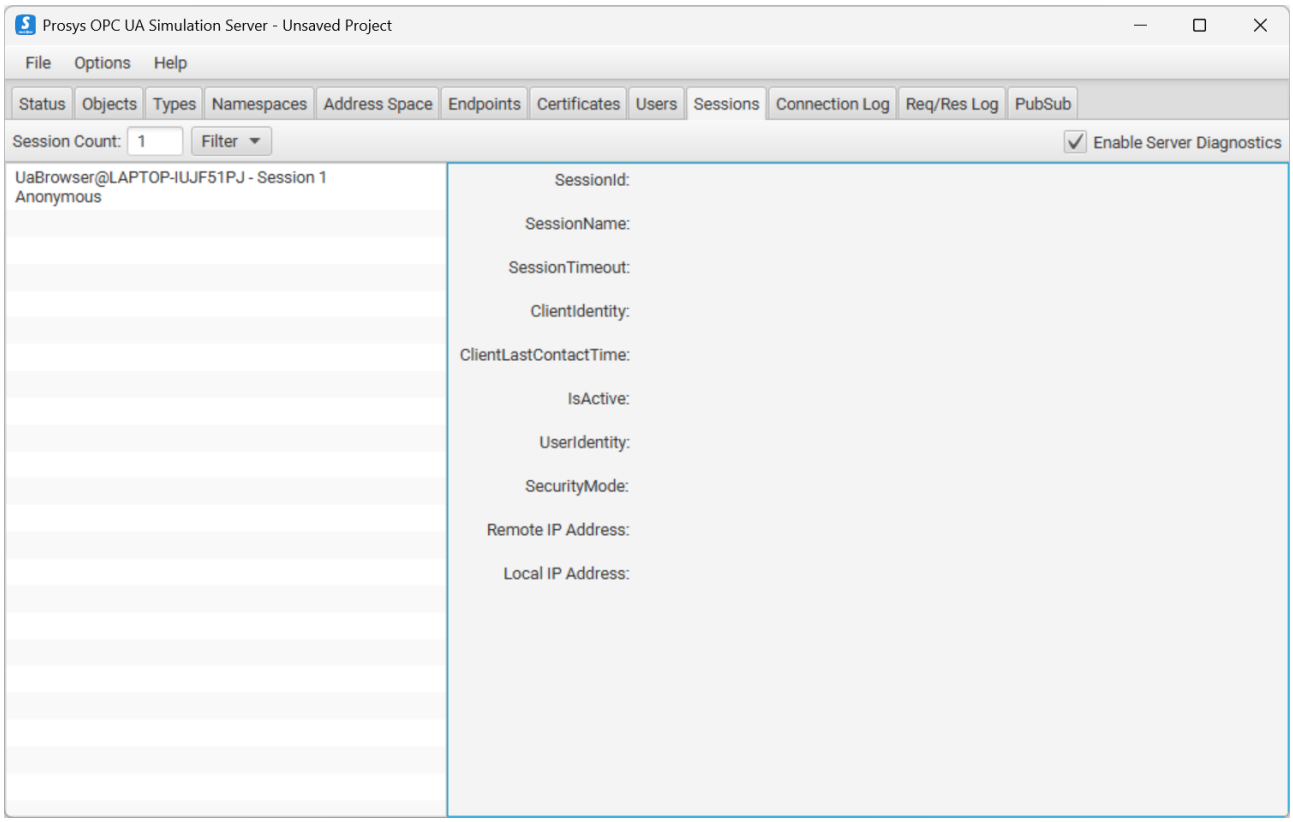

Figure 26. The Sessions View displays all current OPC UA Sessions in the OPC UA Server.

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 27) 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.

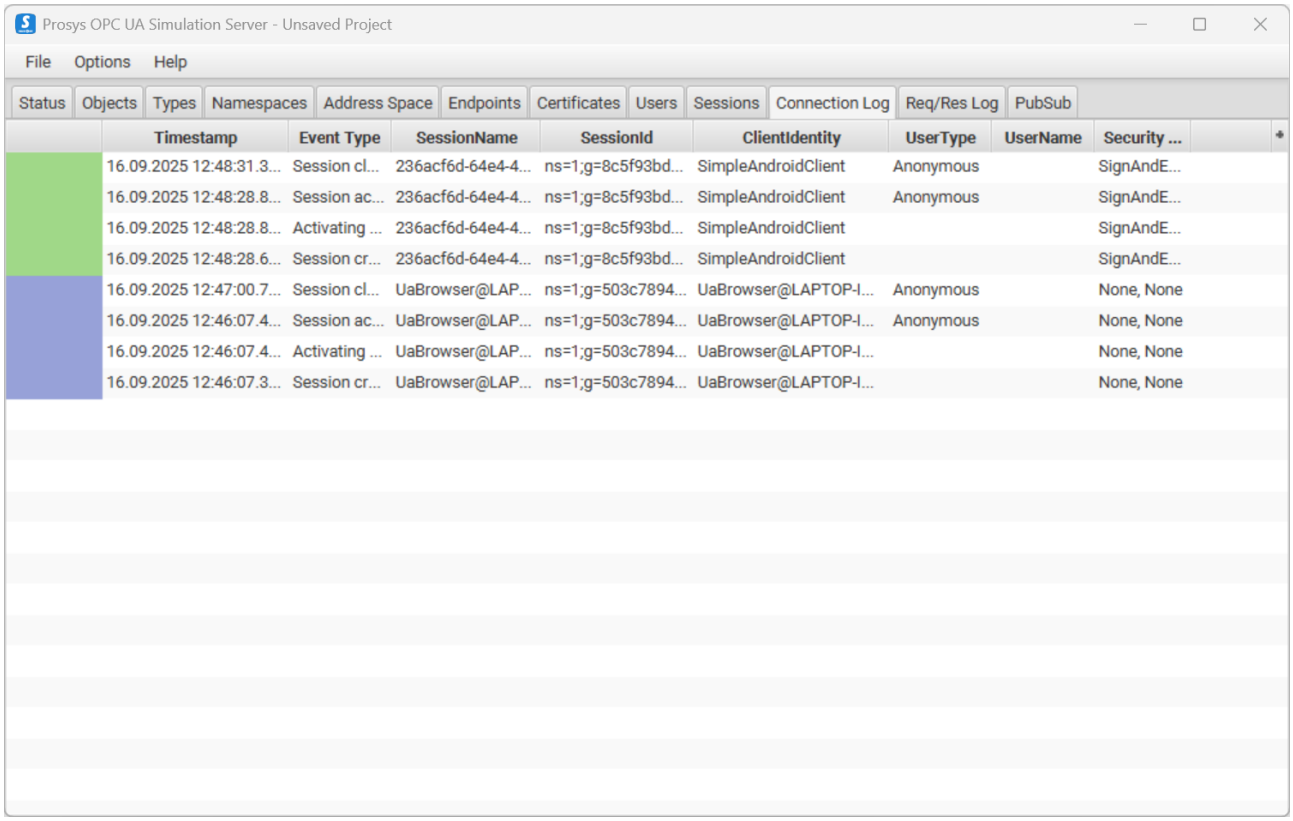

Figure 27. Connection Log View shows all established client connections to the server.

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.

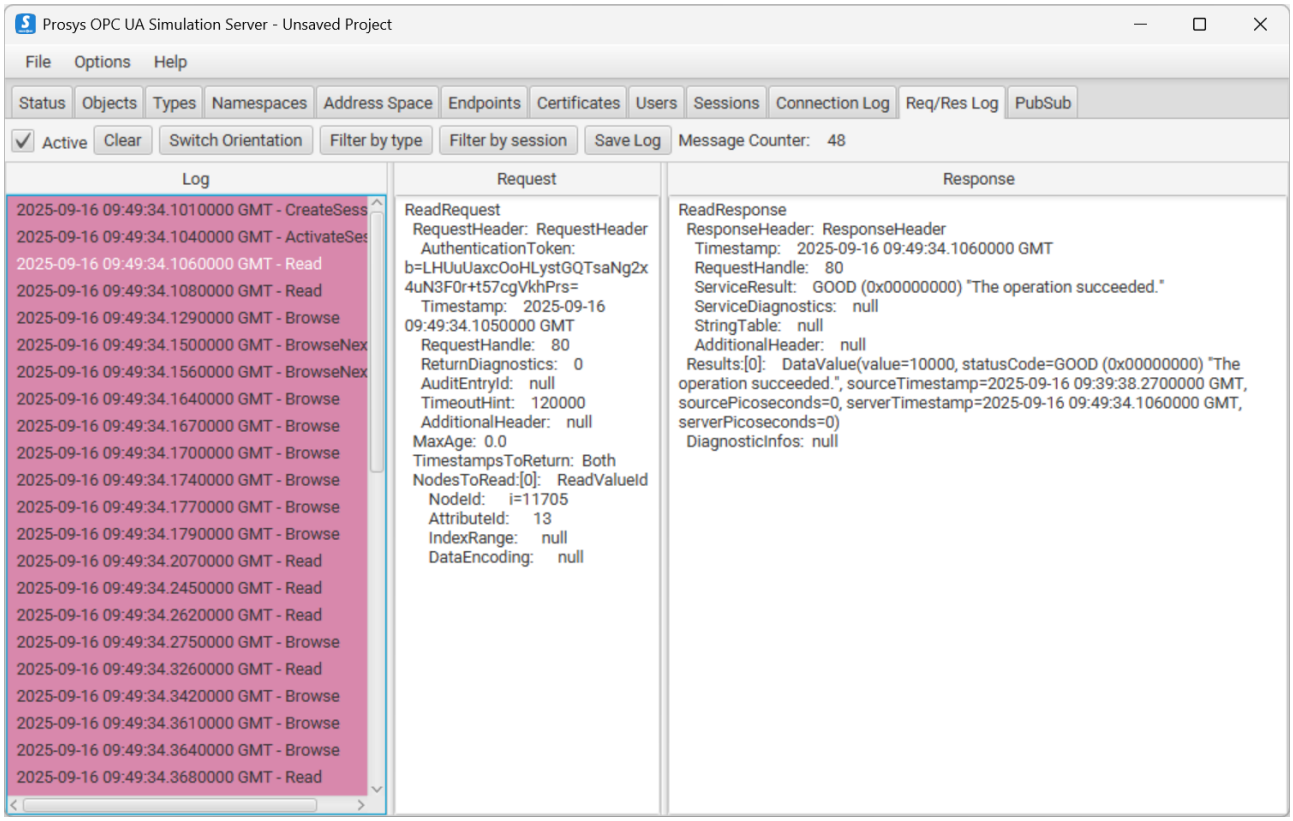

Figure 28. Request-Response Log View.

The view is divided into three parts under the tooldbar. 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.

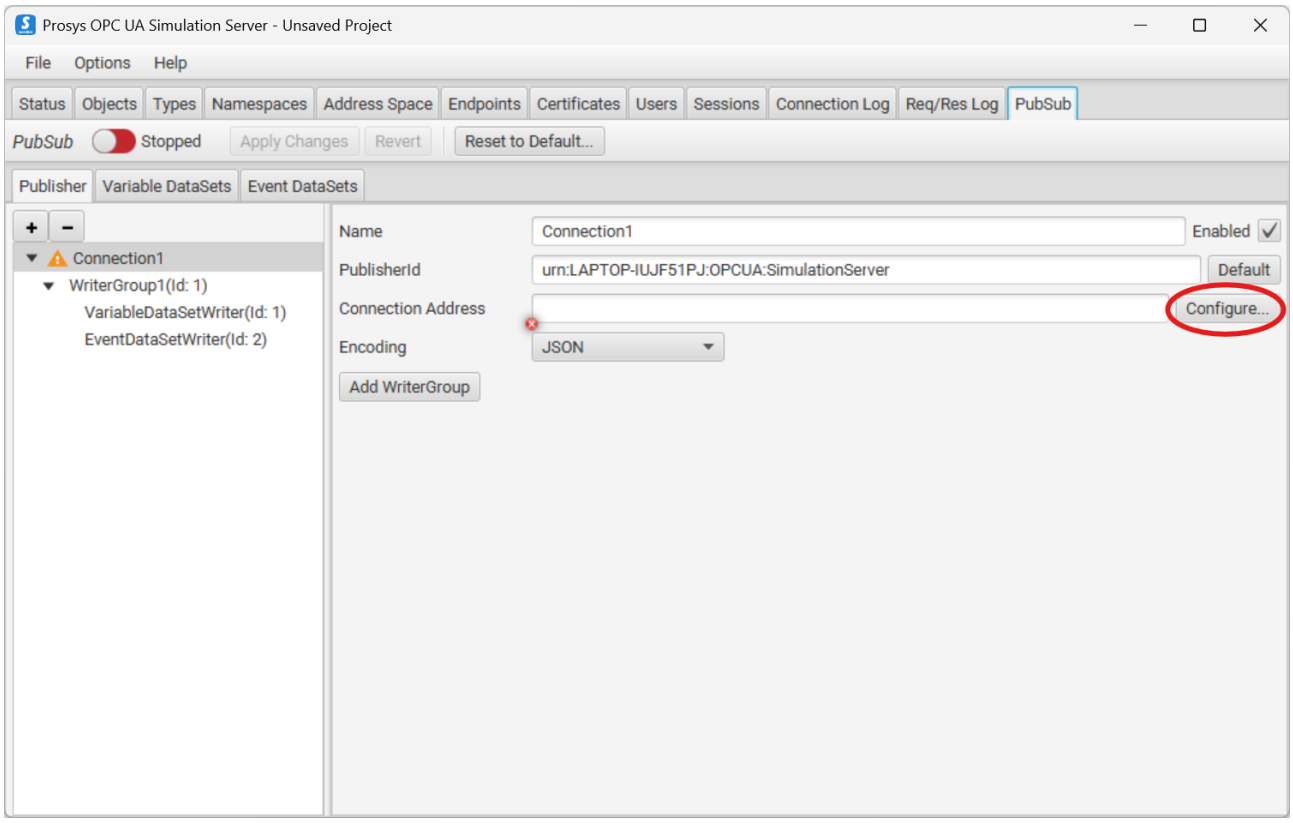

Figure 29. PubSub View on startup.

As can be seen from Figure 29, 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 30). 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.

mqtts

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

```
mqtts://<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.

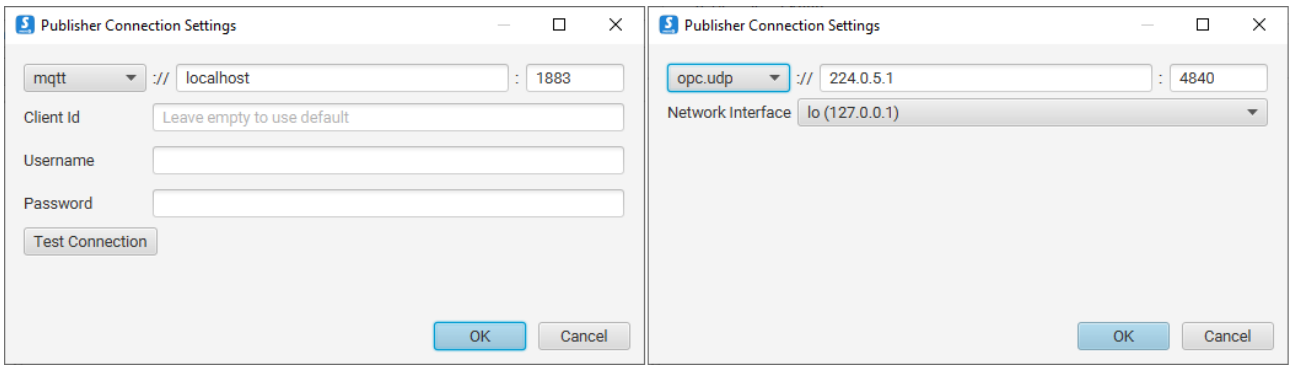

Figure 30. The dialog for configuring the connection settings for the Publisher.

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.

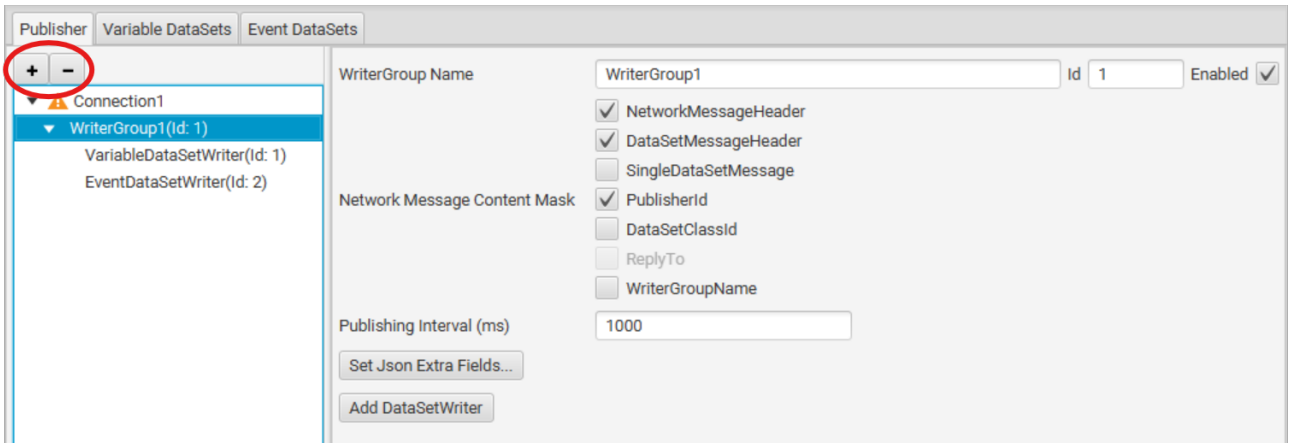

Figure 31. You can add, configure and remove WriterGroups that write the DataSetMessages that will be sent to the PubSub Network.

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 32, 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.

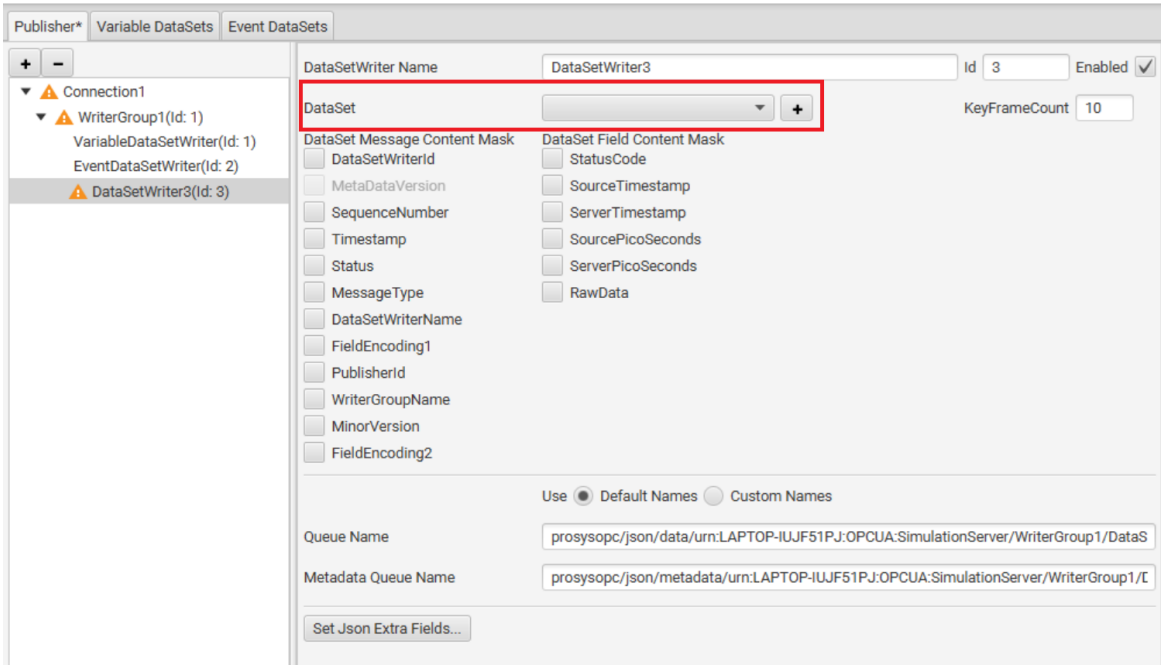

Figure 32. Freshly added DataSetWriter.

You can select the DataSet for your *DataSetWriter* from the drop-down selector highlighted in Figure 33. Simulation Server offers you two options by default: SampleEventDataSet and SampleVariableDataSet.

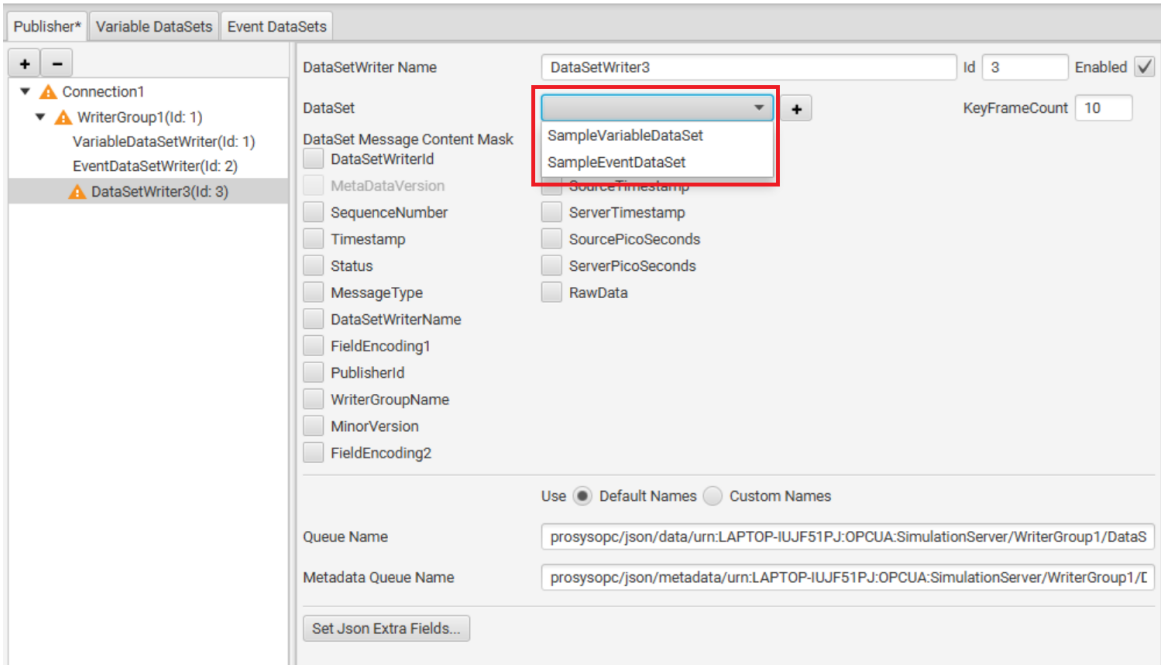

Figure 33. Two DataSet options.

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.

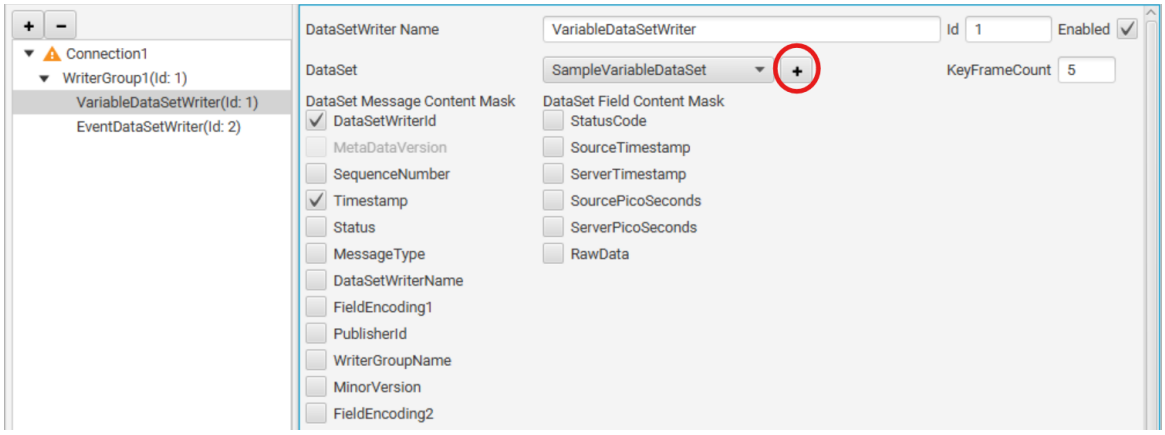

Figure 34. You can add a custom DataSet by clicking the + button.

Variable DataSets View

In the *Variable DataSets View* (Figure 35) 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.

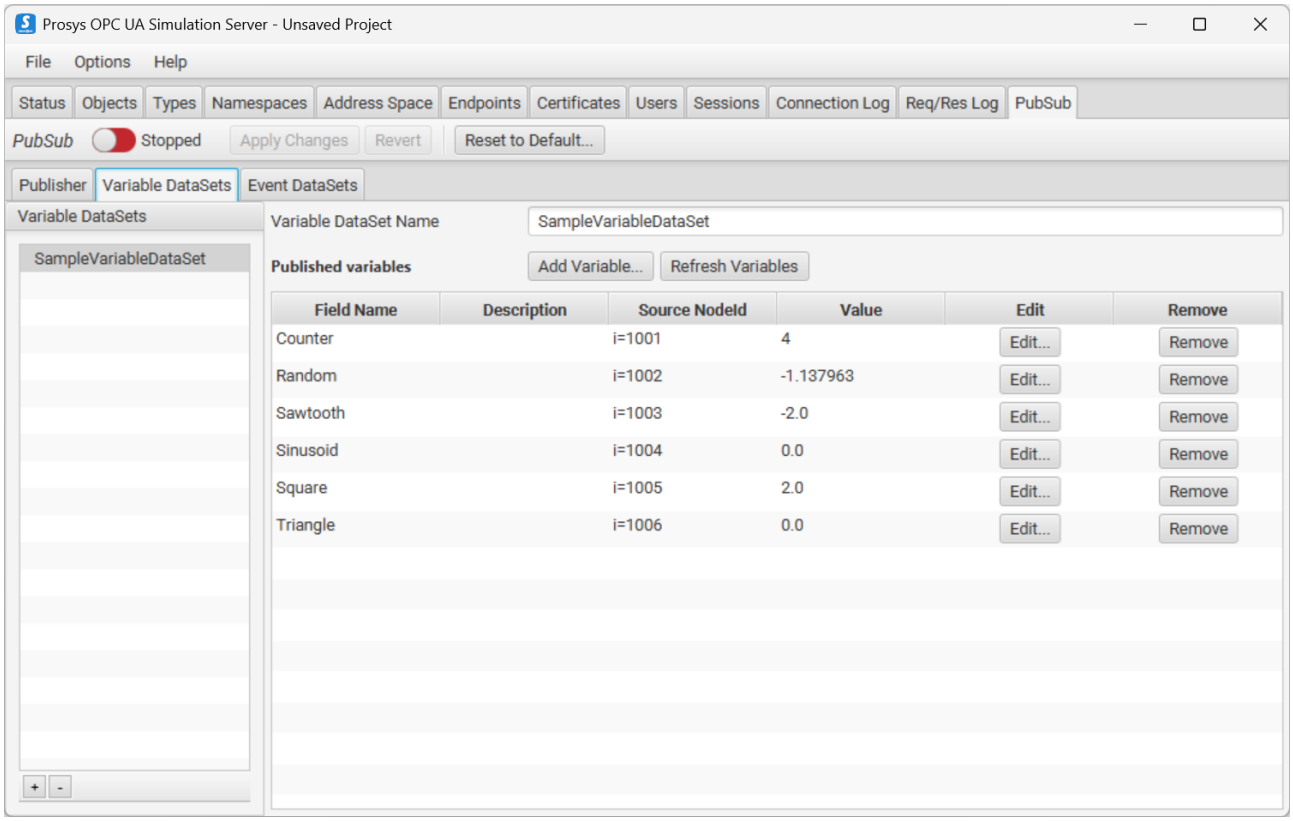

Figure 35. Variable DataSets View.

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.

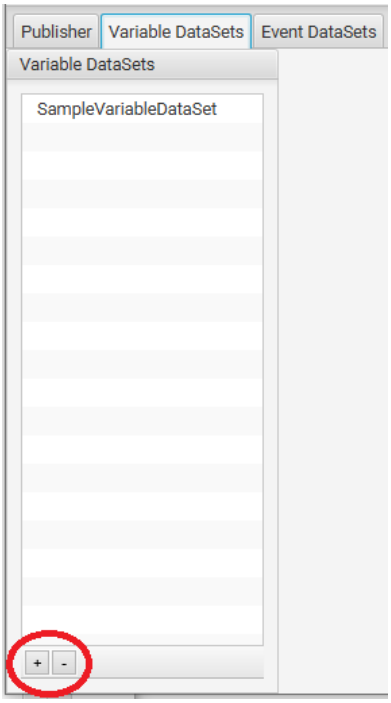

Figure 36. Add or remove a Variable DataSet.

Adding Fields to a Variable DataSet

By clicking the Add button, you can open a dialog (see Figure 37) 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.

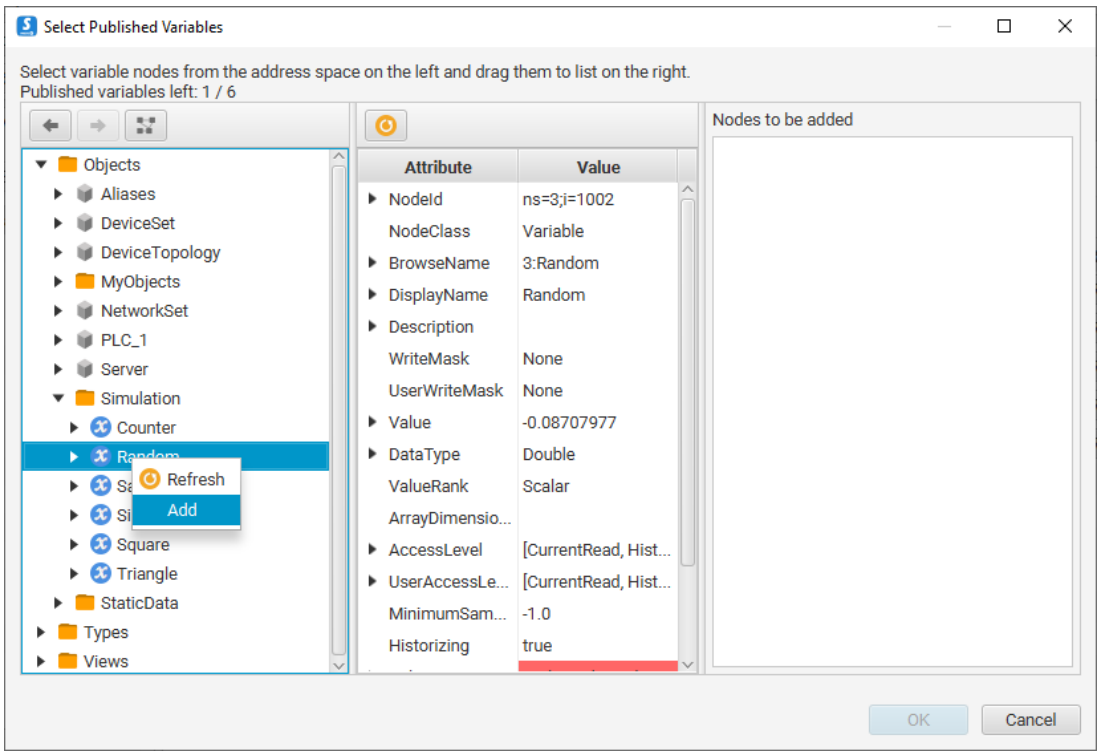

Figure 37. The dialog for selecting Variables to publish.

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 38 you can select which Node will be monitored for EventNotifications, and which Fields will be included in the DataSetMessage.

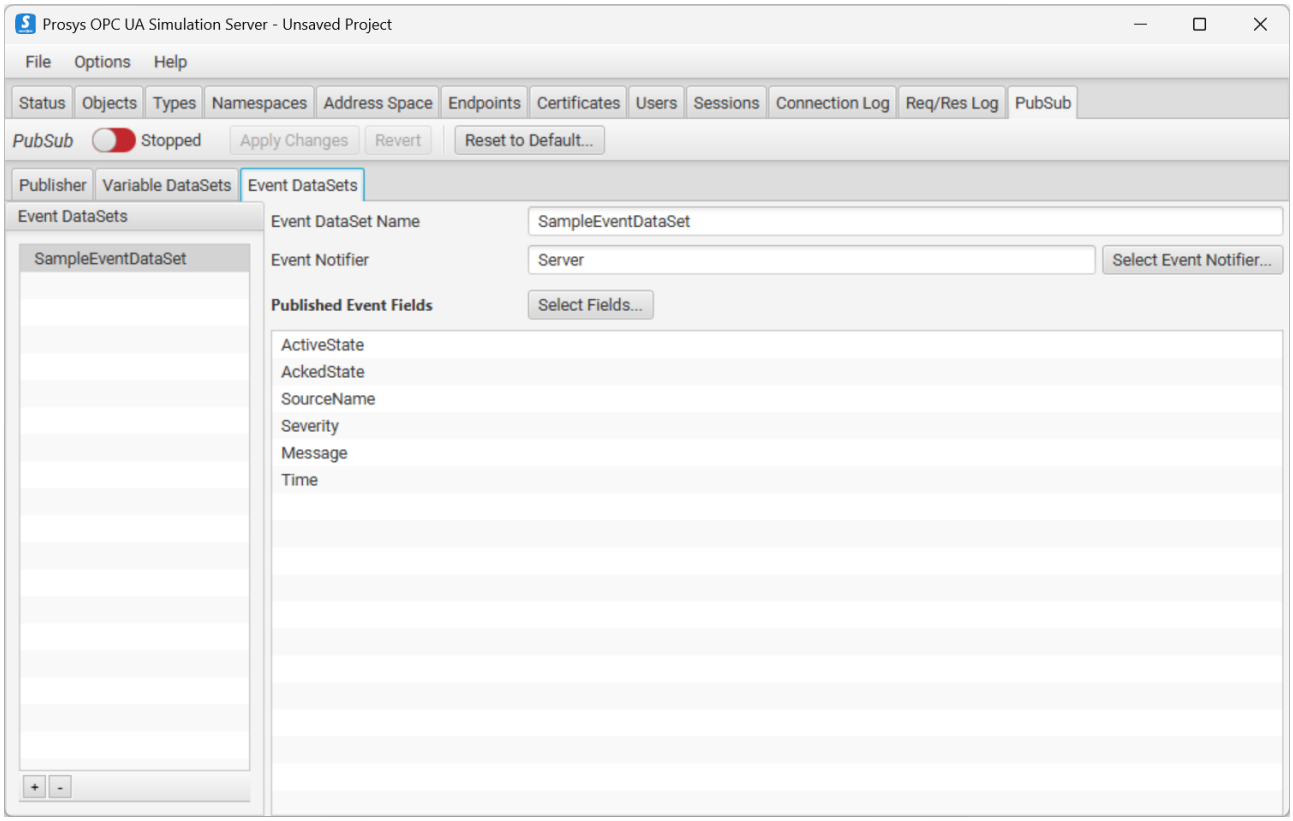

Figure 38. Event DataSets View.

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.

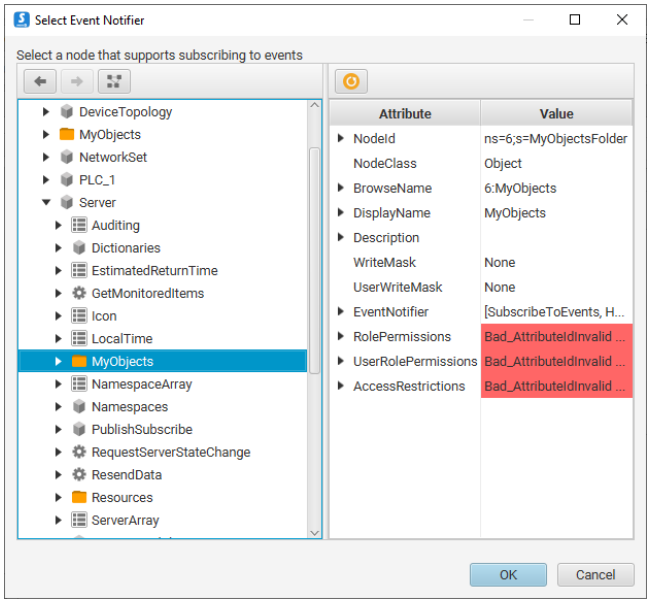

Figure 39. Event Notifier selection dialog.

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 40. 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.

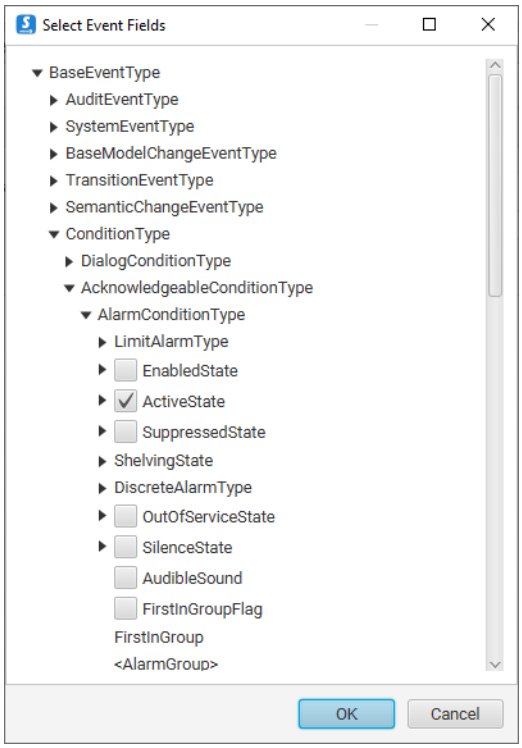

Figure 40. Event Fields selection dialog.

Adding and removing Event DataSets (Professional Edition only)

You can add and remove Event DataSets with the buttons highlighted in Figure 41. 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.

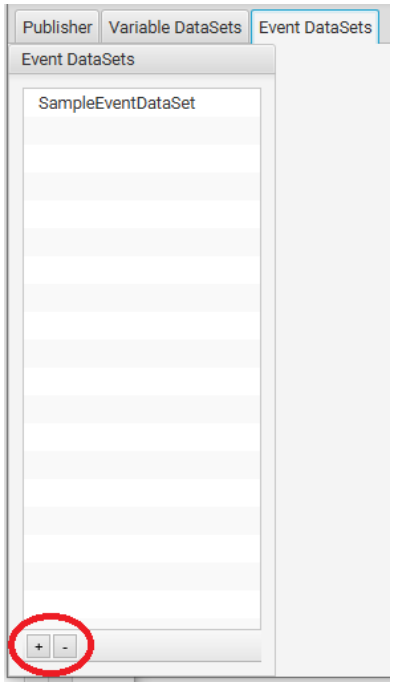

Figure 41. Add and remove Event DataSets.

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 42, 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/).

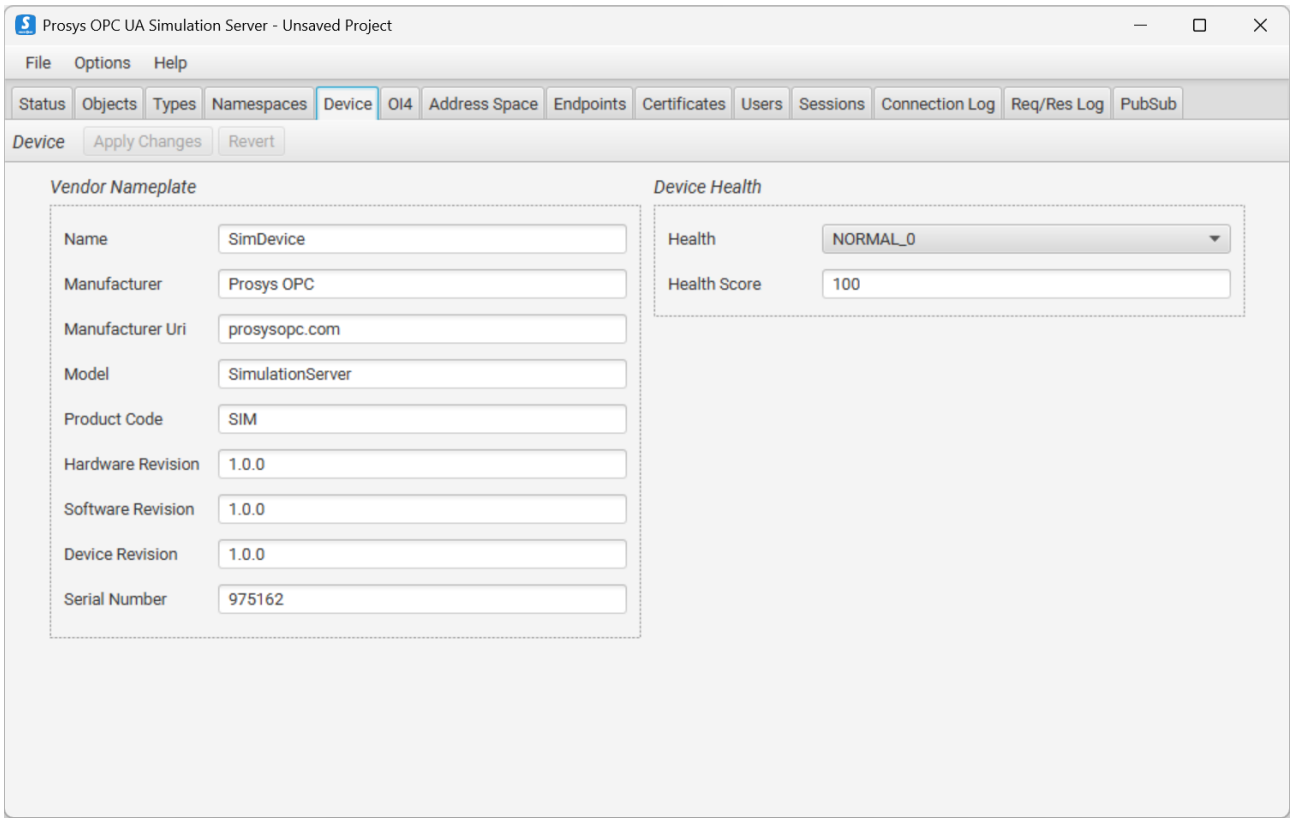

Figure 42. Device View for simulating OEC.

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 43, 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.

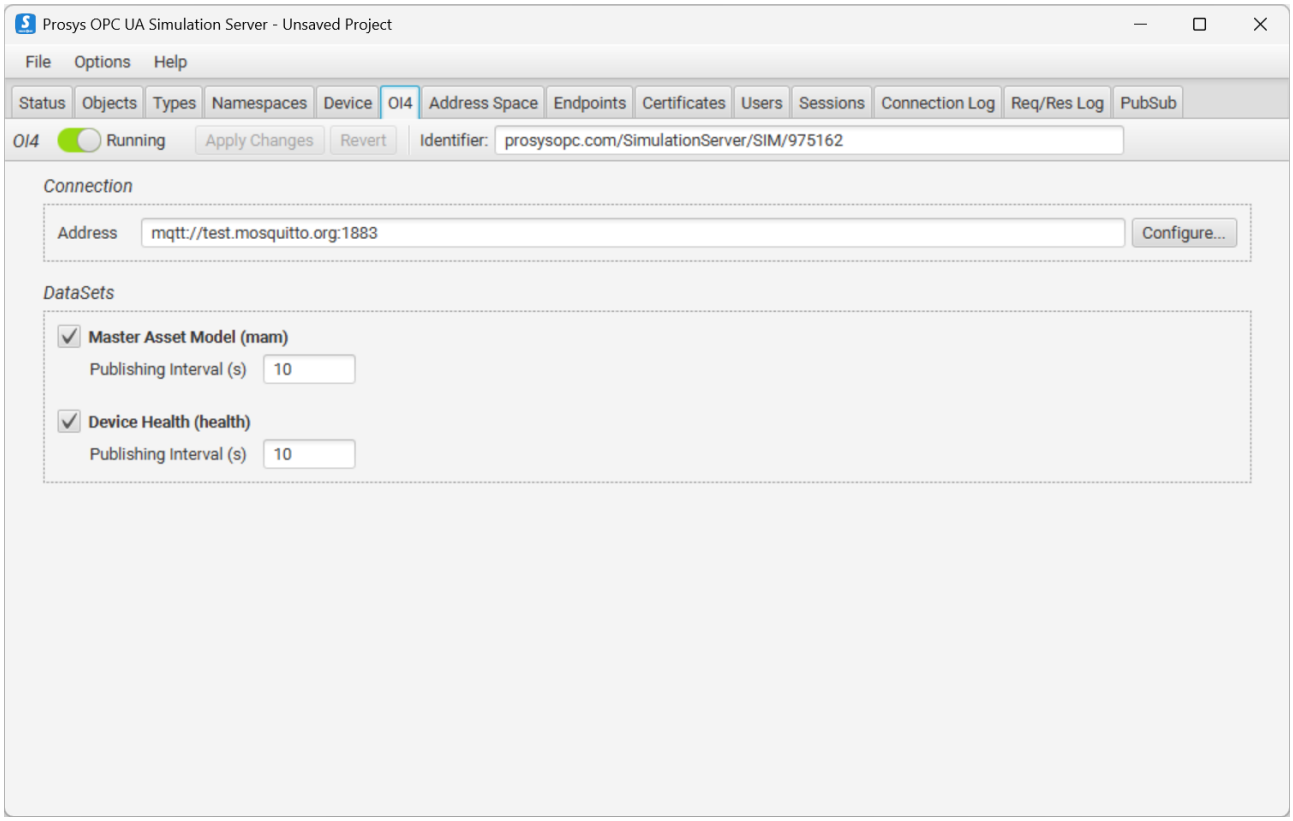

Figure 43. Device View for simulating OEC.

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

Headless and Container Mode (Professional Edition only)

The Simulation Server can also be run *headless*, meaning without the configuration UI. A headless server runs in the foreground and is configured with command-line options and with a Project that you have prepared with the desktop application. It is meant for automated use, such as CI/CD pipelines and test environments.

Headless mode is the installed desktop application started with the --headless option, so there is nothing separate to install. The same server is also available as a ready-made container image, described in Docker Container distribution.

Headless mode requires a Professional or Evaluation license.

Starting the server

Start the server with --headless and optional arguments. The log is written both to the console and to the file described in File Locations.



The installed launcher runs one instance per user, in headless mode as well. If an instance is already running, a second launch exits with code 0 without printing anything, and the running instance writes a warning to its log with the ignored arguments. To run several servers on one host, use the container image.

Windows

Add the -console option to the installed launcher UaSimulationServer.exe to get the output.

```
"C:\Program Files\ProsysOPC\Prosys OPC UA Simulation Server\UaSimulationServer.exe" -console --headless
--project C:\path\project.uasim
```

Linux

The launcher in the installation folder works like a normal command and needs no special handling:

```
/opt/prosys-opc-ua-simulation-server/UaSimulationServer --headless --project [/path/to/project.uasim]
```

macOS

Start the server directly with java, using the libraries inside the bundle:

```
java -cp "/Applications/Prosys OPC UA Simulation Server.app/Contents/java/app/lib/*" \
  com.prosysopc.ua.app.Main --headless --project [/path/to/project.uasim]
```

Command-line options

Option	Description
--headless	Run without the configuration UI.
--project <file>	The Project (.uasim) file to load. A plain path to a .uasim file works as well. --config is a deprecated alias.
--license <file>	The license file to read. May be read-only and outside the data folder. Default is license.lic in the data folder.
--data-dir <dir>	The folder for settings, logs and the working copy of the Project. Default is the folder described in File Locations.
--pki-dir <dir>	The folder holding the PKI and USERS_PKI certificate stores. Default is the data folder. Reuse the same folder to keep the certificates between runs.
--log-config <file>	A log4j2 configuration file to use instead of the generated one. Its appenders then decide where the logging goes.
--save-project	Save the changes made at runtime back into the Project file. See Saving the Project.
--port <port>	Serve on this OPC UA TCP port instead of the one stored in the Project. Applies to this run only and is never written back.
-T, --trust-all-certificates	Trust every client certificate, but keep validating them. See Client certificates in headless mode.
-A, --accept-all -certificates	Accept every client certificate without validating it. See Client certificates in headless mode.
--no-server-control	Do not expose the ServerControl Object. See Controlling the server over OPC UA.
--help	Print the available options and exit.

The --port, --trust-all-certificates, --accept-all-certificates, --no-server-control, --save-project and --playback -exit-when-finished options are only available in headless mode.

Exit codes

Code	Meaning
0	The server stopped normally.
1	The command line was invalid, or the server could not be started. Exception: when another instance is already running for the same user, the installed launcher exits with 0 (see the note in Starting the server).
6	Playback failed, or stopped before reaching its end position.
7	The data folder is missing or not writable.

Code	Meaning
8	The Project is not writable. Only checked with --save-project.
9	The Project file is damaged, or a path given with --pki-dir or --log-config cannot be used.
10	The license file is missing or unreadable.
11	The license cannot be used, so the headless start is refused: it is expired, issued for another host or another product, or the file is damaged.

Saving the Project

By default a headless run never writes the Project: it is read again on every start, the changes made at runtime are discarded when the server stops, and the Project file may be read-only.

With --save-project the changes made at runtime are written back into the Project file, so they survive a restart. The Project's *folder* must therefore be writable, not just the file, or the server stops at startup.

Certificates are separate from Project saving. Reuse --pki-dir to keep the server certificate and trusted client certificates between runs.

Running Playback from the command line

With --playback, the server starts Playback of the Project's CSV data as soon as it is ready. The Project must contain the CSV files and the Playback settings, which are prepared with the desktop application (see Playback).

The --playback-* options below require --playback. They override the Project's Playback settings for this startup only; a later StartPlayback over OPC UA uses the stored settings again.

Option	Description
--playback	Start Playback on startup, using the Playback settings of the Project.
--playback-speed <n>	Override the speed multiplier, as a whole number.
--playback-loop	Override looping to enabled.
--playback-loop-delay <seconds>	Override the delay between loops.
--playback-original-time	Use the original timestamps of the CSV data.
--playback-start <index>	Override the start position, as a time block index.
--playback-end <index>	Override the end position, as a time block index.
--playback-exit-when-finished	Replay the data once and then stop the server. Headless only. Cannot be combined with --playback-loop.

The block indices are counted from zero, while the Playback window numbers the blocks from one: --playback-start 0 is the block shown as # 1.

Breakpoints are ignored when Playback is started from the command line.

For a test job that should replay the data once and then finish, combine --playback with --playback-exit-when-finished. The server stays in the foreground until Playback reaches its end position, then shuts down normally.



With --playback the first values may be applied before your client has subscribed to them. If your test needs every value, start the server without --playback and call the StartPlayback Method once the client is connected and subscribed.

Controlling the server over OPC UA

A headless server is controlled over OPC UA. A ServerControl Object is available in the address space under the standard *Objects* folder. Any OPC UA client can use it to start and stop Playback and the value simulation, and to follow their state.

The state of the server is exposed as read-only Variables:

BrowseName	Type	Description
PlaybackActive	Boolean	Playback is running.
PlaybackPaused	Boolean	The running Playback is paused.
PlaybackFinished	Boolean	Playback reached its end position without looping.
PlaybackCurrentBlockIndex	Int32	The current time block index, or -1 when Playback is not running.
PlaybackBlockCount	Int32	The number of time blocks in the loaded data.
PlaybackCurrentTime	DateTime	The timestamp of the current position.
PlaybackLastError	String	The most recent Playback error, or empty when there is none.
SimulationActive	Boolean	The value simulation is active.

The Methods take no arguments:

Method	Description
StartPlayback	Start Playback using the Playback settings of the Project. Stops the value simulation. Requires a Professional license.
StopPlayback	Stop a running Playback.
StartSimulation	Stop Playback and start the value simulation.

Method	Description
StopSimulation	Stop the value simulation.

StartPlayback returns Bad_InvalidState when the Project holds no Playback data. The most recent Playback error is readable from the PlaybackLastError Variable.

The Methods are available to every client that can open a Session, including anonymous ones, because the server has no per-user permissions. On a server that untrusted clients can reach, leave the Object out of the address space with --no-server-control. It is stored as a Project setting, so --save-project keeps it, and it is shown as *Enable ServerControl Object* in the Preferences Window. The change takes effect on the next server restart.

Client certificates in headless mode

A headless server has no UI for trusting a client certificate, so a client that connects with security must be trusted in advance. There are three ways to handle this:

- Trust the certificate through the PKI/CA/certs folder, keeping certificate validation in effect. Copy the certificate there before starting the server, or, if the client has already connected and been turned away, move its certificate from PKI/CA/rejected into PKI/CA/certs. The server reads the folder again on every connection attempt, so the move takes effect without a restart. See Certificate Stores.
- Start the server with --trust-all-certificates, which trusts every client certificate that is presented. The certificates are still validated, so an expired or otherwise invalid one is rejected.
- Start the server with --accept-all-certificates, which skips client certificate validation entirely, so expired, revoked and malformed certificates are accepted too. It overrides --trust-all-certificates.

Neither option changes user authentication.



Use these options only in a controlled network and never on a server reachable by untrusted clients.

Both options are stored as a Project setting, so --save-project keeps them and the Project carries them to any computer that opens it, where they are shown as *Trust All Client Certificates* and *Accept All Client Certificates* in the Preferences Window.

Docker Container distribution

The headless server is also available as a downloadable package that contains a ready-built Linux container image and two sample launcher scripts, a .sh for Linux and macOS and a .ps1 for Windows. Choose the package that matches your Docker host: docker-amd64 or docker-arm64v8.

Extract the package and run the sample launcher:

```
./bin/simulation-server-docker.sh
```

On Windows, run bin/simulation-server-docker.ps1 with PowerShell instead.

The launcher loads the image and runs the server in the foreground, publishing the OPC UA port 53530. Set SIMSERVER_LICENSE to your license file before running it, or the launcher stops with an error. Without a Project the server starts with the default settings; set SIMSERVER_PROJECT to load one.



Use a fixed SIMSERVER_HOSTNAME when you need a stable server certificate, advertised endpoint or hostname-locked license. Docker otherwise assigns a random hostname, and the server generates a new certificate every time the container is replaced.

The environment variables below configure the launcher. Any arguments given to the launcher are passed on to the server, for example ./bin/simulation-server-docker.sh --playback.

Variable	Description
SIMSERVER_HOSTNAME	The hostname to give the container.
SIMSERVER_PROJECT	Path to a Project file. Its folder is mounted at /project, read-only unless SIMSERVER_SAVE_PROJECT=1.
SIMSERVER_LICENSE	Required. Path to a license file. It is mounted read-only at /license/license.lic, so the file does not have to be writable.
SIMSERVER_SAVE_PROJECT	Set to 1 to save runtime changes back into the Project, as the desktop application does.
SIMSERVER_PKI	The host folder mounted at /pki for the certificate stores. Default is pki in the extracted package folder, the one containing bin.

The launcher is a convenient way to get started, but it is not required. The image can also be run directly:

```
docker load -i docker-image/*-image.tar
docker run -d --name simulation-server --stop-timeout 15 \
  --hostname <licensed-host> \
  -p 53530:53530 \
  -v "$PWD/project.uasim:/project/project.uasim:ro" \
  -v "$PWD/license.lic:/license/license.lic:ro" \
  -v simserver-pki:/pki \
  <image> --headless --project /project/project.uasim \
    --license /license/license.lic --pki-dir /pki
```

Any of the command-line options above can be added at the end, for example --playback to start Playback right away.

Stop it with docker stop -t 15 simulation-server. The timeout lets the server complete its shutdown before Docker kills it, which is what a run with --save-project needs in order to write the Project back.

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 Prosys 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.

Prosys 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:

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 desktop 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.

Each store keeps its certificates under a CA sub-folder:

- PKI/CA/certs for the trusted certificates.
- PKI/CA/rejected for the rejected certificates.

Prosys 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.