AXOOM Gate

Configuration Guide

For

078 – OPC UA Client Plugin

Rev 2018-07-26 (for AXOOM Gate v 4.106.0.00)

July 26, 2018

Table of Contents

[Chapter 1 Introduction to AXOOM Gate Plugins 4](#_Toc522008906)

[1.1 Overview 4](#_Toc522008907)

[1.2 About AXOOM Gate 4](#_Toc522008908)

[1.3 About Plugins 4](#_Toc522008909)

[1.4 Plugin Deployment Patterns 5](#_Toc522008910)

[1.5 The Plugin Dashboard 6](#_Toc522008911)

[Chapter 2 Connecting to an OPC UA Server 7](#_Toc522008912)

[2.1 About the OPC UA Client Plugin 7](#_Toc522008913)

[2.2 Login to AXOOM Gate 7](#_Toc522008914)

[2.3 Accessing the plugin dashboard 9](#_Toc522008915)

[2.4 Adding connections to OPC UA servers 9](#_Toc522008916)

[2.5 Accessing other settings groups 11](#_Toc522008917)

[2.6 Device Status Group 12](#_Toc522008918)

[2.7 Connectivity Group 13](#_Toc522008919)

[2.8 Advanced Configurations Group 14](#_Toc522008920)

[2.9 Security Group 15](#_Toc522008921)

[Chapter 3 Retrieving OPC UA Server Data 16](#_Toc522008922)

[3.1 Using Configuration Group Settings 16](#_Toc522008923)

[3.1.1 Adding OPC UA tags within things 16](#_Toc522008924)

[3.1.2 Verifying OPC UA tags in things 17](#_Toc522008925)

[3.1.3 Adding OPC UA tags in properties 17](#_Toc522008926)

[3.1.4 Verifying OPC UA tags in properties 18](#_Toc522008927)

[3.1.5 Selecting subsets of OPC tags 19](#_Toc522008928)

[3.1.6 The Create Tag Things button 20](#_Toc522008929)

[3.1.7 The Has Tag Thing checkmark 20](#_Toc522008930)

[3.1.8 The Monitor as Property button 20](#_Toc522008931)

[3.1.9 Deadband History Start 21](#_Toc522008932)

[3.1.10 Custom Property Host Thing Name 21](#_Toc522008933)

[3.2 Tag import group 21](#_Toc522008934)

[3.2.1 Limitations 21](#_Toc522008935)

[3.2.2 Exporting tags 22](#_Toc522008936)

[3.2.3 Importing tags 22](#_Toc522008937)

[3.2.4 The Replace existing tags checkbox 23](#_Toc522008938)

[3.2.5 The Bulk apply tags checkbox 23](#_Toc522008939)

[Chapter 4 Other Configuration Settings Groups 25](#_Toc522008940)

[4.1 Reconnects and Timeouts Group 25](#_Toc522008941)

[4.2 Status info and data formats Group 26](#_Toc522008942)

[4.3 Browsing and logging Group 27](#_Toc522008943)

[Appendix A: AXOOM Gate Navigation Icons 28](#_Toc522008944)

[Appendix B: Plugins that ship with AXOOM Gate 29](#_Toc522008945)

[Appendix C: Error and Logging Messages for plugin OPC UA Client 30](#_Toc522008946)

# Introduction to AXOOM Gate Plugins

## Overview

This document covers configuring the OPC/UA Client plugin running in AXOOM Gate (and compatible platforms).

## About AXOOM Gate

AXOOM Gate provides remote and mobile access to industrial production equipment in a secure, efficient, extensible manner. In the interest of operational efficiency and reduced complexity, industrial systems are often configured with data security features disabled. With its built-in, IT-friendly data security settings, AXOOM Gate seamlessly links production systems with office and mobile systems without compromising either factory operation or IT data security requirements.

The installation folder for AXOOM Gate in Microsoft Windows is: C:\Program Files (x86)\AXOOM\AXOOM-Gate. The main executable program is AxoomGateService.exe, which can run both as a Windows service (akin to a Linux daemon) as well as a command-line program.

## About Plugins

An AXOOM Gate plugin is a dynamic-link library (DLL) built with the C-Labs™ C-DEngine™ SDK. Plugins must have a filename with a prefix of CDMy or C-DMy (examples: CDMyCharts.dll and C‑DMyNetwork.dll).

Plugins enable custom features in AXOOM Gate. The many types of plugins include:

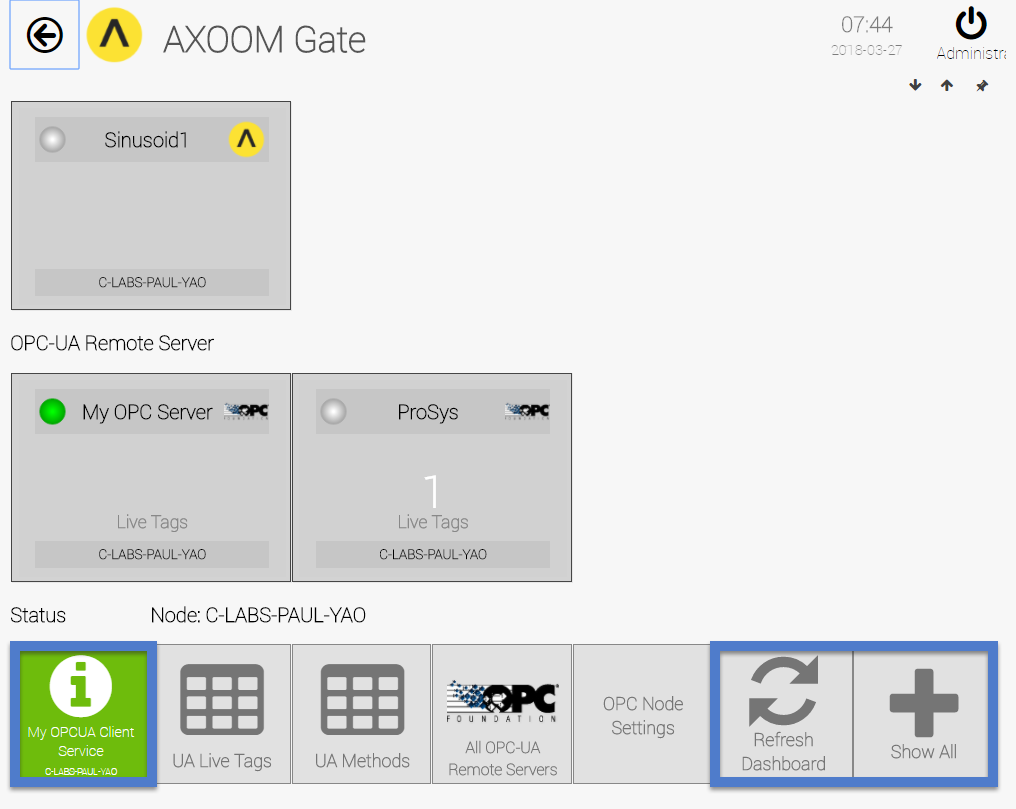
* Configuration Plugins – provide a user interface for configuring hardware or software.
* Connector Plugins – enable a communication channel between AXOOM Gate nodes.
* Device Plugins – enable connections to and data collection from local sensors and devices directly connected to the system running AXOOM Gate.
* User Interface (NMI) Extension Plugins – provide custom controls and other user interface extensions to AXOOM Gate.
* Protocol Plugins – support industry standard protocols like OPC / UA, Modbus, MT Connect, serial ports, and proprietary programmable logic controllers (PLCs) such as the Siemens S7.
* Service Plugins – support other plugins.

As of this writing, there are over 100 plugins. AXOOM Gate v2.102 ships with 22 plugins (see Appendix B for a complete list). Most of these plugins exist to enable live capture of one or more data points from a running system. Such data capture plugins all share a common pattern in terms of how they are configured.

## Plugin Deployment Patterns

To simplify the proper plugin deployment and configuration, this deployment pattern applies to almost every AXOOM Gate plugin:

1. Secure login – A username and password are required to log into AXOOM Gate.
2. Plugin button on home page – All plugins have a button on the AXOOM Gate home page, the first page displayed when you log into AXOOM Gate (see Figure 2.3). Click the plugin’s button to access the plugin dashboard.
3. Plugin dashboard – A plugin dashboard displays all configured items (connections, object, devices, sensors). A plugin dashboard also has a button for creating new items (see Figure 1.1).
4. Editing existing items – Configuring connections involves setting network or device addresses.
5. Once connections are established, you pick specific data points – also known as “properties” – to read.
6. Such properties are grouped together as “Things,” such as you might expect to find in a package that was created to help connect to the “Internet of Things.”

  
**Figure 1.1. The OPC UA Client plugin dashboard.**

## The Plugin Dashboard

A plugin’s dashboard (see Figure 1.1) provides the primary plugin interface. All “live” items appear at the top of a plugin dashboard. A series of buttons appears at the bottom of the plugin dashboard. In the dashboard shown in Figure 1.1, blue rectangles identify the three standard dashboard buttons:

1) About button: The button with the circled “i” is the About button. Click for details on the dashboard. The button color reflects the status. Status colors, associated status code, and meaning of the status, are summarized here:

* + Gray (0): Idle.
  + Green (1): Active / Ok.
  + Yellow (2): Warning.
  + Red (3): Error.
  + Blue (4): Starting / Setup / Ramp Up.
  + Brown (5): Design / Engineering / Configuration.
  + Purple (6): Shutdown.
  + Black (7): Unknown / Unreachable.

2) Refresh Dashboard button: Reloads the dashboard with latest values.

3) Show All button: Open all forms and tables associated with the plugin.

Figure 1.2 shows the default image on a table button. Click a table button to view and edit a table of items managed by the plugin. For example, you can add new items, edit existing item properties, or delete items.

  
**Figure 1.2. Buttons with this table icon enable viewing and editing tables of plugin items.**

# Connecting to an OPC UA Server

This chapter covers connecting to an OPC UA Server, and includes the following topics:

* About the OPC/UA Client Plugin
* Login to AXOOM Gate
* Accessing the plugin dashboard
* Adding connections to an OPC UA server
* Accessing other settings groups
* Settings Groups: Device Status Group, Connectivity Group, Advanced Configuration Group, Security Group

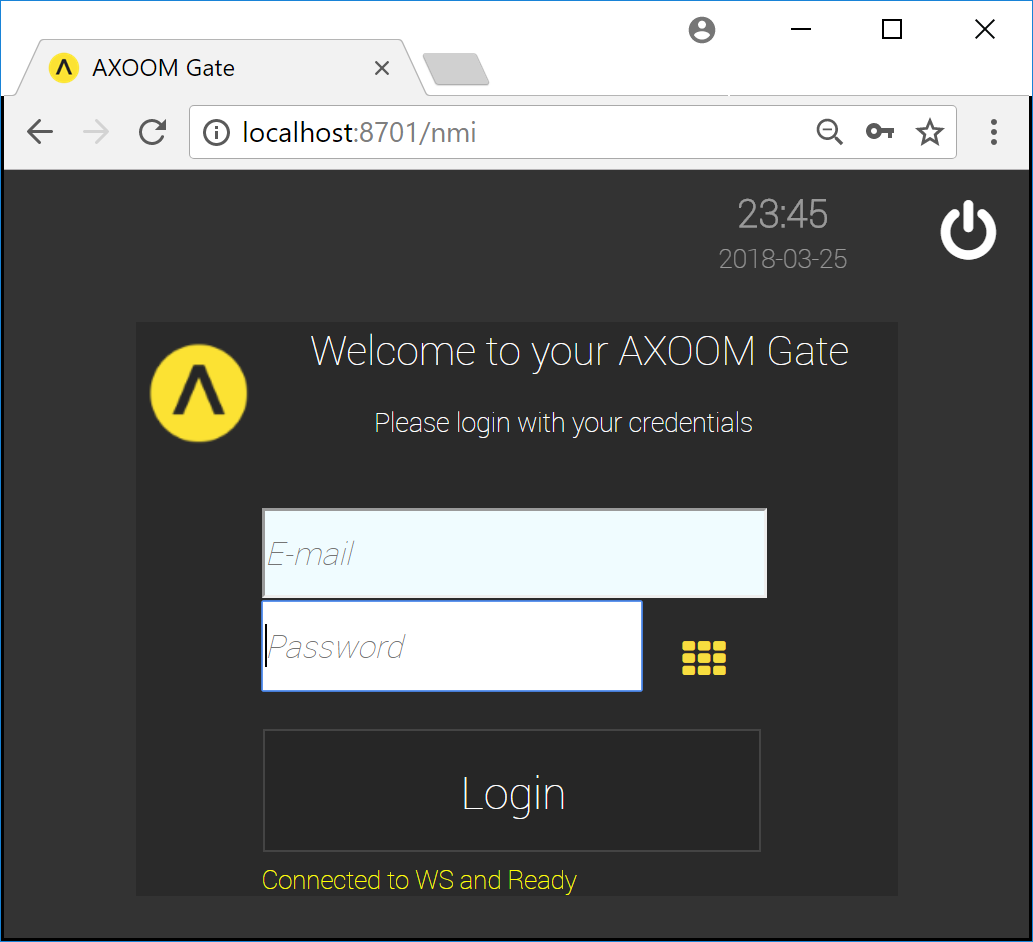
## About the OPC UA Client Plugin

The OPC UA (Unified Architecture) Client plugin enables access to OPC UA Servers using the OPC UA protocol. OPC tags and events can be received and viewed, and OPC UA methods can be invoked using the NMI user interface. The same functionality is also available programmatically to accelerate development of other plugins.

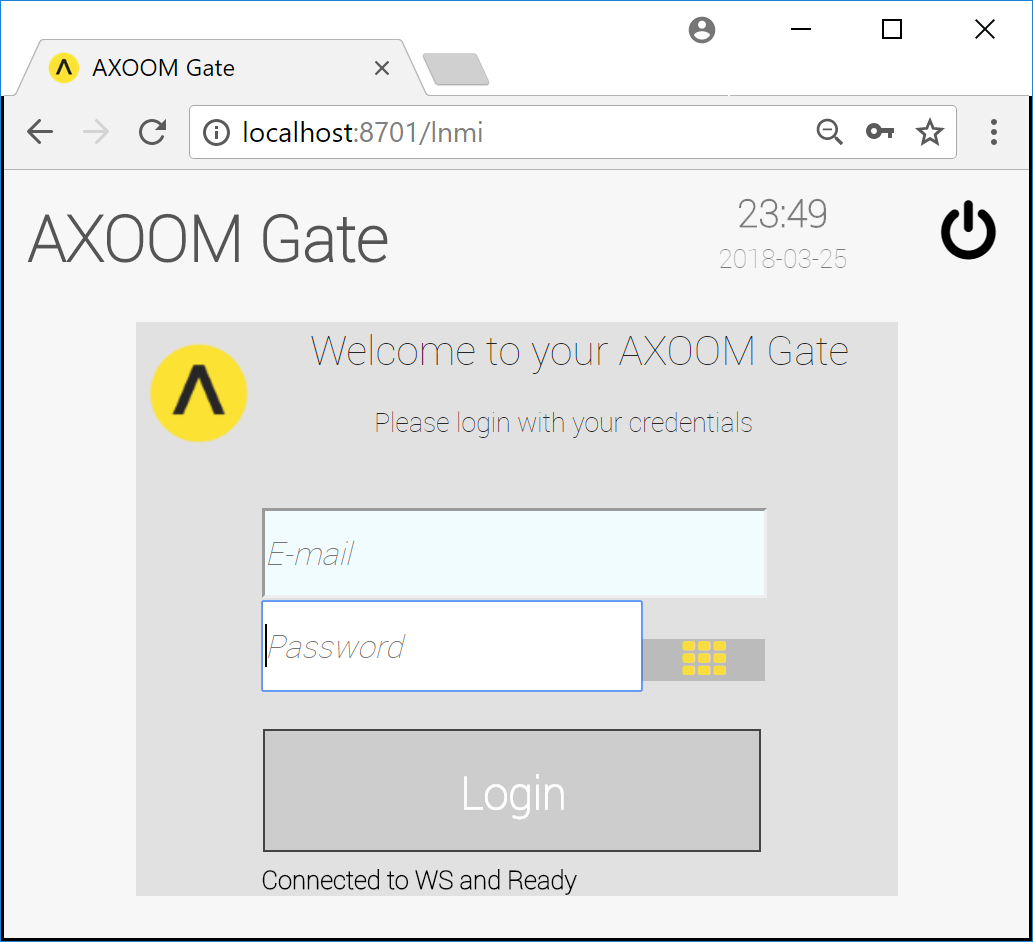
Details on the OPC Unified Architecture can be found at the OPC Foundation website (<https://opcfoundation.org/about/opc-technologies/opc-ua/>).

## Login to AXOOM Gate

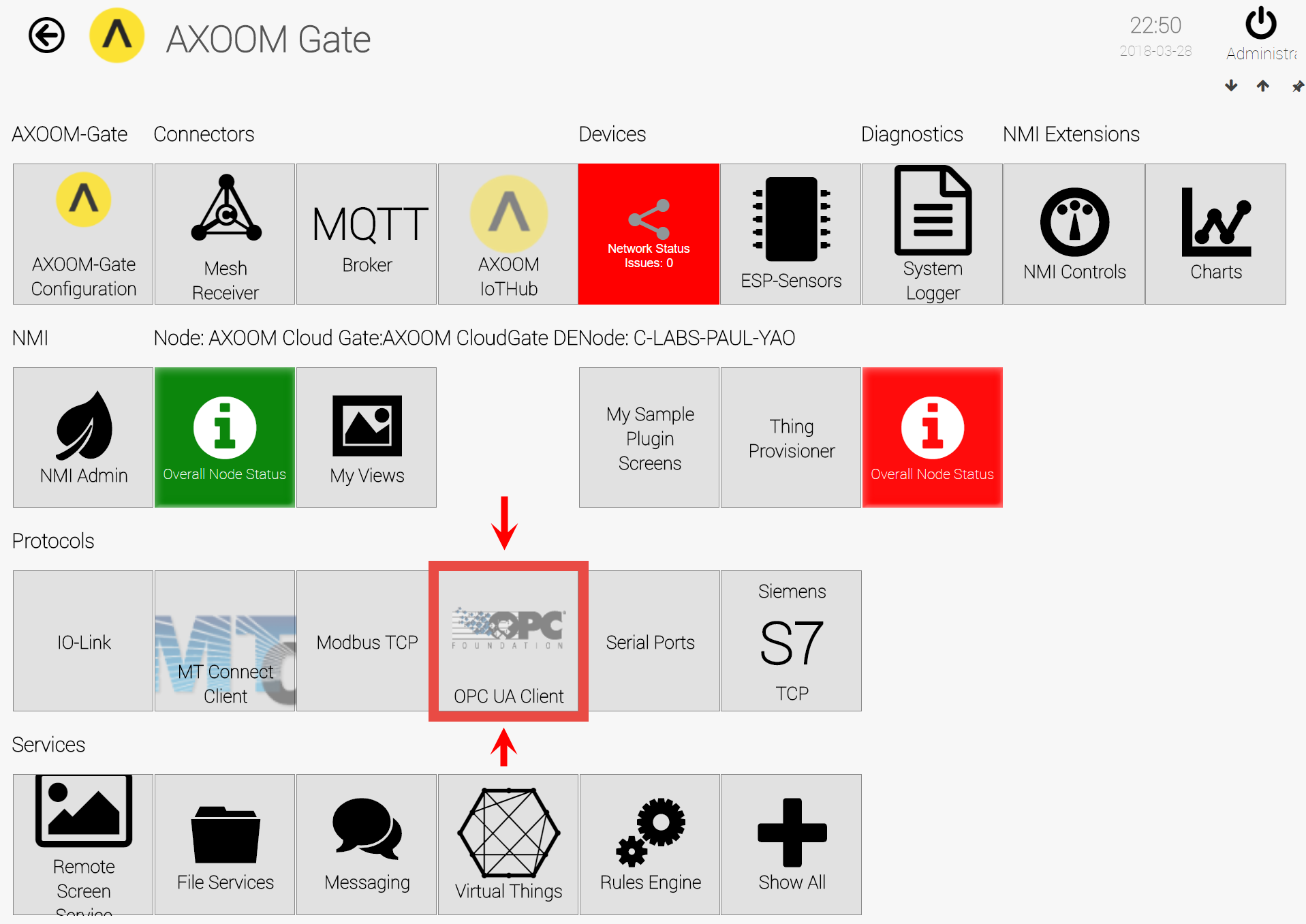
1. Enter this URL, <http://localhost:8701/nmi>, in a web browser. The following login page appears:

  
**Figure 2.1. The dark scheme for the AXOOM Gate login page.**

1. This URL, <http://localhost:8701/lnmi>, enables the “light” display scheme:

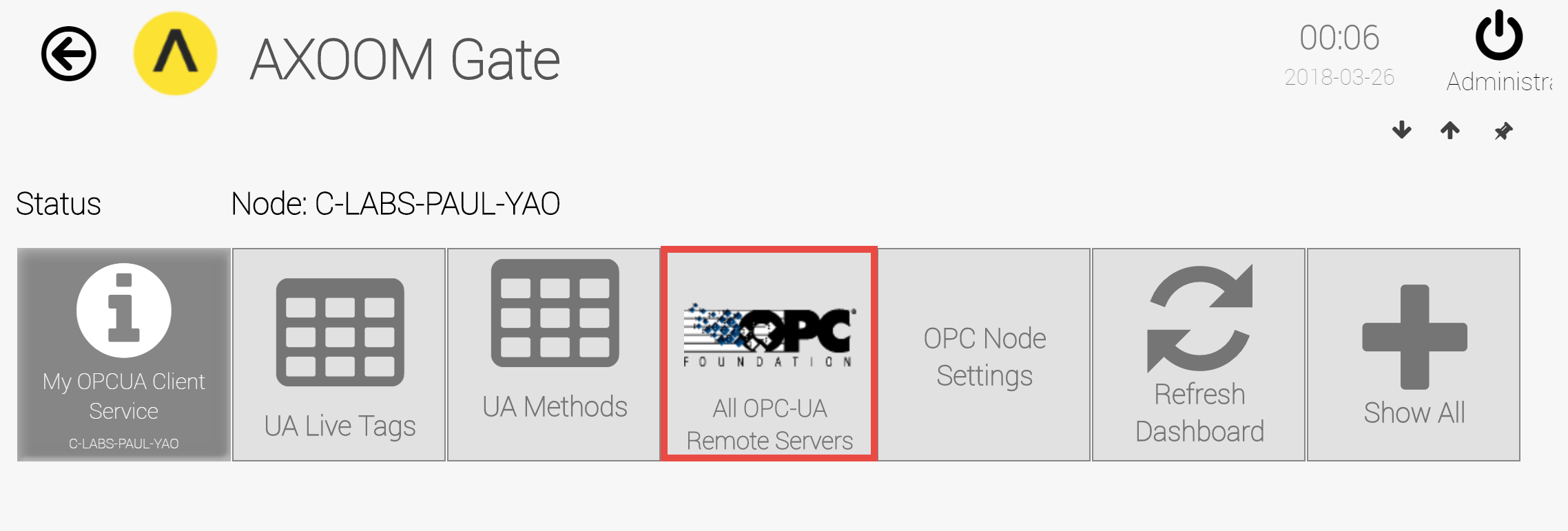
  
**Figure 2.2. The light scheme for the AXOOM Gate login page.**

1. You see the AXOOM Gate home page, as shown in Figure 2.3.

  
**Figure 2.3. The “OPC UA Client” plugin button highlighted on the AXOOM Gate home page.**

## Accessing the plugin dashboard

1. When you click the OPC UA Client button, the OPC UA Client plugin dashboard appears (see Figure 2.4).

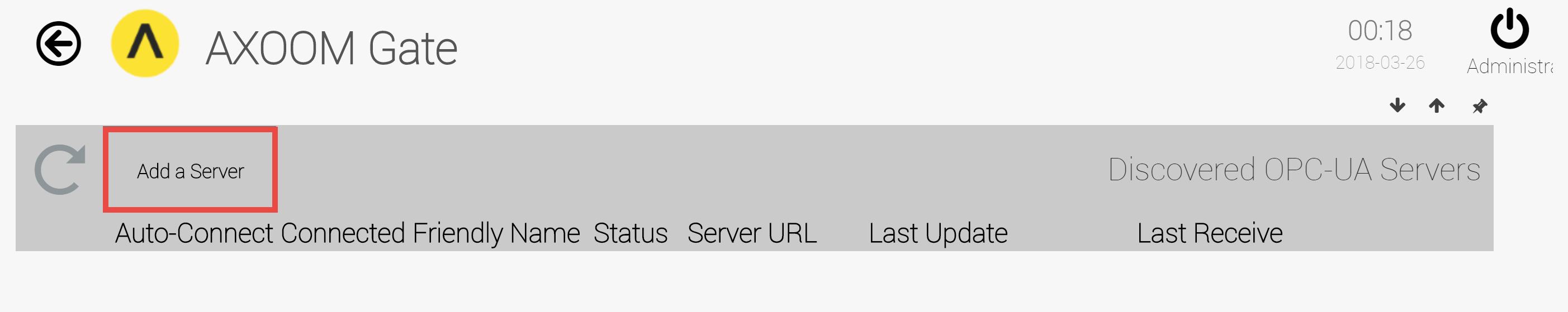
  
**Figure 2.4. The OPC UA Client plugin dashboard.**

1. OPC UA is a client-server protocol. Our plugin provides the client. The first step in setting up this plugin involves establishing a connection to OPC UA servers. To do that, click the All OPC-UA Remote Servers button (highlighted in red).

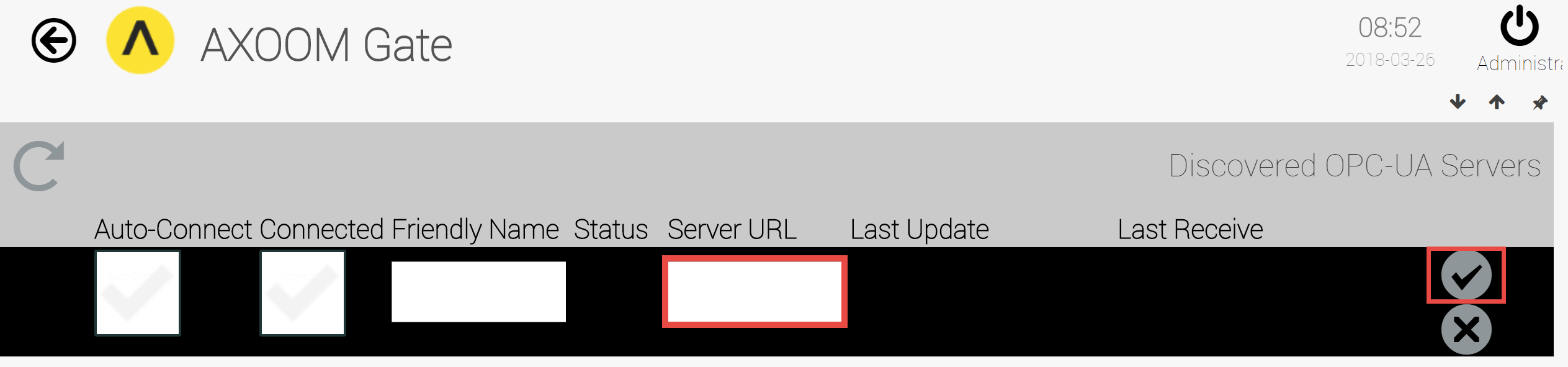
Note: You can enable OPC Server Discovery to locate OPC servers. On the plugin dashboard, click the OPC Node Settings button to find the GLS Discovery settings.

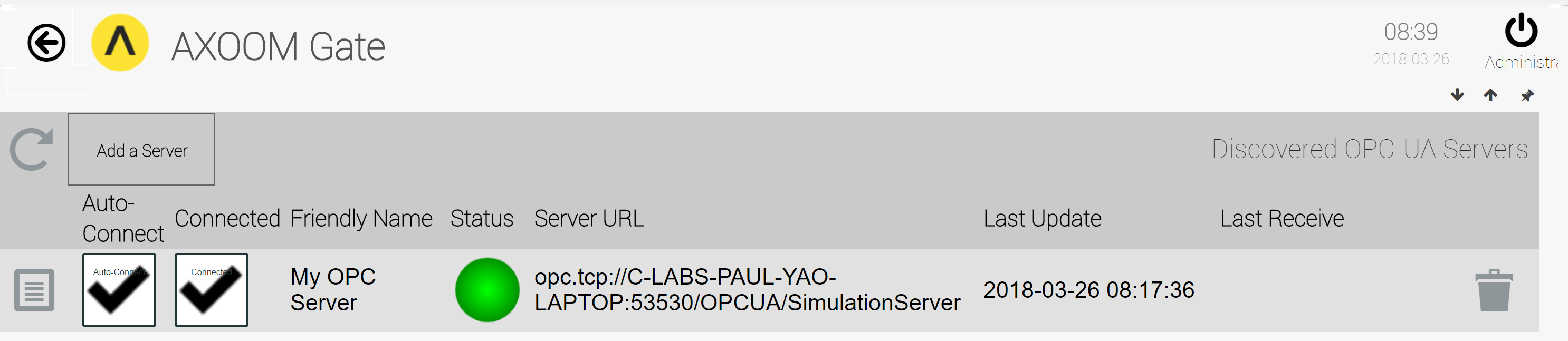
## Adding connections to OPC UA servers

1. Figure 2.5 shows the list of OPC UA servers that have been discovered or which have already been added to the list. To add a server, click the Add a Server button.

  
**Figure 2.5. Click “Add a Server” to manually add a server.**

1. An empty row appears, ready for new OPC UA server details (see figure 2.6).
2. Enter a value in the “Server URL” field, then click the checkmark () to save your work. A new server connection is added to the list (see figure 2.7).

  
**Figure 2.6. Enter server URL. Click checkmark to save.**

  
**Figure 2.7. A connection to an OPC UA Server.**

The round status light will turn green when a connection is established, as shown in figure 2.7. It will be red if no connection has been established. Details about each column in the OPC UA server list are provided in the following table:

| **Column** | **Description** |
| --- | --- |
|  | The Properties icon. Click to view and edit additional server settings. |
| Auto-Connect | Check this box to have AXOOM Gate automatically connect to a server. Without this, you must manually connect after AXOOM Gate restarts. |
| Connected | Whether a server is currently connected (read-only). |
| Friendly Name | A name you pick. In the Device Status Group this is the Device Name. |
| Status | A green circle appears when connected; otherwise a red circle appears. |
| Server URL | URL for the server. |
| Last Update | When the server connection state last changed.  Click the Refresh button () to update the date and time. |
| Last Receive | When the last data item was received. |
|  | Trash can. Delete item on current row. After deleting, refresh the table by clicking the Refresh button (). |

## Accessing other settings groups

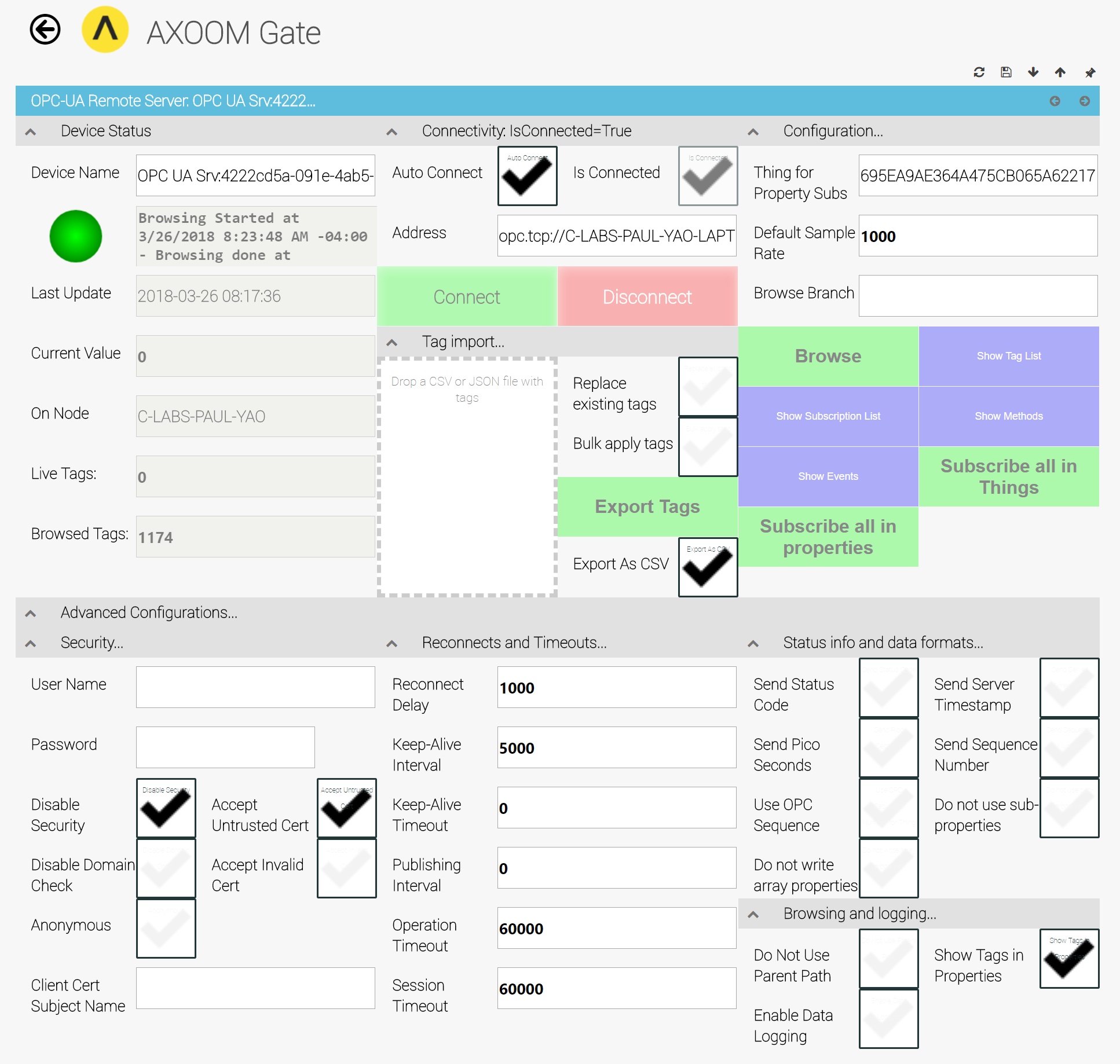
After you create a server table entry, click the properties button () to see more settings. As shown in figure 2.8, when the properties page first appears, one of nine settings groups is visible. Show and hide settings groups by clicking the up and down arrows. These two arrows, plus the left and right arrows, allow you to set the property groups in the browser. This table summarizes these four display buttons:

| **Display Buttons** | **Description** |
| --- | --- |
|  | Up arrow. Closes a group. |
|  | Down arrow. Opens a group. |
|  | Left arrow. Decreases the width of browser space used to display property groups. |
|  | Right arrow. Increases the width of browser space used to display property groups. |

  
**Figure 2.8. When the server properties page first opens, one of nine settings groups is visible.**

Here is a summary of the settings groups, organized by the chapter in which each is discussed.

* Chapter 2 – Establishing a Connection
  + Device Status – Basic OPC UA server details.
  + Connectivity – Connectivity settings plus controls to connect and disconnect.
  + Advanced Configurations – Outer container for four settings groups (Security, Reconnects and Timeouts, Browsing and logging, and Status info and data formats).
  + Security – Set credentials and desired security features.
* Chapter 3 – Retrieving Server Data
  + Configuration – Identify which OPC UA server items to obtain.
  + Tag import – Supports importing and exporting tags.
* Chapter 4 – Other Configuration Settings
  + Reconnects and Timeouts – Set key intervals for polling and disconnecting.
  + Status info and data formats – Set amount of status to provide.
  + Browsing and logging – Navigating the OPC UA hierarchy, and data logging.

  
**Figure 2.9. All groups showing on server properties page.**

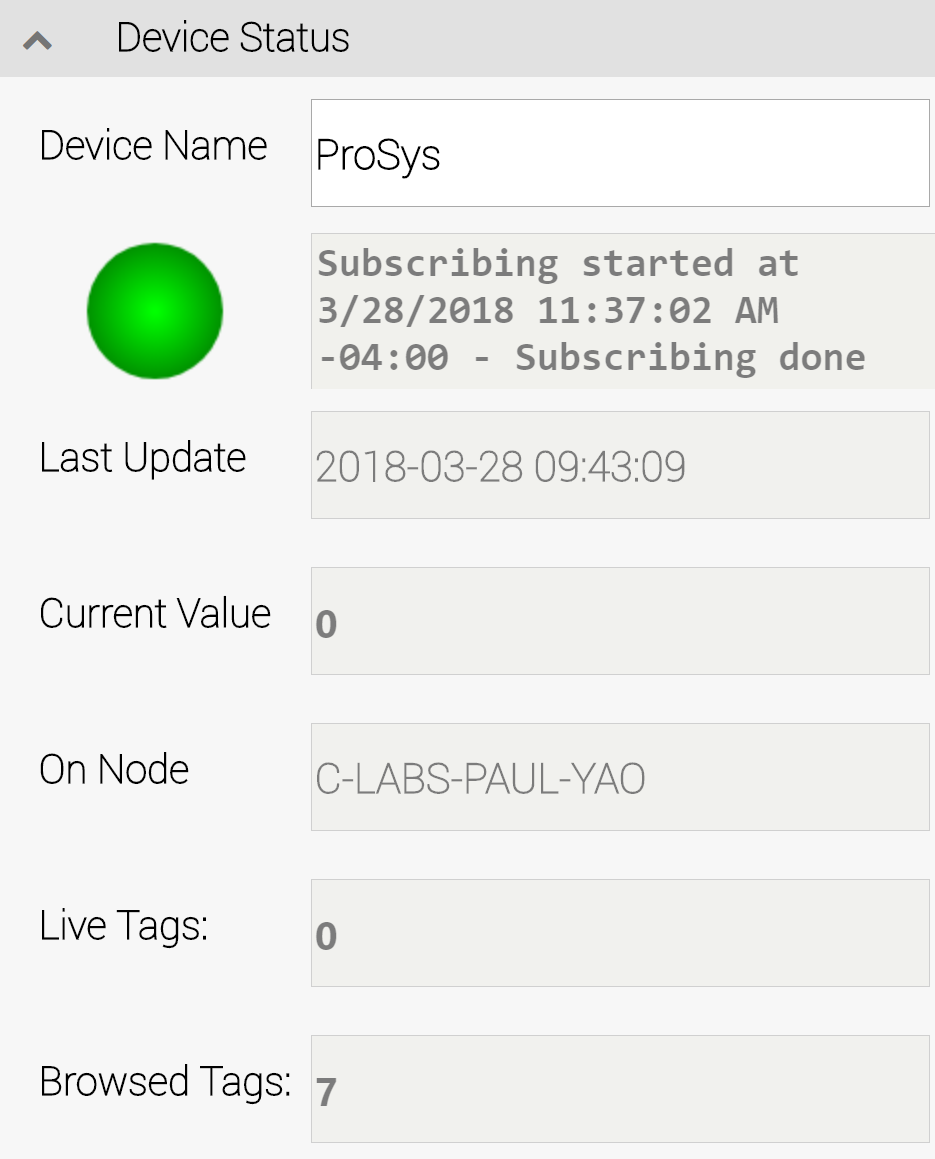
Each group is described in more detail in the sections that follow. Figure 2.9 shows the properties page when all settings groups are opened.

## Device Status Group

The Device Status group (see Figure 2.10) appears by default. It holds basic OPC UA server details.

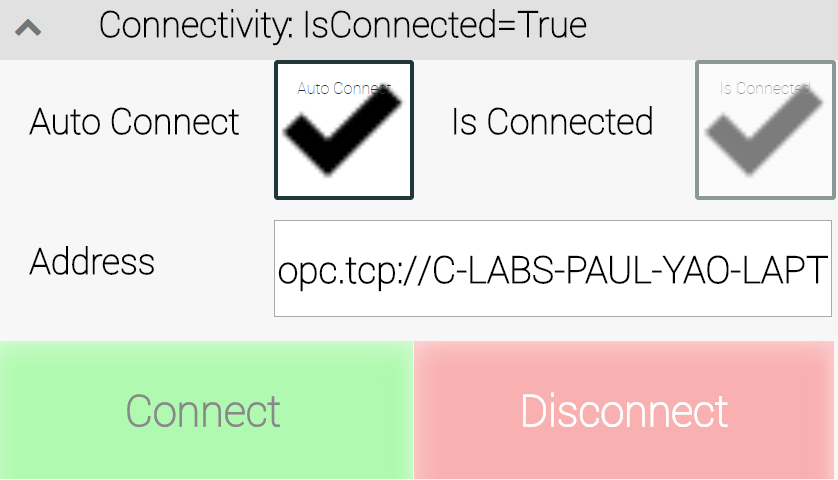
Here are the fields which appear in the Device Status group:

* Device Name – the “Friendly Name.” You can edit this value any time.
* Status light – Green when connected; red when not connected.
* Status text – Details about the current connection when connected; details about connection errors when not connected.
* Last Update – The client-side timestamp when something was received from the OPC UA server.
* Current Value – This value is part of the standard status block and does not apply to OPC UA client.
* On Node – The name of the node where the OPC UA Client plugin is running.
* Live Tags – Count of live tags. This number gets incremented when the OPC UA Client subscribes to a tag from the OPC UA server.
* Browsed Tags – Count of tags encountered in the server. The count of browsed tags reflects the number of tags encountered when the Browse button is clicked (in the Configuration group).

 **Figure 2.10. Device Status group.**

## Connectivity Group

You can check whether your server settings are correct, and whether the server is available, by clicking the Connect button. The indicator labeled “Is Connected” shows the results of your connection attempt. Once connected, click the Disconnect button to close a connection.

  
**Figure 2.11. The Connectivity Group**

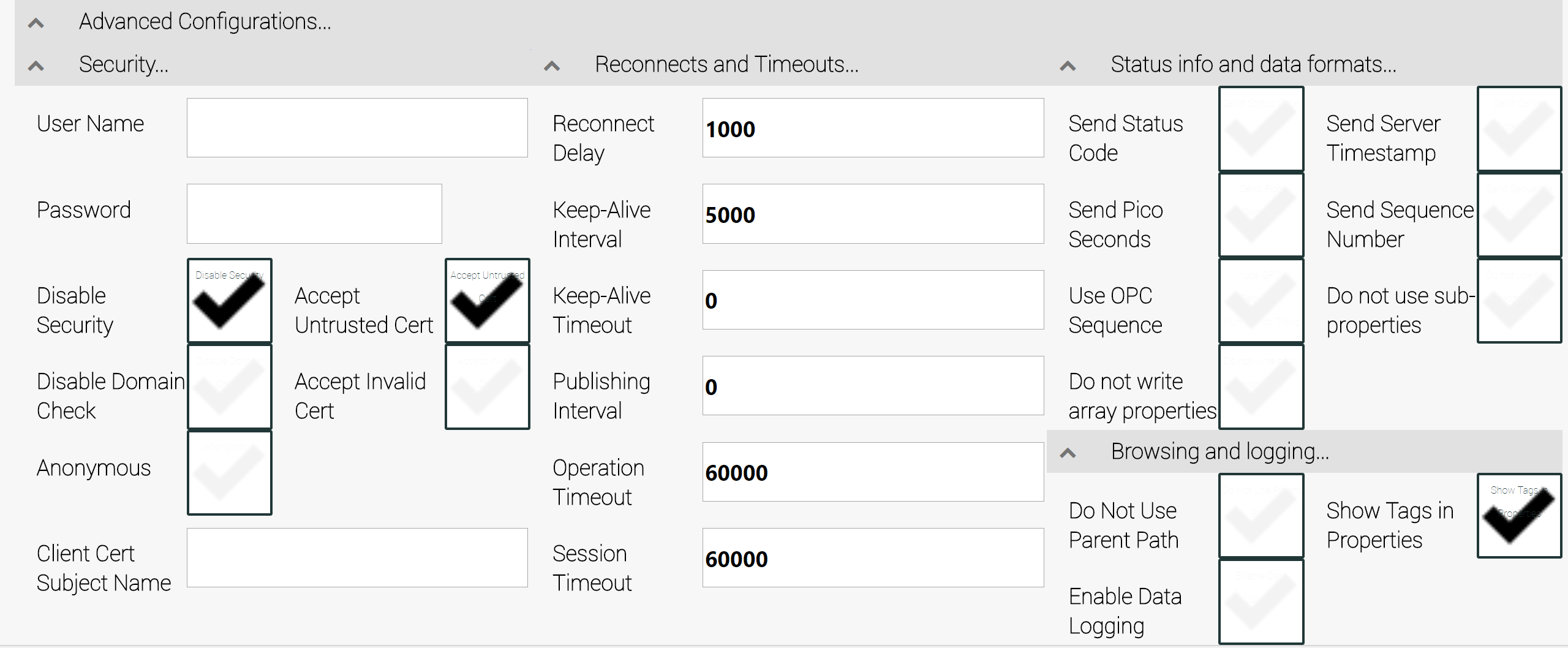
The **Auto-Connect** setting lets you decide whether to always re-establish a connection when the system starts up (when checked), or whether a user must manually establish the connection.

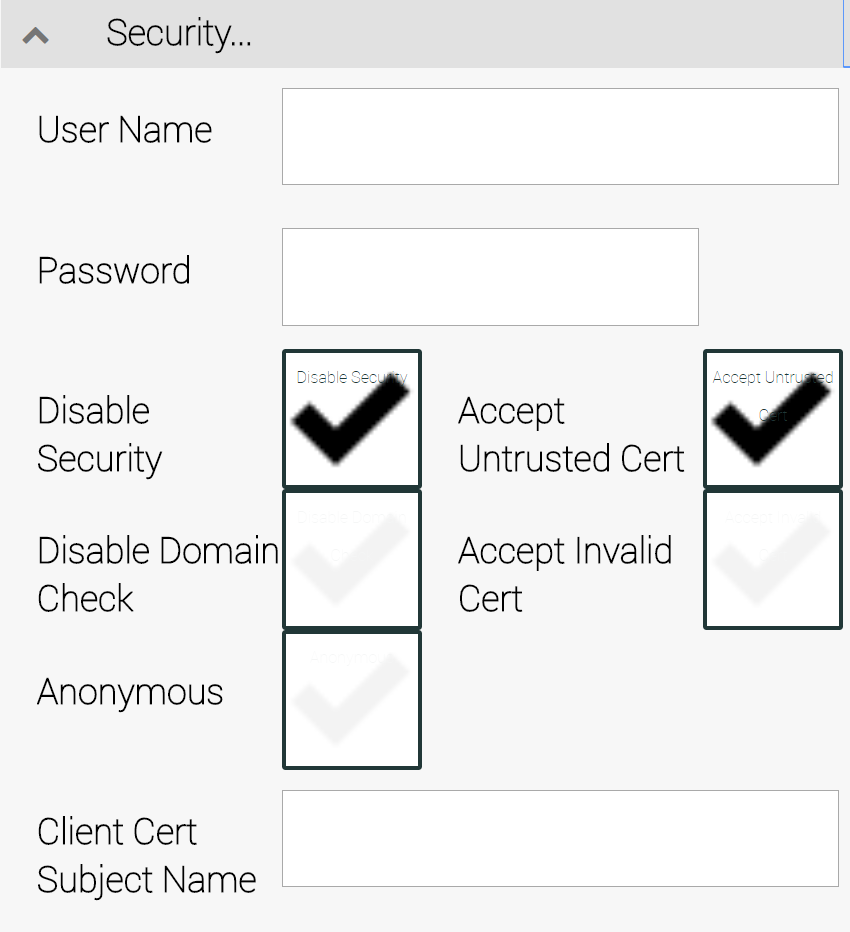
Note: If you have trouble connecting to a server, you might need to adjust the security settings. For additional details, refer to Security Settings group discussion in section 2.8.

## Advanced Configurations Group

This group is a container for the four remaining settings groups:

* Security – Set credentials and desired security features (Chapter 2).
* Reconnects and Timeouts – Set key intervals for polling and disconnecting (Chapter 4).
* Status info and data formats – Set amount of status to provide (Chapter 4).
* Browsing and logging – Navigating the OPC UA hierarchy, and data logging (Chapter 4).

  
**Figure 2.12. The Advanced Configurations Group.**

  
**Figure 2.13. The Security Group.**

## Security Group

The Security Group, shown in Figure 2.13, holds eight security-related fields. By default, every check field is set to **False** (that is, they are unchecked). This sets security at its most stringent. When testing and during the development process, you might need to disable security. In a production environment, enable security to avoid problems caused by unauthorized access to critical resources.

Here are the fields included in the security settings group:

* + - 1. **User Name** – the username used for authentication with an OPC UA server.
      2. **Password** – the password used for authentication with an OPC UA server.
      3. **Disable Security** – when checked, avoid creating and using a self-signed X-509 certificate.
      4. **Disable Domain Check** – when checked, allow a client connection to an OPC server even when the server domain does not match the server certificate.
      5. **Anonymous** – Ignore username / password values when connecting to server.
      6. **Accept Untrusted Cert** – connect with a server even with certificate validation errors.
      7. **Accept Invalid Cert** – connect with a server without validating the certificate chain. This is necessary when, for example, there is no internet access – either because of network problems or because of the deliberate decision to prevent internet connectivity in a location because of security concerns.
      8. **Client Cert Subject Name** – When certificates are used for authentication, this field allows you to replace a certificate’s subject name (an X.509 “Distinguished Name” or DN).

When troubleshooting a server connection, you might need to select the Disable Security checkbox.

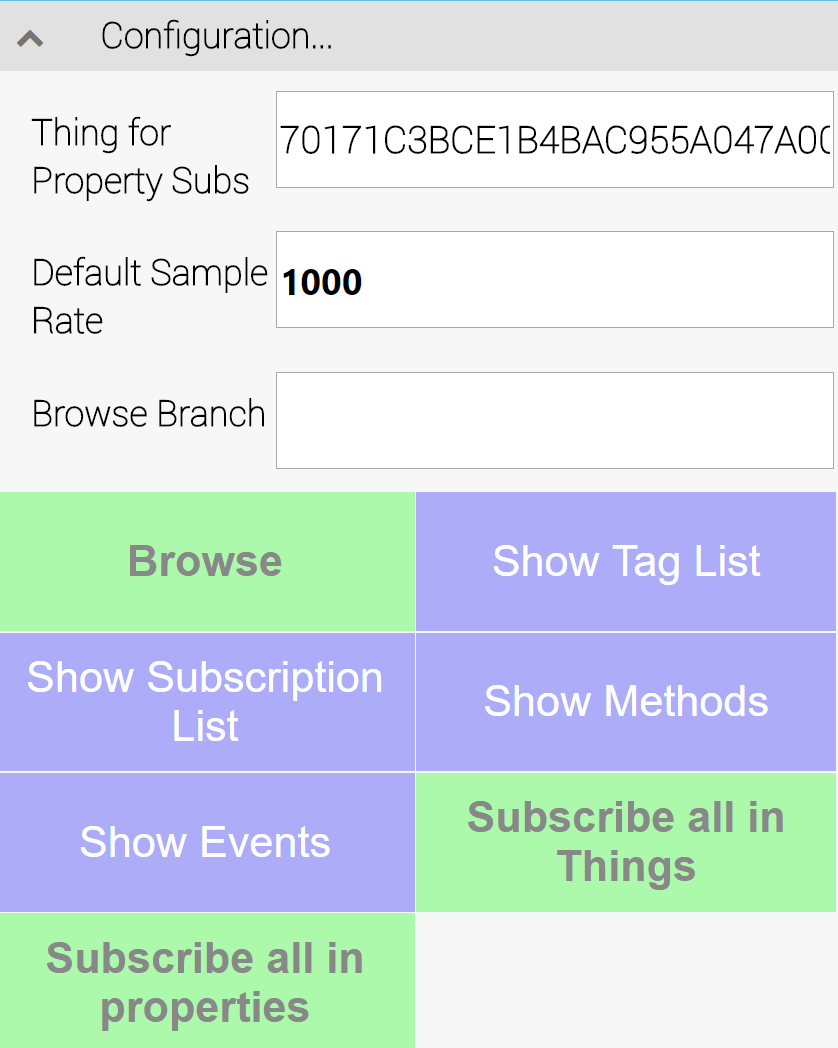
# Retrieving OPC UA Server Data

This chapter covers retrieving data from an OPC UA Server, and includes the following topics:

* Using Configuration Group settings
* Using Tag Import Group settings

## Using Configuration Group Settings

The Configuration Group allows you to access the items from a connected OPC UA server.

  
**Figure 3.1. The Configuration Group**

There are three input fields:

* Thing for Property Subs – A GUID indicating the object (or “Thing”) to be used for a “Subscribe All in properties” operation (the last button in this group).
* Default Sample Rate – The sample rate (in milliseconds) requested from the server for new subscriptions.
* Browse Branch – This field allows for the specification of a portion of an OPC UA tag hierarchy. This reduces the set of tags that are returned to a subset, making it easier to locate the desired tags when creating subscriptions. Example: Objects.Simulation.

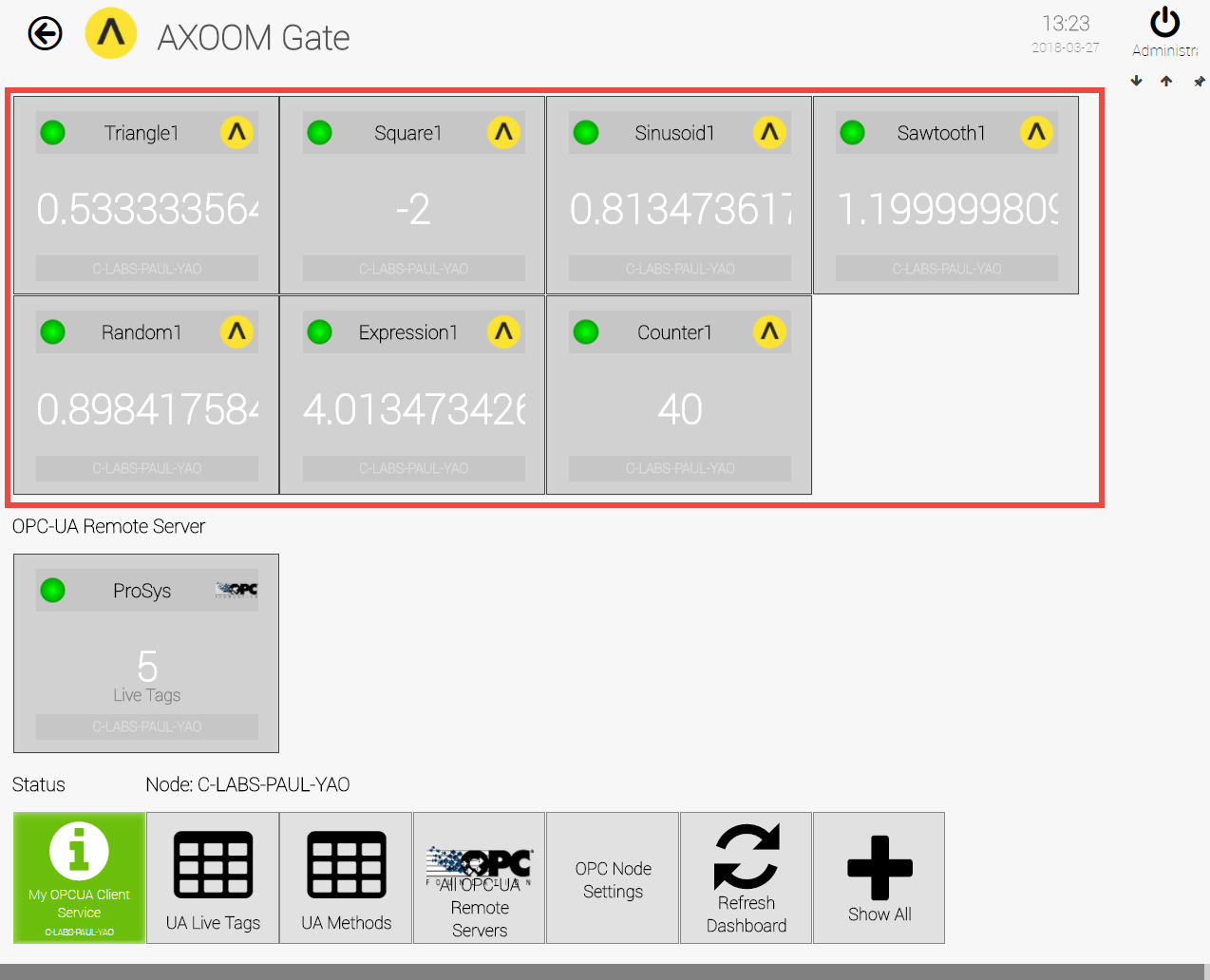
### Adding OPC UA tags within things

The simplest and easiest way to subscribe to tags is to take all tags and provide each one with its own AXOOM Gate thing. An AXOOM Gate thing is a general-purpose container used by AXOOM Gate for any real-world entity including a device, a sensor, or even an OPC tag value. To create one thing per OPC tag, click the Subscribe all in Things button. This works best when there are not too many (< 500) tags.

Before clicking Subscribe all in Things button, check the number of items in the “Browsed Tags” field of the “Device Status” settings group. That is the number of things that will be generated.

### Verifying OPC UA tags in things

To see the set of “Things” that have been created, go to the dashboard for the OPC UA Client plugin. Figure 3.2 shows an example of the things created from the Prosys OPC UA Simulation Object.

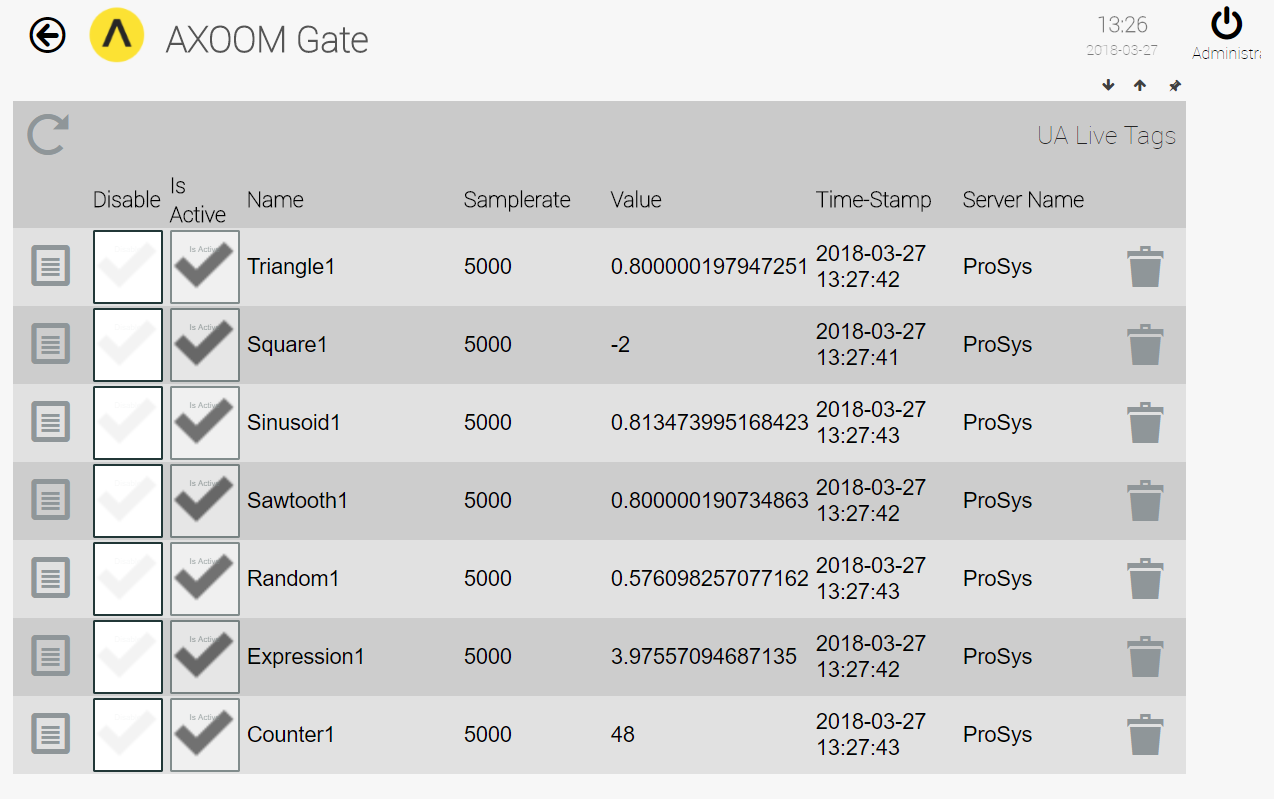
  
**Figure 3.2. The seven things created from the** Prosys **OPC UA Simulation Object.**

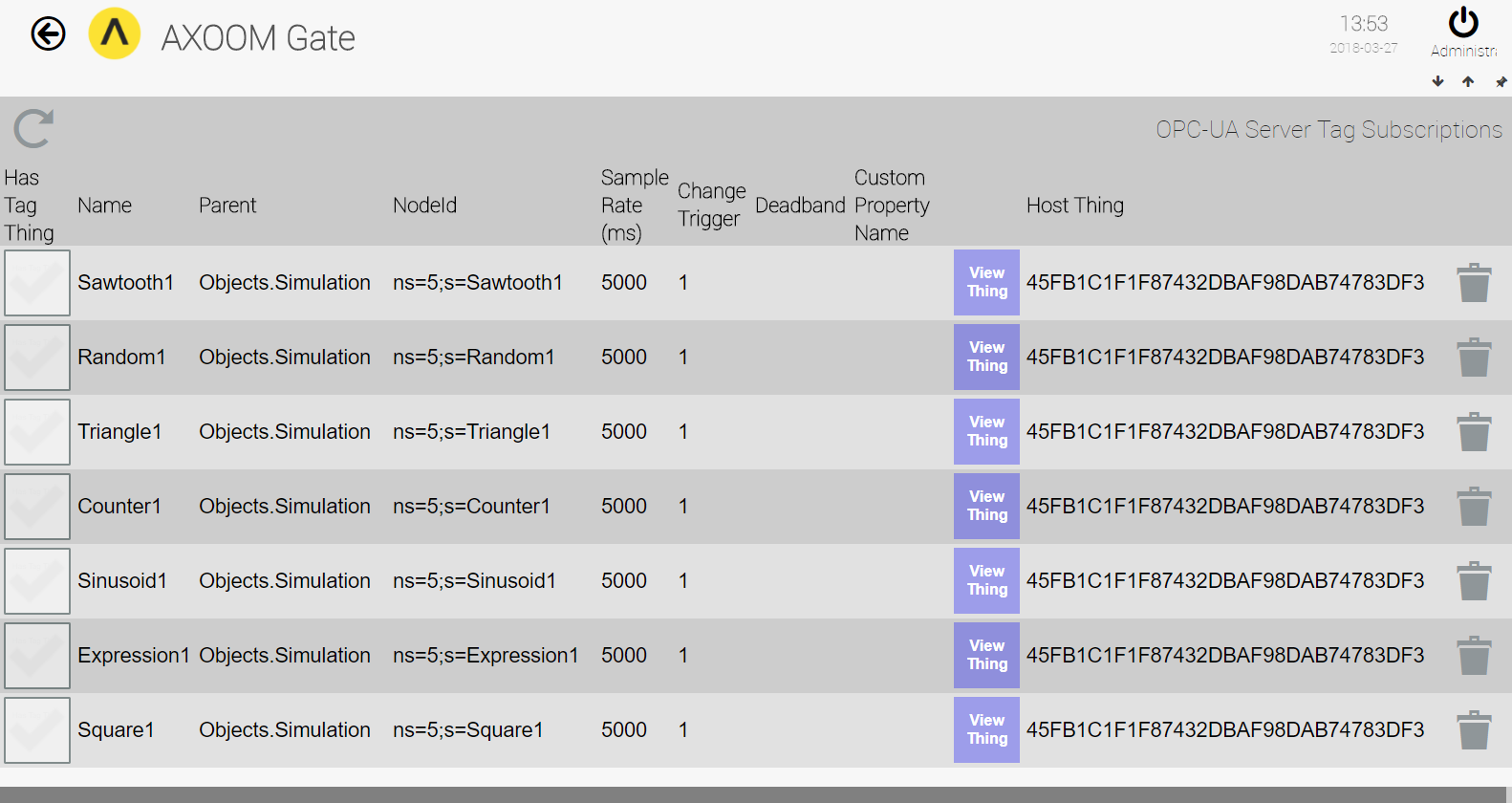
To see a list of the available tags, you can also click on the UA Live Tags button on the OPC UA Client plugin dashboard. Figure 3.3 shows an example of the list that is available. Notice that this is a table view of these list items, enabling you to edit properties, disable (or enable) items, and to delete tags.

### Adding OPC UA tags in properties

In settings where there are many – hundreds or even thousands – of tags, AXOOM Gate provides the best performance with tags linked to properties instead of individual things. A single AXOOM Gate thing contains multiple properties, and so this approach saves system memory and processing time. You assign OPC tags to AXOOM Gate properties by clicking the Subscribe all in properties button.

While the title of this section is “Adding OPC UA tags in properties,” it would have been more accurate to use the title “Adding OPC UA tags in properties within things.” But that would have been too wordy.

  
**Figure 3.3. Verifying the OPC UA tags in things that were created.**

  
**Figure 3.4. Subscription List for Tags in Properties.**

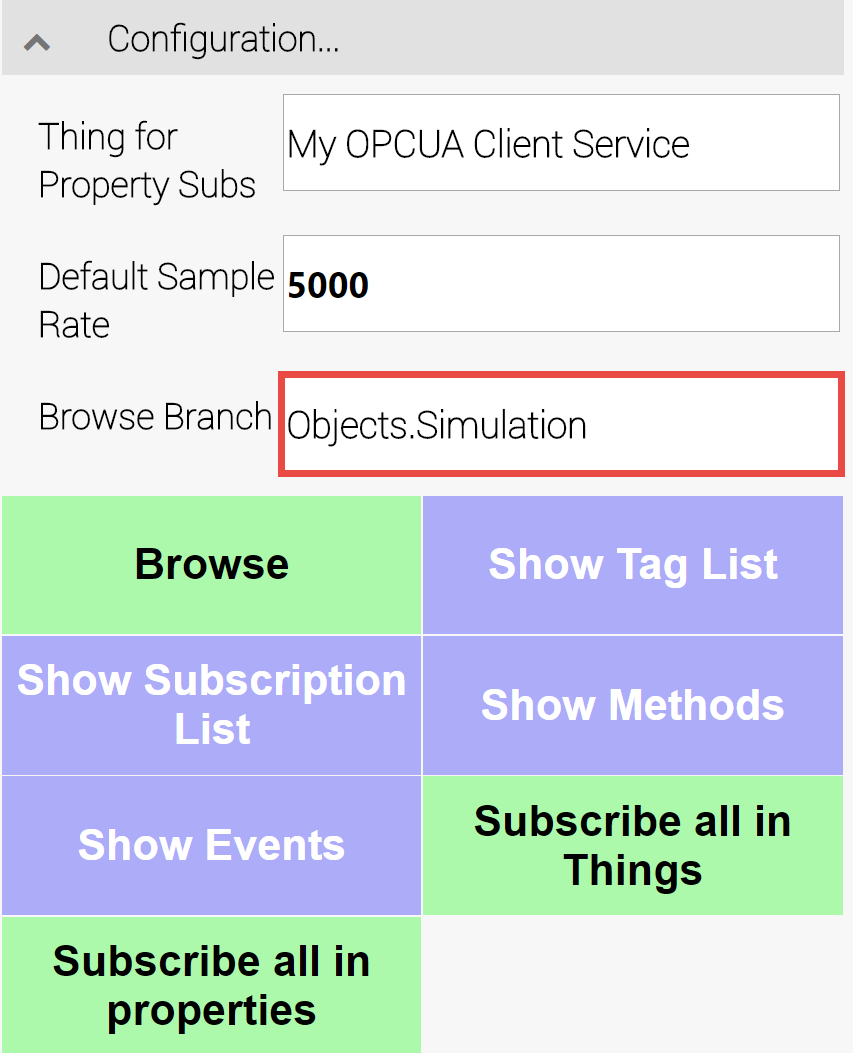
### Verifying OPC UA tags in properties

To verify that the tags were, in fact, created, click on the “Show Subscription List” in the Configuration settings group. Figure 3.4 shows an example like the list you might see. There are a few items worth pointing out in this figure. One is the column labeled “Has Tag Thing.” There is no checkmark for any item in this column. The reason, of course, is that each of these seven tags are properties of a single “thing,” and are not themselves “things.”

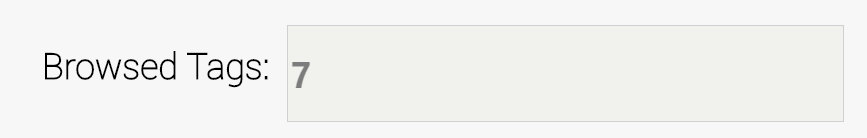
In the column labeled “Host Thing,” notice that the values in each row are identical. Things are identified by a GUID. A GUID (globally unique identifier) is a very big number. GUIDs have many uses. In the context of AXOOM Gate, a GUID is used to identify all sorts of things, including machine nodes, users, and things. The thing that is hosting each of these tags is the same, which confirms what was stated earlier: multiple properties with tag information are hosted by a single “thing.”

### Selecting subsets of OPC tags

A third approach to adding tags involves manually filtering the selected tags. For example, you can enter an OPC/UA tag branch in the **Browse Branch** field, and then click the Browse button. In the following example (created using the Prosys OPC UA Simulation Server), we enter Objects.Simulation in the “Browse Branch” field and click the Browse button.

  
**Figure 3.5. The Configuration group set up to browse a portion of available server tags.**

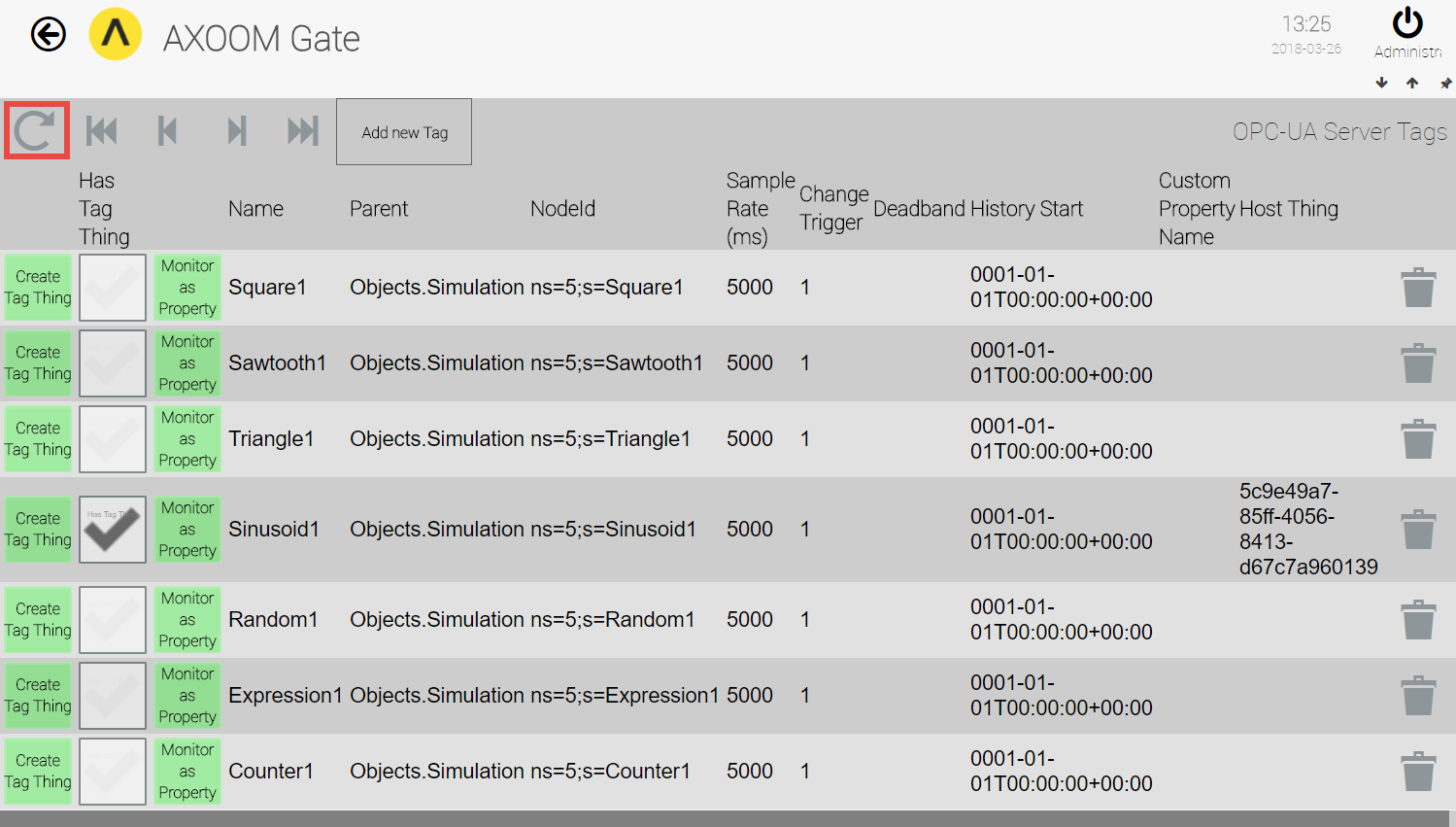
After clicking the Browse button, the value in the “Browsed Tags” field is 7 – reflecting the number of simulation fields provided by the server. Figure 3.6 shows the Browsed Tags field.

  
**Figure 3.6. Number of tags found in the** Objects.Simulation **branch.**

To see the results of browsing, you can click one of three buttons:

* Show Tag List – to see available data tags.
* Show Methods – to see available methods.
* Show Events – to see available events.

No matter which one you select, be sure to click the Refresh button when the page appears. Figure 3.7 shows the tags that are found after clicking the Show Tag List button (and refreshing the page).

  
**Figure 3.7. Tags from the** ProSys **OPC UA server simulator.**

### The Create Tag Things button

Each row represents a different OPC UA tag. Each row contains two different buttons: “Create Tag Thing” and “Monitor as Property.” What is the difference between these two? The first set of buttons creates tags as things, in the same way that the Subscribe all in things button did. In this case, of course, clicking a single button in a row creates a thing for a single tag, and not multiple items.

### The Has Tag Thing checkmark

The presence of a checkmark indicates that a tag thing has been created for the tag in that row.

### The Monitor as Property button

When the Monitor as Property button is clicked, a property is created for that OPC UA tag in the thing indicated in the Host Thing field (or if none is specified, the “Thing for Property Subs” from the Configuration group is used). This button generates similar results to clicking the Subscribe all in properties button. One difference is that a single property is generated for a single tag, instead of having multiple properties created for all the browsed items.

One challenge when configuring an OPC UA client is setting up the specific OPC UA tags, especially when there are many tags to set up. One solution is to subscribe to all tags, but this can create performance issues when there are many servers or many tags. Another solution is to create a list of tags to create, and then use this list to automate the tag creation process. The “Tag import group” provides this service.

### Deadband History Start

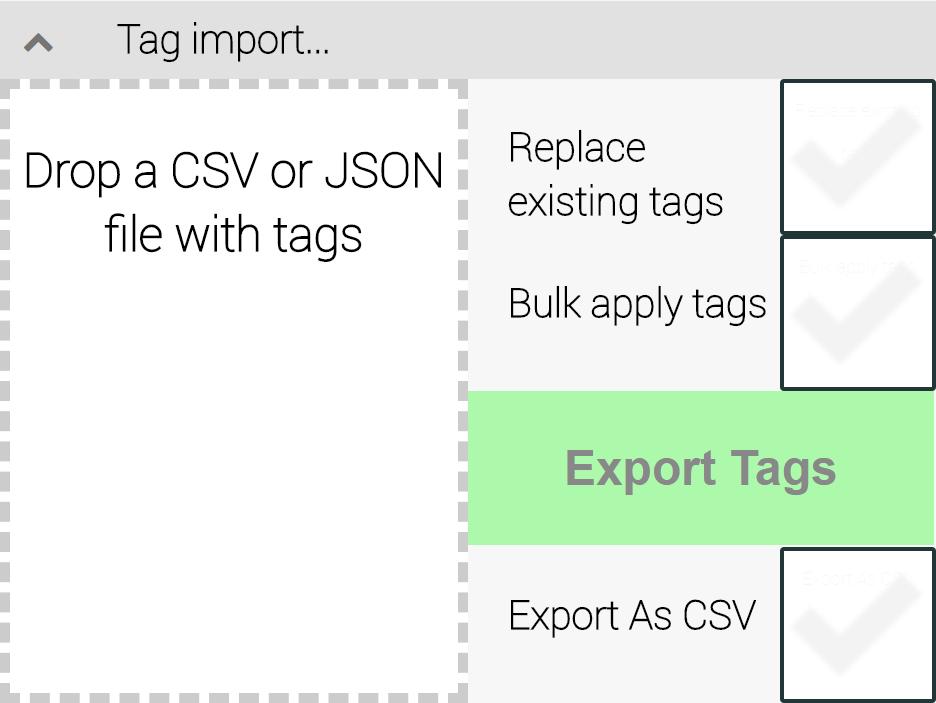
This column indicates the date and time when the history for a specific tag should begin. When this value is defined (meaning when it is not equal to January 1, Year 1 Time 00:00:00), the client connects to the OPC server with a request from an Historical Data Access (HDA) server to perform a one-time backfill of the timeseries.

### Custom Property Host Thing Name

This column indicates the name that has been chosen for a property that is different from the OPC UA tag name, the default name used for a tag.

## Tag import group

This group supports importing and exporting tags from the OPC UA Client plugin.

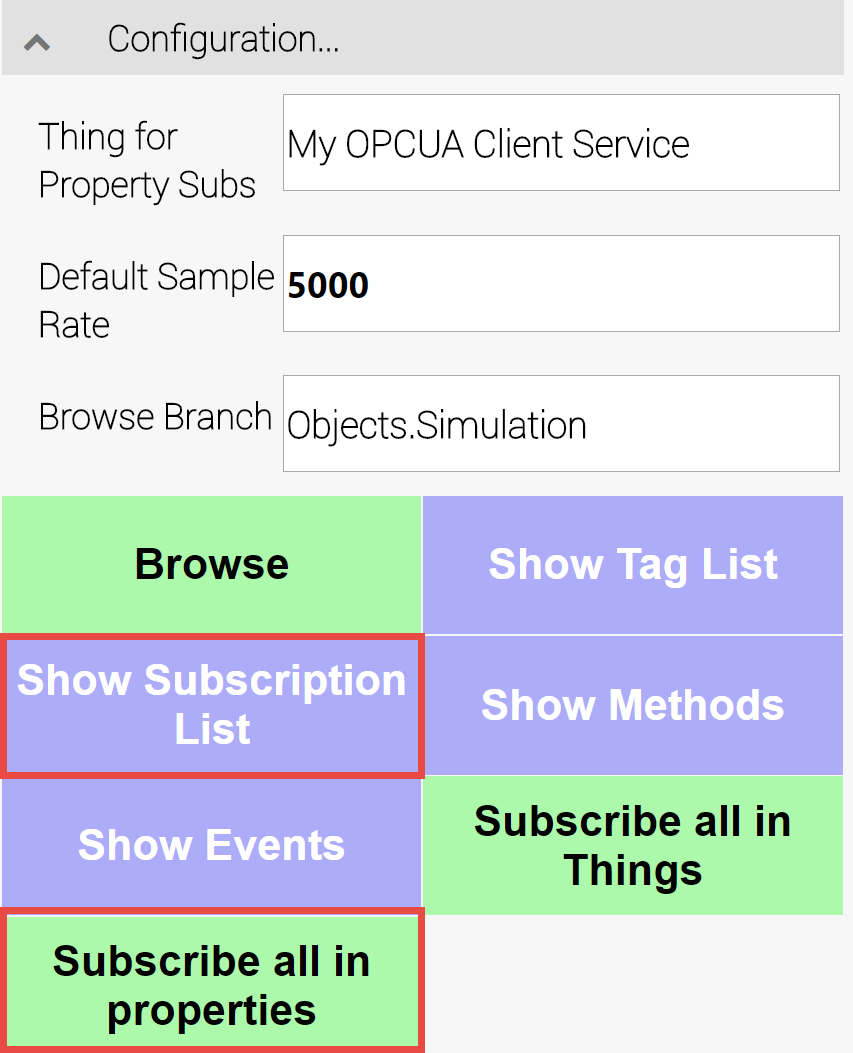
  
**Figure 3.8. The Tag import group**

### Limitations

The import / export process has a few limitations worth mentioning:

* You can only import / export tags created as “properties of things.”
* You cannot import / export tags created as “things.”
* Two file formats are supported for importing and exporting tags: JSON and CSV (comma-separated values).

None of these limitations prevents you from importing or exporting. At most they represent minor inconveniences. The “properties of things” limitation means that a set of tags are associated with a specific server. As such, importing, exporting, viewing tags, and editing tags are all handled on the settings pages for a specific server. These tags do not appear on the plugin’s main dashboard.

  
**Figure 3.9. Check for Properties of Things with the highlighted “Show Subscription List” button.   
Create Properties of Things subscriptions with the highlighted “Subscribe all in properties” button.**

### Exporting tags

Before starting tag export, check for tags as properties by clicking on the Show Subscription List button (in the Configuration settings group – see Figure 3.9). When the tag list shows up, make sure to click the Refresh button (see Figure 3.10). If there are no “Properties of Things” tags, you can create some by clicking on the Subscribe all in properties button (see Figure 3.9).

Note: This assumes a few things, namely: (1) you are connected to an OPC server (2) you have some “Browsed Tags” listed in the “Device Status” settings group.

Once you are sure that there are “Properties of Things” tags defined, click the Export Tags button in the “Tag imports” settings group. An automatic file download is started, in either JSON (the default) format or in CSV format, depending on whether the Export as CSV checkbox is checked or not.

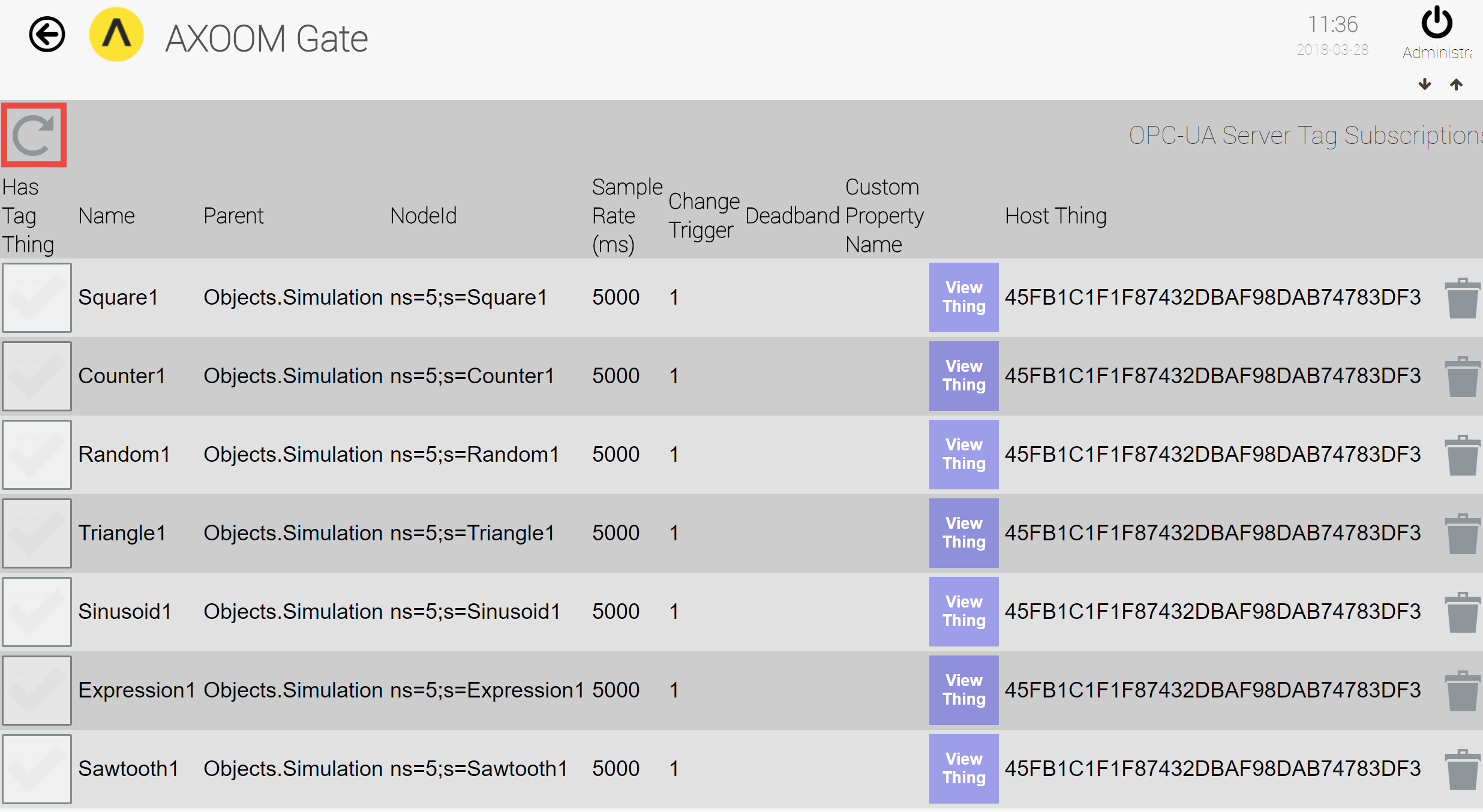
The exported JSON file’s contents appear in Figure 3.11. This file has been formatted for ease of reading. It is also worth noting that only two tags are represented in this example.

NOTE: The export feature does not work with Internet Explorer on Windows Server Operating System because IE is in protected mode and does not allow to download files.

### Importing tags

To import tags, drag a JSON file from the Windows File Manager and drop the file in the drop area. This is the area in the “Tag import” settings group that is surrounded by a dashed line, with the instructions “Drop a CSV or JSON file with tags.”

To check the results from importing tags, click on the Show Subscription List button in the “Configuration” settings group. When the page with the table of items appears, make sure to click the Refresh button so that you see the most up-to-date list.

  
**Figure 3.10. This page appears when you click the “Show Subscription List” button. When it appears, make sure to click the Refresh button (highlighted in red).**

### The Replace existing tags checkbox

This setting affects the import of tags. When it is checked, all existing tags are cleaned out prior to importing the new set of tags. In doing so, the set of imported tags are viewed as the desired state of the client in terms of tag subscriptions. When this setting is unchecked, the imported tag subscriptions are added to the existing set. (The value for the ReplaceAllTags variable in the import file enables you to achieve the same result as for setting the checkbox.)

### The Bulk apply tags checkbox

This setting affects the import of tags. When checked, it causes all tag subscriptions to be submitted to the server in a single request. When there a larger number of tags, having the subscriptions made in a single request is significantly faster than requesting the subscriptions one at a time. One disadvantage to a bulk subscription request is that error messages that are returned are less granular, which makes it more difficult to identify the causes for failed subscription requests. (The value for the BulkApply variable in the import file enables you to achieve the same result as the setting in the checkbox.)

  
**Figure 3.11. The (formatted) contents of an exported JSON file.**

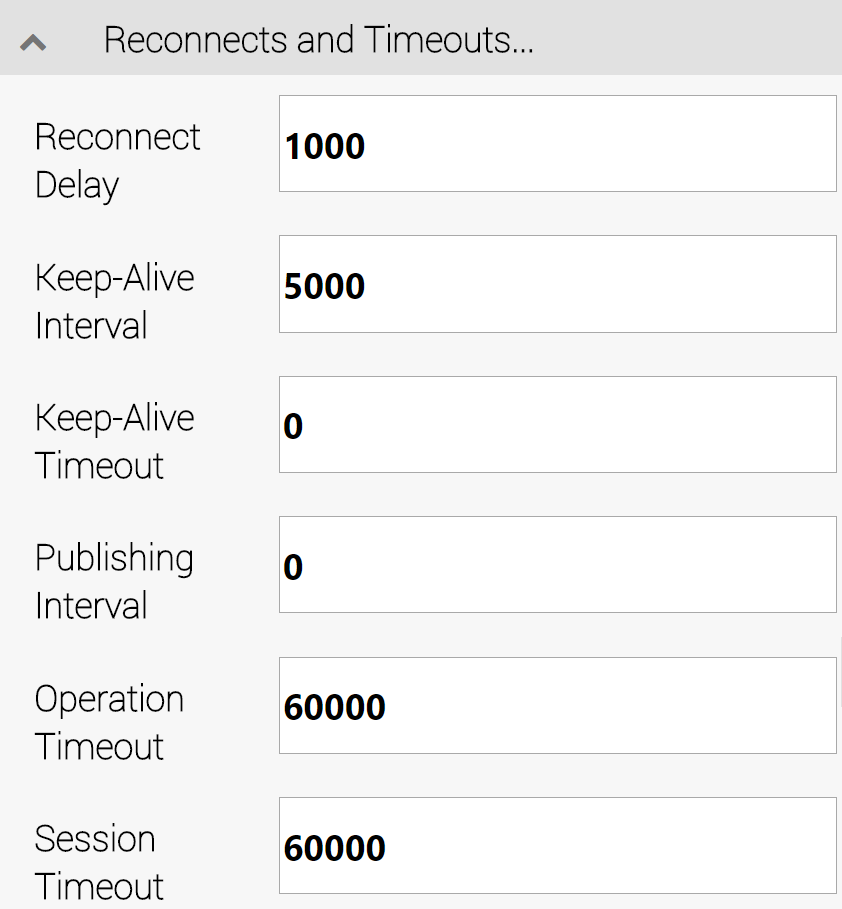
# Other Configuration Settings Groups

This chapter covers other OPC UA client plugin settings groups, including:

* Reconnects and Timeouts Group
* Status info and data formats Group
* Browsing and logging Group

## Reconnects and Timeouts Group

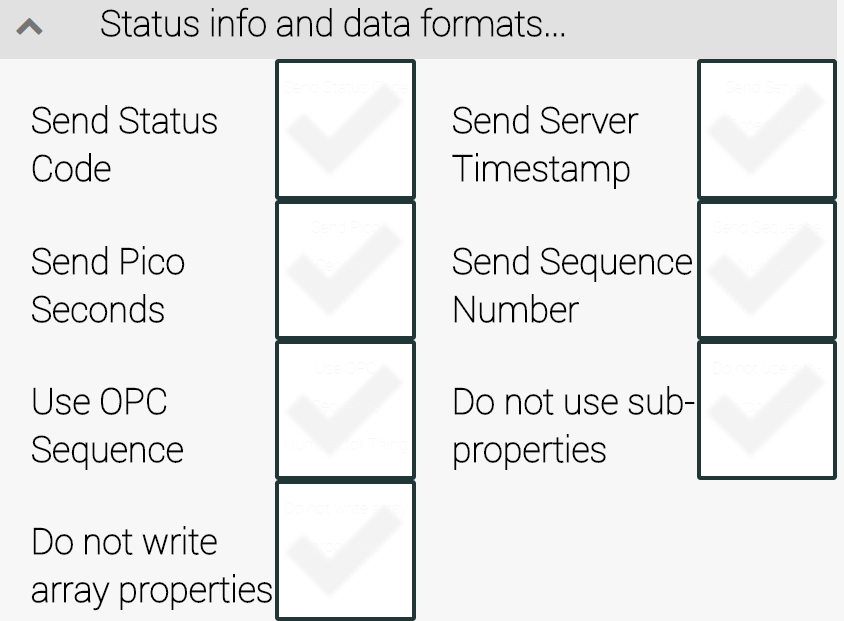
There are six timing-related fields in this group.

  
**Figure 4.1. The Reconnects and Timeouts Group.**

1. Reconnect Delay – the amount of time (in milliseconds) to wait before attempting to reconnect after a connection is dropped.
2. Keep-Alive Interval – the minimum amount of time (in milliseconds) after which to send a keep-alive message to the server when no other data is received.
3. Keep-Alive Timeout - the minimum amount of time (in milliseconds) to wait before declaring a server connection as disconnected (and initiate reconnects if so configured). This is an internal OPC UA keep-alive timeout.
4. Publishing Interval – The frequency (in milliseconds) that an OPC Server will make available data changes.
5. Operation Timeout – the amount of time (in milliseconds) to allow for individual client to server operations.
6. Session Timeout – the amount of time (in milliseconds) to wait before closing a channel server.

## Status info and data formats Group

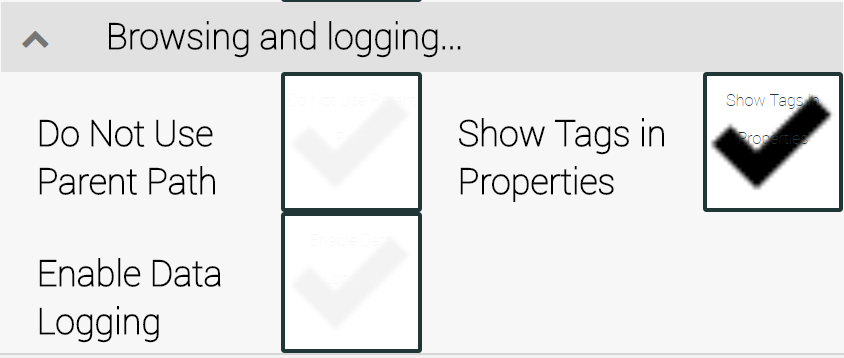
The following three settings enable the capture of status codes and timestamps, as specified in the OPC UA Specification Part 4: Services. Any additional requested item is stored as a property with an AXOOM Gate thing for tags that are “Thing Tags” and not “Property Tags.”

  
**Figure 4.2. The Status info and data formats Group.**

* + - Send Status Code – Whether to request the 32-bit OPC server status code. OPC server status codes are documented in the OPC UA Specification Part 4: Services, Chapter 7, Section 7.34.1). Retrieving this status code can provide additional details needed to help resolve server-side issues.  
        
      By default, status codes are stored as a sub-property of the data item called statusCode. If, however, there is a checkmark for the Do not use sub-properties checkbox, this field is serialized in a JSON string (see below).
    - Send Pico Seconds – Requests that the OPC server provide timestamps with the highest possible resolution. The highest supported resolution is 10 picoseconds, but this request can be filled only for devices with a very high clock resolution.  
        
      By default, picosecond timestamps are stored as a sub-property of data item called sourcePicoseconds. If, however, there is a checkmark for the Do not use sub-properties checkbox, this field is serialized in a JSON string (see below).
    - Use OPC Sequence – Every C-DEngine “thing” has a unique sequence number, which increases every time a new thing is created. The sequence number survives system reboots. The OPC sequence, on the other hand, gets restarted every time the OPC UA server is restarted. With a checkmark in this field, the OPC sequence is used for the thing sequence.
    - Do not write array properties – Array properties are, by default, nested as sub-properties with the index number as the sub-property name. To reduce system resource usage, check this item.
    - Send Server Timestamp – Requests the OPC server to send a server-side timestamp (ordinarily only the client-side timestamp is recorded).  
        
      By default, server-side timestamps are stored as a sub-property called serverTimestamp. If, however, there is a checkmark for the Do not use sub-properties checkbox, this field is serialized in a JSON string (see below).
    - Send Sequence Number – Requests the OPC server to send the OPC sequence number with whatever data items are requested.
    - Do not use sub-properties – When this field is checked, the format of the stored value changes to JSON. The JSON will also store any of the following values that are also requested: statusCode, serverTimestamp, serverPicoseconds, sourcePicoseconds.  
        
      Example: When all of the possible other fields are requested, the JSON string would be as follows: {"value":123,"statusCode":0,"serverTimestamp": "2017-02-13T06:51:51.4262223Z","serverPicoseconds":1234,"sourcePicoseconds":2345}.

## Browsing and logging Group

The items in this group set the usage of tag names and whether data logging is enabled or not.

  
**Figure 4.3. The Browsing and logging Group.**

* + - Do Not Use Parent Path – Exclude OPC parent hierarchy in tag names. Use just the individual item name for the property name where the value is stored (default is OFF).
    - Show Tags in Properties — Can be turned off for performance reasons if the user interface is not used to verify which properties are subscribed. (default is ON).
    - Enable Data Logging – When any tag value changes, stores details in the file **opcclientdata.log**. This file contains details about every change to OPC tag values in a single location (default is OFF).

Note: Enable data logging only for diagnostic purposes, because logging may cause significant performance delays. No cleanup is performed on log files. In addition, all clients write to a single log file.

# Appendix A: AXOOM Gate Navigation Icons

Here is a summary of the AXOOM Gate navigation icons:

| Icon | Name | Comments |
| --- | --- | --- |
|  | Home | Click to navigate to home page. |
|  | Back | Use instead of the browser’s built-in back button. |
|  | Refresh | Use instead of the browser’s built-in refresh button. |
|  | Properties | Click to view properties. |
|  | Trash Can | Delete an item. |
|  | Up arrow | Closes a group of controls. |
|  | Down arrow | Opens a group of controls. |
|  | Left arrow | Decreases the width of browser space used to display property groups. |
|  | Right arrow | Increases the width of browser space used to display property groups. |

# Appendix B: Error and Logging Messages for plugin OPC UA Client

| **ID** | **DebugLevel** | **EngineName** | **UserMessage** |
| --- | --- | --- | --- |
| 78008 | OFF | MyBaseThing.EngineName | OPC Trace: {e.TraceMask:X04} - {output} |
| 78008 | OFF | MyBaseThing.EngineName | OPC Client Library Trace: Error processing event |
| 78201 | OFF | OPC | [{MyOPCServer.GetLogAddress()}] Internal Error - monitored item {previousMonitoredItem.StartNodeId} replaced with {TagRef}. Change will not take effect! |
| 78201 | ESSENTIALS | OPC | [{MyOPCServer.GetLogAddress()}] Internal Error: Monitored item not found after applying changes {TagRef}. Actual values: Sampling {MyMonitoredItem.Status.SamplingInterval}, Queue {MyMonitoredItem.Status.QueueSize} |
| 78201 | VERBOSE | OPC | [{MyOPCServer.GetLogAddress()}] Added monitored item {TagRef}. Actual values: Sampling {MyMonitoredItem.Status.SamplingInterval}, Queue {MyMonitoredItem.Status.QueueSize} |
| 78201 | FULLVERBOSE | OPC | [{0}] Error adding monitored item {1} |
| 78202 |  | OpcChangeValue | [{0}] Error Change Value: {1} |
| 78201 |  | GetOPCValueFromCDEValue | Error Change Value : |
| 78301 | VERBOSE | TheOPCTag | [{MyOPCServer.GetLogAddress()}] Error processing source monitored item notification. |
| 78301 | OFF | TheOPCTag | [{MyOPCServer.GetLogAddress()}] Server Queue overflow reported for monitored item: {TagRef}, {pValue.SourceTimestamp} |
| 78301 | VERBOSE | TheOPCTag | [{opcServer.GetLogAddress()}] Error converting source timestamp. Attempting to use ServerTimestamp. |
| 78301 | VERBOSE | TheOPCTag | [{opcServer.GetLogAddress()}] Error converting server timestamp. Using MinValue. |
| 78301 | VERBOSE | TheOPCTag | [{opcServer.GetLogAddress()}] No source timestamp for: { property.Name}, {pValue.SourceTimestamp} |
| 78301 | ESSENTIALS | TheOPCTag | [{opcServer.GetLogAddress()}] Internal error validating timestamp for minvalue: {property.Name}, {pValue.SourceTimestamp} |
| 78301 | OFF | TheOPCTag | [{opcServer.GetLogAddress()}] Unable to log data to file after retries: { property.Name}, {pValue.SourceTimestamp} |
| 78301 | ESSENTIALS | TheOPCTag | [{opcServer.GetLogAddress()}] Internal error for: { property?.Name}, {pValue.SourceTimestamp} |
| 78301 |  | TheOPCTag | [{0}] Error Reading Value : {1} |
| 78302 |  | MyBaseThing.EngineName | [{0}] Error Writing Value : {1} |
| 78401 | FULLVERBOSE | MyBaseThing.EngineName | [{0}] Method called concurrently |
| 78402 | FULLVERBOSE | MyBaseThing.EngineName | [{0}] Error processing method response for OPC Server |
| 78403 | VERBOSE | MyBaseThing.EngineName | [{0}] Method Call failed |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] Error adding tag {tag.ToString()}: {error} |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] Adding tag {tag.ToString()}: {error} |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] Initializing. |
| 78102 | VERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] {response} |
| 78102 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] {response} |
| 78102 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] {response} |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] MonitoredAllTagsInServer for {hostThingMID} |
| 78102 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] Error updating monitored tag |
| 78102 | ESSENTIALS | MyBaseThing.EngineName | [{GetLogAddress()}] Updated monitored tags |
| 78102 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] Internal error updating monitored tags |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] Initialized. |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] Monitored tag {pTag.ToString()}: {error} |
| 78102 | VERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] Error Monitoring tag {pTag.ToString()}: {error} |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] Monitored tag updated {pTag.ToString()}: {error} |
| 78102 | VERBOSE | MyBaseThing.EngineName | [{0}] Updated subscription. |
| 78102 | ESSENTIALS | MyBaseThing.EngineName | [{0}] Internal error: unexpected number of subscriptions found {1}. Expected 1 subscription. |
| 78101 | VERBOSE | MyBaseThing.EngineName | [{0}] Connecting to server. |
| 78102 | VERBOSE | MyBaseThing.EngineName | [{0}] Adjusting KeepAliveInterval to {1} instead of configured {2} due to server adjustment |
| 78001 | FULLVERBOSE | MyBaseThing.EngineName | [{0}] Connected to server. Subscribing. |
| 78002 | VERBOSE | MyBaseThing.EngineName | [{0}] Connected to server. Subscriptions Active. |
| 78103 | ESSENTIALS | MyBaseThing.EngineName | [{0}] Failed to connect to server: Failed to find root node. |
| 78104 | OFF | MyBaseThing.EngineName | [{0}] Failed to connect to server. Failed to create session. |
| 78003 | ESSENTIALS | MyBaseThing.EngineName | [{0}] Connect Failed for server |
| 78104 | EVERYTHING | MyBaseThing.EngineName | [{0}] Session closing for server |
| 78105 | VERBOSE | MyBaseThing.EngineName | [{0}] Session user identify not renewed for server |
| 78107 | ESSENTIALS | MyBaseThing.EngineName | [{1}] Create TagSubscriptions failed for {0} |
| 78108 | ESSENTIALS | MyBaseThing.EngineName | [{0}] Create TagSubscriptions failed |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | [{GetLogAddress()}] UnmonitorAllTagsInServer for {hostThingMID} |
| 78109 | ESSENTIALS | MyBaseThing.EngineName | [{0}] Property Subscription failed for property {1} |
| 78109 | OFF | MyBaseThing.EngineName | [{0}] Property Subscription failed for one or more tags |
| 78110 | OFF | MyBaseThing.EngineName | [{0}] Unexpected exception from server. Deleting subscription on disconnect |
| 78004 | OFF | MyBaseThing.EngineName | [{0}] Disconnected from server |
| 78005 | OFF | MyBaseThing.EngineName | [{0}] Error disconnecting from server |
| 78006 | VERBOSE | MyBaseThing.EngineName | [{0}] Received keep alive for old session. |
| 78006 | VERBOSE | MyBaseThing.EngineName | [{0}] Communication Error for OPC Server. Disconnecting. |
| 78007 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] {fullMessage} |
| 78007 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] {fullMessage} |
| 78111 | OFF | MyBaseThing.EngineName | Stopping reconnect due to keepalive |
| 78111 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] {message} |
| 78111 | OFF | MyBaseThing.EngineName | Ignoring reconnect notification for previous handler |
| 78008 | OFF | MyBaseThing.EngineName | [{GetLogAddress()}] Failed to Reconnect: {e?.Exception} |
| 78009 | OFF | MyBaseThing.EngineName | [{0}] Reconnected |
| 78112 | OFF | MyBaseThing.EngineName | [{0}] Reconnect Failed - internal error |
| 78113 | OFF | MyBaseThing.EngineName | [{0}] Certificate Verification failed |
| 78114 | OFF | MyBaseThing.EngineName | [{0}] Browsing Failed |
| 78116 | ESSENTIALS | MyBaseThing.EngineName | [{0}] Error browsing Server node. Continuing. |
| 78117 | OFF | MyBaseThing.EngineName | [{0}] ReadAttributes Failed |
| 78118 | ESSENTIALS | MyBaseThing.EngineName | [{0}] Duplicate Node Found: {1} |
| 78119 | OFF | MyBaseThing.EngineName | [{0}] Browser Exception |
| 78120 | OFF | MyBaseThing.EngineName | [{0}] ReadAttributes Failed |
| 78121 |  | MyBaseThing.EngineName | [{0}] Browser Exception |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | Tag init for tag {m\_Tag} |
| 78102 | FULLVERBOSE | MyBaseThing.EngineName | Monitoring tag due to DontMonitor property change {m\_Tag} |