



User Manual OPC UA Server for Altus Controllers

MU214609 Rev. B

July 23, 2026

No part of this document may be copied or reproduced in any form without the prior written consent of Altus Sistemas de Automação S.A. who reserves the right to carry out alterations without prior notice.

According to current legislation in Brazil, the Consumer Defense Code, we are giving the following information to clients who use our products, regarding personal safety and premises.

The industrial automation equipment, manufactured by Altus, is strong and reliable due to the stringent quality control it is subjected to. However, any electronic industrial control equipment (programmable controllers, numerical commands, etc.) can damage machines or processes controlled by them when there are defective components and/or when a programming or installation error occurs. This can even put human lives at risk. The user should consider the possible consequences of the defects and should provide additional external installations for safety reasons. This concern is higher when in initial commissioning and testing.

The equipment manufactured by Altus does not directly expose the environment to hazards, since they do not issue any kind of pollutant during their use. However, concerning the disposal of equipment, it is important to point out that built-in electronics may contain materials which are harmful to nature when improperly discarded. Therefore, it is recommended that whenever discarding this type of product, it should be forwarded to recycling plants, which guarantee proper waste management.

It is essential to read and understand the product documentation, such as manuals and technical characteristics before its installation or use. The examples and figures presented in this document are solely for illustrative purposes. Due to possible upgrades and improvements that the products may present, Altus assumes no responsibility for the use of these examples and figures in real applications. They should only be used to assist user training and improve experience with the products and their features.

Altus warrants its equipment as described in General Conditions of Supply, attached to the commercial proposals.

Altus guarantees that their equipment works in accordance with the clear instructions contained in their manuals and/or technical characteristics, not guaranteeing the success of any particular type of application of the equipment.

Altus does not acknowledge any other guarantee, express or implied, mainly when end customers are dealing with third-party suppliers. The requests for additional information about the supply, equipment features and/or any other Altus services must be made in writing. Altus is not responsible for supplying information about its equipment without formal request. These products can use EtherCAT® technology (www.ethercat.org).

COPYRIGHTS

Nexto, Mastertool, Grano and WebPLC are the registered trademarks of Altus Sistemas de Automação S.A.

Windows, Windows NT and Windows Vista are registered trademarks of Microsoft Corporation.

OPEN SOURCE SOFTWARE NOTICE

To obtain the source code under GPL, LGPL, MPL and other open source licenses, that is contained in this product, please contact opensource@altus.com.br. In addition to the source code, all referred license terms, warranty disclaimers and copyright notices may be disclosed under request.

List of Contents

1. INTRODUCTION	1
Documents Related to this Manual	1
Technical Support	1
Warning Messages Used in this Manual	2
2. TECHNICAL DESCRIPTION	3
Altus Controllers that Support OPC UA Server	3
OPC UA Profile	3
OPC UA Server Features.....	3
Endpoint URL	3
Maximum Number of Sessions	3
Maximum Number of Subscriptions	3
Maximum Number of Monitored Items	3
Values of Operational Limits	3
Revised Parameters for Subscriptions and Monitored Items	4
3. OPC UA COMMUNICATION CONCEPTS	5
Sessions, Servers, and Clients.....	5
Endpoint URL.....	5
Subscriptions, Notifications, Monitored Items, and Nodes.....	6
Creating a Subscription and its Parameters	6
PublishingInterval	7
MaxKeepAliveCount	9
LifeTimeCount.....	9
MaxNotificationsPerPublish	10
PublishingEnabled.....	10
Priority.....	10
Sampling Process in the Server	11
SamplingInterval	11
Filter.....	12
QueueSize	12
DiscardOldest.....	12
Browsing the Server for Creating a Subscription	13
Read Services	15
Write Services	16
Other Services.....	16
Operational Limits	16
MaxMonitoredItemsPerCall.....	16
MaxNodesPerRead.....	16
MaxNodesPerWrite.....	17
MaxNodesPerBrowse.....	17
MaxNodesPerTranslateBrowsePathsToNodeIds	17
OPC UA Security.....	17
Sign and Encrypt	17
Authentication.....	17
Certificates	17
Validity of the Certificate.....	18
Encryption Algorithms.....	18
4. BASIC CONFIGURATION	19

Recommended Sequence of Steps for Configuration	19
First Step - Create the Variables for OPC UA Communication.....	19
Second Step - Insert the Objects "Communication Manager" and "OPC UA Server"	19
Third Step - Selecting the OPC UA Variables and their Access Rights	22
Fourth Step - Download the Project	24
Add More Variables for OPC UA Communication Later.....	24
5. SECURITY CONFIGURATION.....	26
Anonymous Login (No Authentication).....	28
Authentication	28
Creating a New Device User and its Password.....	28
Adjusting the OPC UA Settings for Authentication	30
Sign and Encrypt.....	31
Configuration Options in Altus Servers	31
Configuration Options in OPC UA Clients.....	32
Altus Server Configurations and Allowed Configurations in OPC UA Clients	32
Creating or Importing Certificates	33
Negotiating the Certificate with the OPC UA Client	35
6. RECOMMENDATIONS FOR OPC UA CLIENTS CONFIGURATION	36
Configure only Supported Values of Parameters PublishingInterval and SamplingInterval.....	36
Observe the Operational Limits.....	36
Avoid Small Values for SamplingInterval and PublishingInterval	37
Split a Session in Several Subscriptions	37
Split the Communication in Several Sessions	37
7. DIAGNOSTICS.....	38
Server Diagnostics	38
Operational Limits	39
8. PERFORMANCE ANALYSIS	40
General Test Conditions	40
Test Scenarios	40

1. Introduction

This manual addresses several aspects related to OPC UA communication with Altus controllers that support the OPC UA Server feature:

- Definition of the OPC UA profile and features supported by the Altus controllers;
- Introduction to OPC UA protocol concepts, considering only the profile supported by the Altus controllers;
- Main configuration parameters negotiated with OPC UA clients and their effects on the behavior and performance of communication;
- Operational limits and interoperability issues with OPC UA clients;
- Configuration of OPC UA server in Altus controllers using the Mastertool programmer;
- Performance analysis of OPC UA communication, considering CPU processing consumption and Ethernet communication bandwidth.

Documents Related to this Manual

In order to obtain additional information regarding the Nexto Series, other documents (manuals and technical features) besides this one, may be accessed. These documents are available in its last version on the site <https://www.altusautomation.com>.

Each product has a document designed by Technical Features (CE), where the product features are described. Furthermore, the product may have Utilization Manuals (the manuals codes are listed in the CE).

The following documents are recommended as additional sources of information:

Code	Description	Language
CE114000	Nexto Series – Technical Characteristics	English
CT114000	Série Nexto – Características Técnicas	Portuguese
MU214600	Nexto Series User Manual	English
MU214000	Manual de Utilização Série Nexto	Portuguese
MU299611	Mastertool X User Manual	English
MU299049	Manual de Utilização Mastertool X	Portuguese
MP399609	IEC 61131 Programming Manual	English
MP399048	Manual de Programação IEC 61131	Portuguese

Table 1-1 -Related Documents

Technical Support

To contact Altus Technical Support, send an email to support@altusautomation.com or visit our website at www.altusautomation.com/support. If the equipment is already installed, please have the following information available when requesting assistance:

- Model of the equipment used and the installed system configuration
- Product serial number
- Equipment revision and executive software version, indicated on the label attached to the side of the product
- CPU operating mode information, obtained using Mastertool
- Application software content, obtained using Mastertool
- Mastertool version used

Warning Messages Used in this Manual

In this manual, the warning messages have the following formats and meanings:

DANGER:

Report potential causes, which if not observed, lead to damage to physical integrity and health, property, environment and loss of production.

CAUTION:

Report configuration, application, and installation details that must be followed to avoid conditions that could lead to system failure and its related consequences.

ATTENTION:

Indicate important configuration, application or installation details to obtain the maximum operational performance of the system.

2. Technical Description

This chapter presents the technical features of OPC UA Server for Altus controllers.

Altus Controllers that Support OPC UA Server

All the CPUs of Nexto Series support the OPC UA Server feature.

OPC UA Profile

The Altus controllers support the **Micro Embedded Device Server Profile**, according to part 7 of the OPC UA specification. They support the following features:

- Browsing of data types and variables;
- Standard read/write services;
- Notifications for value changes: subscription and monitored items services.

OPC UA Server Features

This section defines several features of the OPC UA server.

ATTENTION:

The chapter **OPC UA Communication Concepts** explains the meaning of these features for newcomers to OPC UA technology.

Endpoint URL

The endpoint URL, used by the client for connecting to Altus servers, must have the following format:

opc.tcp://<IP address of CPU>:4840

The field <IP address of CPU> indicates the IP address of the Ethernet port of the CPU that must be used for the OPC UA communication.

The TCP port used for the connection is 4840.

Maximum Number of Sessions

The maximum number of simultaneous sessions that can be established between one or more clients and the server is 50.

Maximum Number of Subscriptions

The number of subscriptions, for each session and for the server as a whole, is not limited. In practice, the number of subscriptions is limited by the CPU performance (memory and CPU speed).

Maximum Number of Monitored Items

The number of monitored items, for each subscription, session or for the server as a whole, is not limited. In practice, the number of monitored items is limited by the CPU performance (memory and CPU speed).

Values of Operational Limits

The following operational limits are defined for Altus servers:

- MaxMonitoredItemsPerCall = 1000
- MaxNodesPerRead = 1000
- MaxNodesPerWrite = 1000

- MaxNodesPerBrowse = 1000
- MaxNodesPerTranslateBrowsePathsToNodeIds = 100

ATTENTION:

Sometimes interoperability issues occur when a client exceeds one of these operational limits. The clients can read these limits from the server and respect them. Unfortunately, some clients do not read or do not respect these limits. Most of these clients at least inform about service failures due to exceeded limits, using a diagnostic or a log. In this case, some clients allow to configure parameters for correcting the problem.

Revised Parameters for Subscriptions and Monitored Items

For creating subscriptions and monitored items, the OPC UA client sends service requests to the server (CreateSubscriptionRequest, CreateMonitoredItemsRequest). These service requests contain parameters for the subscription and monitored items.

After this, the server sends service responses to the client (CreateSubscriptionResponse, CreateMonitoredItemsResponse), and these service responses contain **revised parameters**.

In many situations, the revised parameters in the response are equal to the requested parameters. However, sometimes the server may not support specific values for some parameters. In this case, some revised parameters will be different from the corresponding requested parameters. This difference normally does not cause an error in the client, and the communication is established using the revised parameters.

ATTENTION:

Some clients do not inform the user about revision of parameters. So the user may not become aware of the parameter value that is really in use. Other clients may inform the revised parameter through a diagnostic or log.

The following subsections describe the supported values of some parameters in Altus OPC UA servers, and how the server revise parameters for subscriptions and monitored items. With this information, the user can avoid to configure unsupported values for these parameters.

Revision of Parameter PublishingInterval for Subscriptions

Altus servers only accept a PublishingInterval value which is an integer multiple of 100 ms.

For instance, if the client requested PublishingInterval = 850 ms, then the server will revise the value rounding it up to the next multiple of 100 ms, that is, 900 ms.

Revision of Parameter SamplingInterval for Monitored Items

Altus servers only accept the following values for SamplingInterval:

- 100 ms
- 300 ms
- 500 ms
- 1000 ms
- 2500 ms
- 5000 ms

Different values will be revised to the next bigger value (for instance, 600 ms will be revised to 1000 ms). Values bigger than 5000 ms will be revised to 5000 ms.

3. OPC UA Communication Concepts

This chapter provides an introduction to OPC UA communication concepts with an Altus controller. Therefore, this introduction is limited to the profile supported by an Altus controller (see section **OPC UA Profile**).

The user, especially a newcomer to OPC UA technology, is recommended to read this introduction for a better understanding of the further chapters.

In some situations, the following third-party OPC UA clients will be mentioned for providing examples of OPC UA communication:

- UaExpert®: refers to demo client UaExpert™ version 1.7.0 526 from Unified Automation GmbH.
- Kepware®: refers to commercial client KEPServerEX® V6.14.263.0 from Kepware®.

Sessions, Servers, and Clients

Sessions are defined as logical connections between clients and servers. Servers may limit the number of simultaneous sessions based on resource availability, licensing restrictions, or other constraints.

A TCP connection is allocated for each session. Altus controllers, as servers, accept connections for OPC UA sessions in the TCP port 4840.

If multiple clients want to connect to a single server, each one of these clients must create at least one session with this server.

In small applications (few variables), normally each client creates a single session with each server. In bigger applications (big number of variables), the client may create several sessions with the same server. In most clients, each session allocates a thread in the operating system (e.g.: Windows). Creating several sessions may optimize the communication performance on the client side.

For instance, in client Kepware®, each channel corresponds to a session (add multiple channels pointing to the same server for having multiple sessions for the same server).

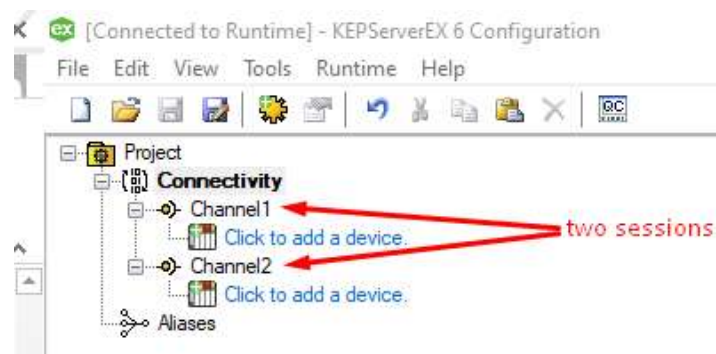


Figure 3-1. Example of multiple sessions in client Kepware®

Endpoint URL

The Endpoint URL is the address used by the client for finding the server in the network. It is necessary for creating a session between the client and the server.

For instance, the Altus OPC UA server requires the following format:

opc.tcp://<IP address of CPU>:4840

The field <IP address of CPU> indicates the IP address of the Ethernet port of the CPU that must be used for the OPC UA communication. The TCP port used for the connection is 4840.

The following figure shows an example of the configuration of the endpoint URL for a Session in Kepware®. For opening the window with this property editor, right-click over the Session (Channel in Kepware®) and select Properties.



Figure 3-2. Example of Endpoint URL in client Kepware®

Subscriptions, Notifications, Monitored Items, and Nodes

A **subscription** is an endpoint in the server that publishes notifications to clients.

A **notification** is a message with a set of monitored items. The number of monitored items in a notification is variable (from zero to all the nodes configured in the subscription). The server uses a service called PublishResponse for transmitting notifications with monitored items.

A **monitored item** corresponds to a node that has changed its value. It is composed of the node value and the timestamp informing the moment when the value change was detected.

A **node** corresponds to a variable in the server.

The main ideas of a subscription are the following:

- Only the variables (nodes) which changed value are transmitted from the server to the client as monitored items (except right after initialization of the communication, when all nodes are transmitted as monitored items). This saves bandwidth in the network and processing power in the server and client.
- The server must detect those variables (nodes) which changed value, for including them as monitored items in the next notification. This process, and its parameters, are described in the section **Sampling Process in the Server**.

Creating a Subscription and its Parameters

A subscription is created on the client side. For each client session, it is possible to create many subscriptions.

For instance, in client Kepware®, each device corresponds to a subscription (add multiple devices below the same channel (session) for having multiple subscriptions in the same session). The following figure shows two subscriptions (Device1 and Device2) below the same session (Channel1).

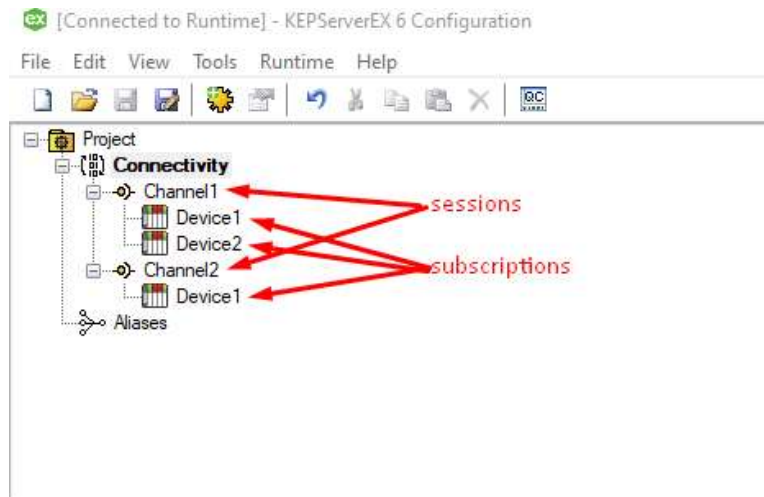


Figure 3-3. Example of multiple subscriptions in the same session in client Kepware®

For creating a subscription on the client side, the user must:

- Select the nodes that compose the subscription (in other words, a set of variables in the server). This process is described in the section **Browsing the Server for Creating a Subscription**.
- Set the following subscription parameters:
 - PublishingInterval
 - MaxKeepAliveCount
 - LifeTimeCount
 - MaxNotificationsPerPublish
 - PublishingEnabled
 - Priority

The subscription parameters are described in the following subsections.

ATTENTION:

The previous parameter names are those used in the OPC UA specification. Unfortunately, some clients may use different names.

Furthermore, some clients do not allow the user to configure all these parameters. Some clients allow the user to configure a subset of these parameters and set the other parameters internally (normally with constant values).

In case of doubts, please check the client documentation or contact the support team of the client manufacturer.

PublishingInterval

The PublishingInterval is the minimum interval between two PublishResponse services transmitted by the server to the client with notifications for the same subscription. This limits the bandwidth consumption in the network and CPU consumption in the server and client.

Right after creating a subscription, the client transmits one (or more) PublishRequest services for this subscription. This authorizes the server to transmit one (or more) PublishResponses services containing notification messages.

After receiving a PublishResponse from the server, the client typically transmits a further PublishRequest as soon as possible, for authorizing the server to transmit the next PublishResponse.

The following notes about PublishRequest services transmitted by the client are important:

- This service has two objectives. First, it acknowledges a notification received from the server in a previous PublishResponse service for the subscription. Second, it authorizes the server to send the next notification using a PublishResponse.

- Normally, the delay between a PublishResponse and the next PublishRequest transmitted by the client is very low (a few milliseconds), except when the network has high latency or when the client has low processing power.
- When the server receives a PublishRequest from the client, it can send a PublishResponse for any of the subscriptions created inside the current session. If there are two or more subscriptions in a session waiting for a PublishRequest, only one of them will use the received PublishRequest for authorizing a PublishResponse.
- Some clients may queue several PublishRequest services. This helps for not delaying PublishResponse services, especially in networks with higher latency. For instance, Kepware® normally queues three PublishRequest services for each subscription, right after creating the subscription and their monitored items. According to the OPC UA specification, a server must accept more queued PublishRequest services than the number of created subscriptions per session.

The observed interval between two PublishResponse services for the same subscription can be bigger than the PublishingInterval. If during this interval none of the subscription variables (nodes) changed value, the server will not transmit a PublishResponse.

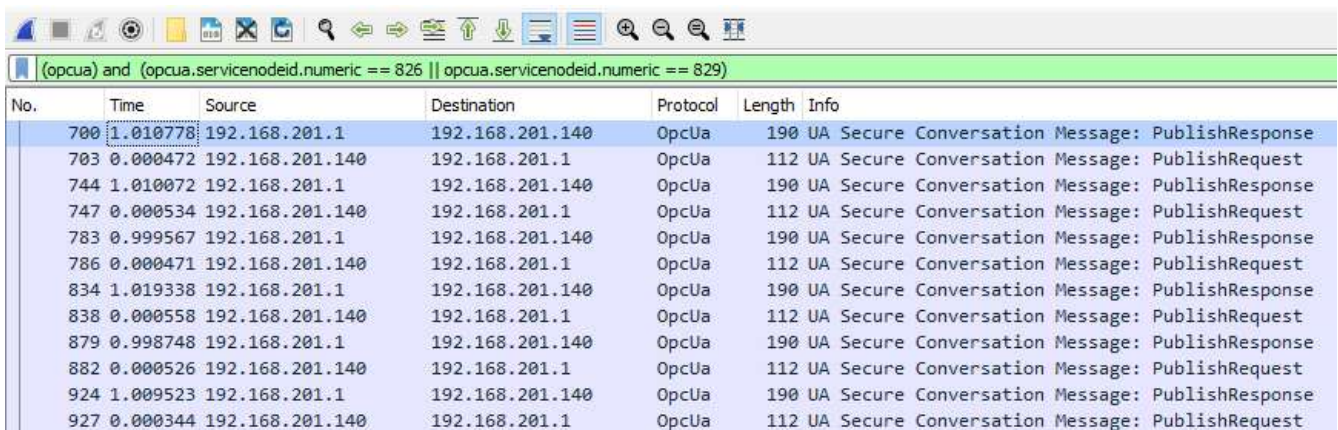
Consider the following examples:

1. Subscription with PublishingInterval = 1000 ms, where the last PublishResponse was transmitted at $t = 0$ ms. If the server detects the next variable change only at $t = 3400$ ms, then PublishResponse services will not be transmitted at $t = 1000, 2000,$ and 3000 ms. The next PublishResponse will only be transmitted after $t = 3400$ ms.
2. Subscription contains 100 variables, but only two of them changed value. The next PublishResponse service will contain a notification with only two monitored items.

Example of OPC UA communication with PublishRequest and PublishResponse services

The following figure shows a fragment of OPC UA communication captured with the Wireshark® protocol analyzer, filtered to show only the services PublishRequest and PublishResponse. The "Time" column shows the interval between successive packets in seconds.

In this example, PublishingInterval was configured as 1000 ms, and the subscription variables change every 100 ms (much faster than PublishingInterval).



No.	Time	Source	Destination	Protocol	Length	Info
700	1.010778	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse
703	0.000472	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest
744	1.010072	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse
747	0.000534	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest
783	0.999567	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse
786	0.000471	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest
834	1.019338	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse
838	0.000558	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest
879	0.998748	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse
882	0.000526	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest
924	1.009523	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse
927	0.000344	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest

Figure 3-4. Example of OPC UA communication with variables varying faster than PublishingInterval

Note that right after the PublishResponse in packet No. 700, a new PublishRequest (packet No. 703) was sent for authorizing the next PublishResponse. The delay between PublishResponse at PublishRequest is lower than 1 ms.

However, the next PublishResponse (packet No. 744) was sent only one second later (the PublishingInterval). This sequence repeats in the next packets.

ATTENTION:

See section **Revision of Parameter PublishingInterval for Subscriptions**.

Example of this parameter in Kepware®:

- Name and location of parameter = Subscription / Publishing Interval (ms)
- Default value = 1000 ms

MaxKeepAliveCount

As mentioned before, a PublishResponse will only be transmitted by the server after at least one variable of the subscription changed value.

If all the variables of a subscription keep unchanged for some time, the server will transmit a keep-alive message. The keep-alive message is an empty PublishResponse that contains zero monitored items. The objective is to keep some traffic in the session so that the client can know that the server is alive.

The time with unchanged variables in the subscription before transmitting a keep-alive message is equal to:

$$\text{MaxKeepAliveCount} * \text{PublishingInterval}$$

Example of OPC UA communication with a keep-alive message

The following figure shows a fragment of OPC UA communication captured with the Wireshark® protocol analyzer, filtered to show only the services PublishRequest and PublishResponse. The "Time" column shows the interval between successive packets in seconds.

In this example, PublishingInterval was configured as 1000 ms and MaxKeepAliveCount as 5.

(opcua) and (opcua.servicenodeid.numeric == 826 opcua.servicenodeid.numeric == 829)							
No.	Time	Source	Destination	Protocol	Length	Info	
822	0.998956	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse	
825	0.000477	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest	
869	1.009594	192.168.201.1	192.168.201.140	OpcUa	190	UA Secure Conversation Message: PublishResponse	
872	0.000560	192.168.201.140	192.168.201.1	OpcUa	112	UA Secure Conversation Message: PublishRequest	
1114	6.048225	192.168.201.1	192.168.201.140	OpcUa	143	UA Secure Conversation Message: PublishResponse	
1117	0.000428	192.168.201.140	192.168.201.1	OpcUa	104	UA Secure Conversation Message: PublishRequest	
1310	5.051425	192.168.201.1	192.168.201.140	OpcUa	143	UA Secure Conversation Message: PublishResponse	
1313	0.000472	192.168.201.140	192.168.201.1	OpcUa	104	UA Secure Conversation Message: PublishRequest	
1504	5.023262	192.168.201.1	192.168.201.140	OpcUa	143	UA Secure Conversation Message: PublishResponse	
1507	0.000428	192.168.201.140	192.168.201.1	OpcUa	104	UA Secure Conversation Message: PublishRequest	
1694	5.045147	192.168.201.1	192.168.201.140	OpcUa	139	UA Secure Conversation Message: PublishResponse	
1697	0.000386	192.168.201.140	192.168.201.1	OpcUa	104	UA Secure Conversation Message: PublishRequest	

Figure 3-5. Example of OPC UA communication with variables frozen during some time

Initially, the subscription variables change every 100 ms (much faster than PublishingInterval). This situation stopped after packet No. 872 when all the subscription variables became frozen.

After this, the PublishResponse services became separated by approximately "MaxKeepAliveCount * PublishingInterval" (5 seconds). The first PublishResponse after variables became frozen may take one extra PublishingInterval (6 seconds).

Example of this parameter in Kepware®:

- Name and location of parameter = Connection / Keep-Alive Count
- Default value = 5

LifeTimeCount

If the server does not receive a PublishRequest from the client during a time longer than:

$$\text{LifetimeCount} * \text{PublishingInterval}$$

then the server must delete the subscription. From the server point-of-view, this means that a failure occurred in the client or in the network. In this case, the client will need to recreate the subscription and their monitored items later.

According to the OPC UA specification, the parameter `LifeTimeCount` must be at least three times bigger than `MaxKeepAliveCount`.

After creating a subscription and their monitored items, the client sends one or more `PublishRequest` services. This authorizes the server to send `PublishResponse` services.

When the server sends a `PublishResponse`, normally the client sends a new `PublishRequest` in a few milliseconds, for authorizing the server to send the next `PublishResponse`. The next `PublishResponse` will be transmitted by the server in the following conditions:

1. At least one subscription variable changed value, and the time `PublishingInterval` passed since the previous `PublishResponse` was transmitted.
2. The time "`MaxKeepAliveCount * PublishingInterval`" passed while none of the subscription variables changed value.

This explains why `LifetimeCount` must be bigger than `MaxKeepAliveCount`. Otherwise, the subscription could be deleted unnecessarily by the server when none of the subscription variables change value.

Example of this parameter in Kepware®:

- Name and location of parameter = `Connection / Lifetime Count`
- Default value = 60

MaxNotificationsPerPublish

Informs the maximum number of monitored items that can be transmitted in a `PublishResponse` service for this subscription.

The special value "0" means that no limit is imposed.

Example of this parameter in Kepware®:

- Name and location of parameter = `Subscription / Max. Notifications per Publish`
- Default value = 0

PublishingEnabled

Enables or disables the transmission of `PublishReponses` for this subscription.

If the subscription was created with this parameter equal to false, it can be modified to true later using the service `SetPublishingMode`. The service `SetPublishingMode` can also be used for disabling temporarily `PublishReponses` for a subscription.

Example of this parameter in Kepware®:

The parameter is not available for the user. `PublishResponses` are always enabled after the subscription and monitored items are created.

Priority

This is a byte value which means the priority of a subscription relative to other subscriptions of the same session. The bigger the value, the higher the priority.

`PublishReponses` for subscriptions with higher priority will be transmitted sooner.

Example of this parameter in Kepware®:

- Name and location of parameter = `Connection / Priority`
- Default value = Lowest (0)

Sampling Process in the Server

For each subscription, the server must detect which variables (nodes) changed values for transmitting them as monitored items in a notification message. This task is known as the sampling process on the server side. The sampling process must be executed periodically for all variables (nodes) that compose the subscription. It is executed internally in the server.

The OPC UA specification defines the following parameters for each monitored item:

- SamplingInterval
- Filter
- QueueSize
- DiscardOldest

The client must configure these parameters for each monitored item.

ATTENTION:

Most commercial clients do not allow the user to configure these parameters individually for each monitored item. Instead, most clients allow configuring common parameters that apply to all the monitored items that compose a subscription.

In addition, some clients do not allow configuring all these parameters. In these situations, please check the client documentation or contact the support team of the client manufacturer.

For instance, some clients set SamplingInterval of all monitored items equal to PublishingInterval of the subscription.

The following subsections describe each parameter configurable for a monitored item.

SamplingInterval

This is the period of execution of the sampling process. In other words, it defines the interval of the server task responsible for detecting value changes in the variables that compose the subscription, and for queueing the changed variables for transmission in the next PublishResponse service.

Normally the SamplingInterval must be smaller than or equal to the PublishingInterval of the subscription (described before).

Some commercial clients do not allow the user to configure the SamplingInterval, and set it equal to the PublishingInterval of the subscription.

The sum between SamplingInterval and PublishingInterval means the **worst-case response time** since the variable changed value until it is transmitted in the next PublishResponse service:

- SamplingInterval: worst-case time for discovering the value change;
- PublishingInterval: worst-case time for transmitting the value change using a PublishResponse service.

The value of SamplingInterval is an interval specified in milliseconds. In addition, two special values can be specified:

- 0: this means that the server must execute the sampling process as fast as it can;
- -1: this means that the server must assume SamplingInterval equal to PublishingInterval of the subscription.

ATTENTION:

See section **Revision of Parameter SamplingInterval for Monitored Items**.

ATTENTION:

The sampling process is an intensive CPU-consuming task. Therefore, avoid configuring low values for the `SamplingInterval` because this will increase CPU consumption.

A good recommendation is to set the `SamplingInterval` equal to the `PublishingInterval` of the subscription.

Furthermore, for reducing CPU consumption, do not configure unnecessarily low values for the `PublishingInterval`.

Example of this parameter in Kepware®:

- Name and location of parameter = Monitored Items / Sample Interval (ms)
- Default value = 500 ms

Filter

The Filter parameter is not evaluated in Altus OPC UA servers.

Filters can be useful, for instance, to define a deadband for a variable that corresponds to an analog input. An analog input normally has some noise that affects the least-significant bits. Therefore, the sampling process would detect changes in the analog input all the time (even if these changes are very small), and the variable would be transmitted very frequently in `PublishResponse` services. As a consequence, the network bandwidth and CPU consumption would increase unnecessarily.

Even if not supporting filters, it is possible to use a strategy in Altus OPC UA servers for circumventing this problem, considering the analog input example:

1. Create a function block that makes a deadband filter over the analog input and produces another variable derived from the analog input.
2. Do not map the analog input directly to the OPC UA server. Instead, map the variable derived from the analog input with deadband filtering.

QueueSize

Consider a situation where `SamplingInterval` is much lower than `PublishingInterval`. In this case, the sampling process can detect several changes in a variable during a `PublishingInterval`. For instance, if `SamplingInterval` = 100 ms and `PublishingInterval` = 1000 ms, the sampling process could detect up to 10 changes in the same variable during a single `PublishingInterval`.

The default value of the `QueueSize` parameter normally is 1. This means that only one of the detected changes will be transmitted in the next `PublishResponse` for the same variable. However, if `QueueSize` = N, then the next `PublishResponse` can transmit up to N changes for the same variable.

If the number of detected changes in a variable is bigger than `QueueSize`, it is necessary to discard some of the changes. The discarded changes will not be transmitted in the next `PublishResponse` service.

The discarded changes can be the oldest or the newest. This depends on the configuration of the parameter `DiscardOldest`, described in the next subsection.

Example of this parameter in Kepware®:

- Name and location of parameter = Monitored Items / Queue Size
- Default value = 1

DiscardOldest

See the previous section that describes the parameter `QueueSize`.

If the sampling process needs to discard changes, it must decide which will be discarded: the oldest or the newest. If this parameter is true or enabled, then the oldest changes will be discarded.

Example of this parameter in Kepware®:

- Name and location of parameter = Monitored Items / Discard Oldest
- Default value = Enable

Browsing the Server for Creating a Subscription

Besides defining parameters for a subscription and their monitored items (described in previous sections), it is necessary to select the variables (nodes) that compose the subscription. The OPC UA protocol provides services like Browse and BrowseNext that are used for navigating through the data structures of the server and importing the desired variables.

Not all data structures of the server can be found during the browsing process. Each server has methods for selecting a subset of the data structures that can be accessed by an OPC UA client. Furthermore, it is also possible to restrict the access of some data structures (read-write, read-only, write-only, none). For Altus OPC UA servers, this is described in the chapter **Basic Configuration**.

The following figures show an example of the browsing process using the Kepware® client.

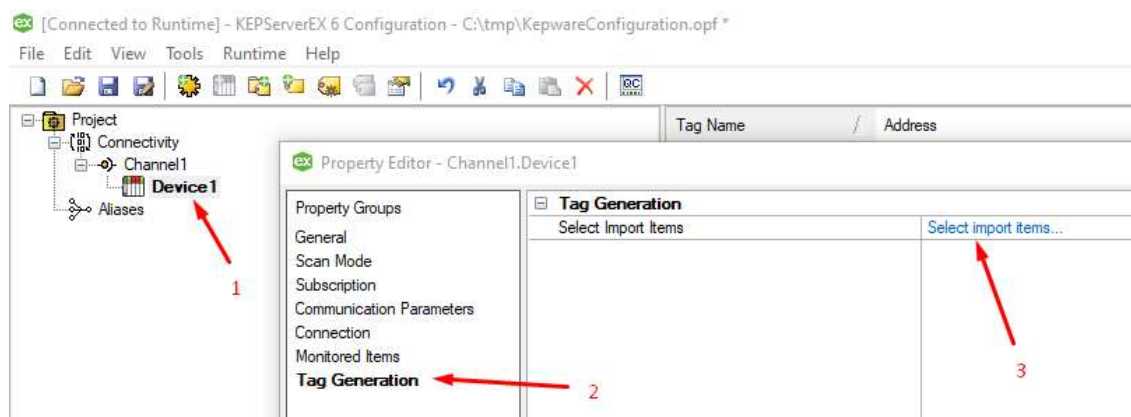


Figure 3-6. Starting the browse process with Kepware® client

First of all, it is necessary to create a Session (called Channel in Kepware®) and a Subscription (called Device in Kepware®) below the session, and adjust their parameters (these parameters were described in previous sections).

After this:

1. Right-click over Device and select Properties for opening the Property Editor;
2. Select Tag Generation;
3. Click over "Select import items..."

This starts the browsing process (communication with the server using Browse services), opening a window like the following.

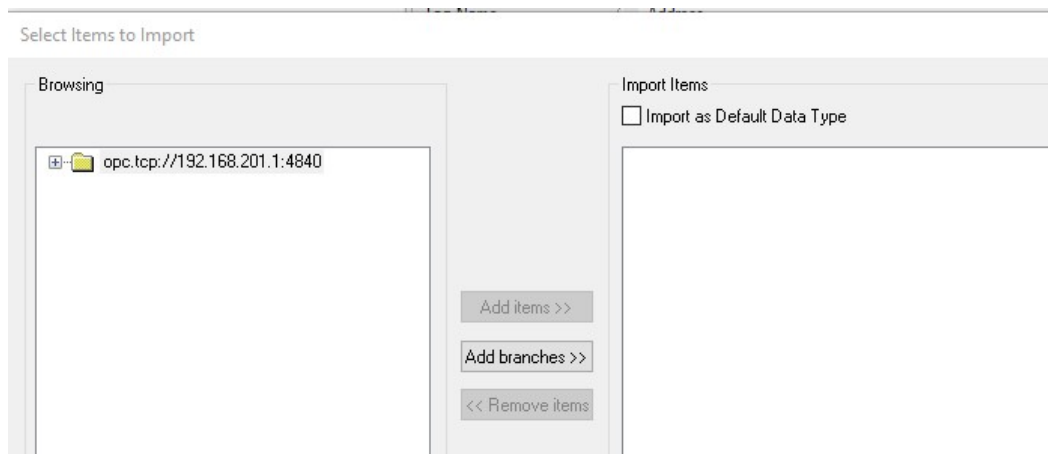


Figure 3-7. Initial state of the browse tree

After this, the user can expand the tree and search for the desired variables for the subscription. In this example, the user may know that all the desired variables are inside a data structure called OPC_UA_DATA. After finding this data structure, the user can click over the button "Add branches >>" for importing all the variables inside this data structure.

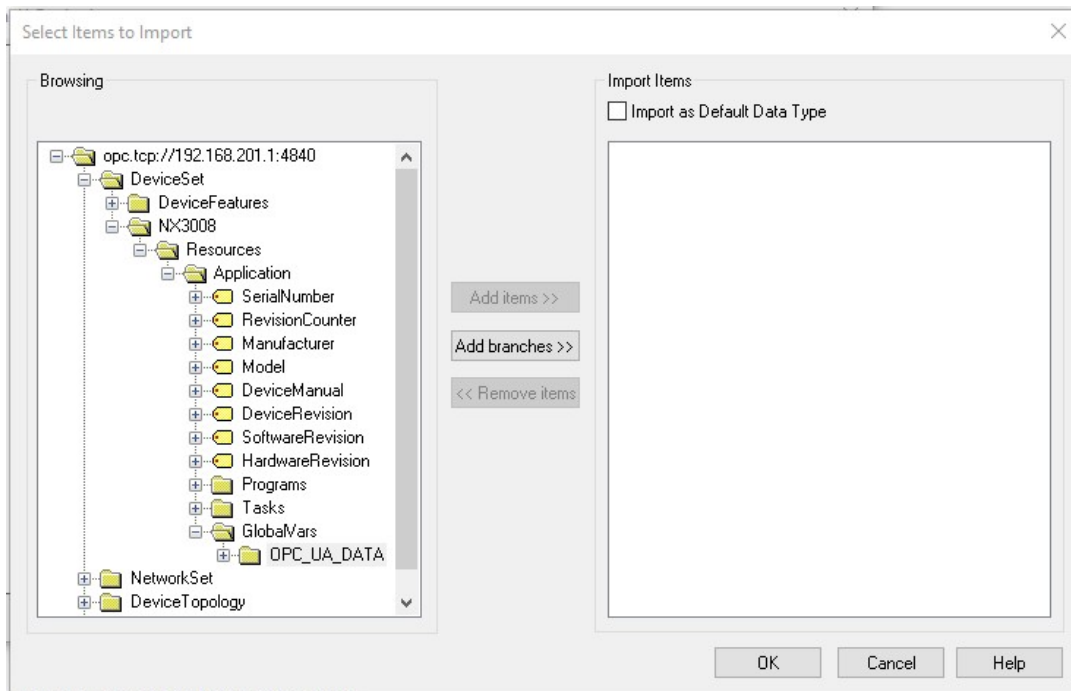


Figure 3-8. Browsing the tree

Finally click the OK button, and Apply / OK buttons in the following windows that will open.

At the end, the nodes for the subscription were imported and saved in the configuration file of Kepware®:

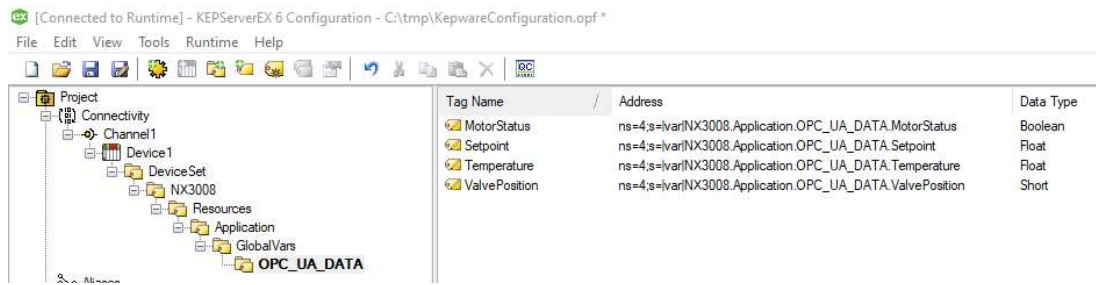


Figure 3-9. Nodes imported for the subscription

Read Services

In most situations, the variables of the server are read by a client using subscriptions with monitored items transmitted using PublishResponse services. This is the more optimized method, considering network bandwidth consumption and CPU consumption on the server and client sides. In terms of bandwidth, this method has the following advantages:

- Only the variables that changed value are transmitted using PublishResponse services;
- In the PublishResponse service, the variable identification uses a 32-bit identifier instead of a long string that can have hundreds of characters. This 32-bit identifier is negotiated between the client and server when the monitored items are created.

However, the OPC UA protocol also provides a Read service. This method is much less optimized than the subscription method, due to the following reasons:

- The client must transmit a ReadRequest service periodically specifying all the variables that it wants to read.
- In response, the server must transmit a ReadResponse service with the value and timestamp of all the variables specified in the ReadRequest service.
- The ReadRequest service contains a long string for identifying each variable. This string can have hundreds of characters.

Some clients allow the user to choose the method for reading variables, among using subscriptions or read services. For instance, the Kepware® client has the parameter Subscription / Update Mode, with two possible values:

- Exception (default value): the subscription method is selected;
- Poll: Read services are used. In this case, Kepware® uses the parameter PublishingInterval as the polling interval for the read services. This method is not recommended due to the disadvantages mentioned before.

Nevertheless, even if the user decides to use the subscription method, in some situations the client needs to use Read services:

- During the subscription creation, after using Browse services for importing the variables (see section **Browsing the Server for Creating a Subscription**). In this case, Read services are used for reading attributes of variables (like DataType and AccessLevel).
- After a communication failure using the subscription method, the variables that changed value during the failure are transmitted by the server using a PublishResponse service. However, the variables that have not changed value during the failure normally are updated after a ReadRequest service is transmitted by the client (some clients use this strategy for safety reasons).
- Some clients may use a small periodic read service just for detecting if the connection with the server is alive. If this service fails, the client may take some actions like trying a reconnection.

Write Services

The OPC UA protocol provides Write services for writing at variables.

The client transmits WriteRequest services specifying the variable name (long string with hundreds of characters) together with the value to be written. The service is not optimized in terms of bandwidth consumption, but normally this service is rarely executed.

The server transmits WriteResponse services for confirming if the variable was written. For instance, if the access right of the variable is "read-only", the response will inform that the variable is not writable.

Other Services

There are other services used during the OPC UA communication typically during the initialization or finalization phase.

A short description of these services will be provided in this manual.

- **GetEndpoints:** this service is used for discovering all the different ways for establishing a session between client and server, considering the Endpoint URL and security policies. The possible ways depend on security settings made in the server.
- **CreateSession:** this service is used for creating a new session.
- **ActivateSession:** this service is used for activating a previously created session.
- **CreateSubscription:** this service is used for creating a new subscription.
- **CreateMonitoredItems:** this service is used for creating monitored items for a subscription.
- **DeleteSubscription:** this service is used for deleting a subscription.
- **DeleteMonitoredItems:** this service is used for deleting monitored items from a subscription.
- **TranslateBrowsePathsToNodeIds:** this service is used by some clients for making adjustments in the real-time database configuration.

Operational Limits

Some services of the OPC UA protocol have operational limits defined on the server. If a client calls a service exceeding its operational limit, the server will respond with ServiceFault, which typically indicates "BadTooManyOperations". This is a frequent source of interoperability issues between clients and servers.

The clients can read the operational limits from the server and respect them. Unfortunately, some clients do not read or do not respect these limits. Most of these clients at least inform about service failures due to exceeded limits, using a diagnostic or a log. In this case, some clients have parameters that can be reconfigured to correct the problem.

The following subsections describe the operational limits that are the most typical sources of interoperability problems.

MaxMonitoredItemsPerCall

This limit indicates the maximum number of monitored items:

- That can be created in a single call of service CreateMonitoredItems;
- That can be modified in a single call of service ModifyMonitoredItems;
- That can be deleted in a single call of service DeleteMonitoredItems;

Note that this limit is defined for every single call of these services. It does not limit the number of monitored items in a subscription. The client can make several subsequent calls to the service to cover any number of monitored items in the subscription.

MaxNodesPerRead

This limit indicates the maximum number of nodes (variables) that can be read using a Read service.

Note that this limit is defined for every single call of the service. The client can make several subsequent calls to the service for reading more nodes.

MaxNodesPerWrite

This limit indicates the maximum number of nodes (variables) that can be written using a Write service.

Note that this limit is defined for every single call of the service. The client can make several subsequent calls to the service for writing more nodes.

MaxNodesPerBrowse

This limit indicates the maximum number of nodes (variables) that can be browsed using a Browse or BrowseNext service.

Note that this limit is defined for every single call of the service. The client can make several subsequent calls to the service for browsing more nodes.

MaxNodesPerTranslateBrowsePathsToNodeIds

This limit indicates the maximum number of nodes (variables) that can be translated using a TranslateBrowsePathsToNodeIds service.

Note that this limit is defined for every single call of the service. The client can make several subsequent calls to the service for translating more nodes.

OPC UA Security

The OPC UA protocol provides some resources for securing communication.

Sign and Encrypt

- **Sign:** This feature ensures that the messages came from the intended source and were not modified in transit.
- **Encrypt:** This feature encrypts the messages so they cannot be read in transit. It is possible to select among several encryption algorithms.

The usage of these security resources is optional. During the server configuration, it is possible to define which security resources will be necessary for connecting to the server.

The following combinations can be used:

- None
- Sign
- Sign&Encrypt

The "None" option disables the Sign and Encrypt features.

The "Sign" option exchanges the key certificates between the client and server using cryptography. Afterwards, the communication proceeds in plain text (without cryptography).

The "Sign&Encrypt" option exchanges the key certificates between the client and server using cryptography. Afterwards, the communication proceeds also with cryptography.

Authentication

If authentication is enabled in the server, an identification (user and password) is required for creating the session.

Certificates

OPC UA makes use of the X.509 certificate standard, which defines a standard public key format. Certificates are necessary for implementing the Sign and Encrypt features.

There are two types of certificates:

- Self-signed certificate
- CA certificate (CA = Certificate Authority)

The difference between a CA certificate and a self-signed certificate is the issuer of the certificate. A self-signed certificate is created, signed, and issued by the subject of the certificate (the entity it is issued to), while a CA certificate is created, signed, and issued by a third party called a certificate authority (CA) that is authorized to validate the identity of the applicant.

A CA certificate signed by a publicly trusted CA can build trust among the communication partners, and therefore, it is used to validate public clients and servers. CA certificates have some per year cost, but not too much. The user can select among several X.509 Certificate Authorities. These entities are responsible for validating the person or organization that requests each certificate.

Self-signed certificates are normally used in cases where the issuer and the users are the same entity.

Validity of the Certificate

Certificates have a validity time, that is, they expire. New certificates must be generated before they expire. Otherwise, the communication will stop.

Encryption Algorithms

Considering the options "Sign" and "Sign&Encrypt", it is possible to select among several encryption algorithms. For instance:

- Aes128_Sha256_RsaOaep
- Basic256Sha256
- Aes256_Sha256_RsaPss

4. Basic Configuration

The configuration of a Nexto CPU as an OPC UA server requires the use of the Mastertool programmer. This section explains how to make this configuration.

Recommended Sequence of Steps for Configuration

This recommendation is usually suitable for creating a new project planned for OPC UA communication.

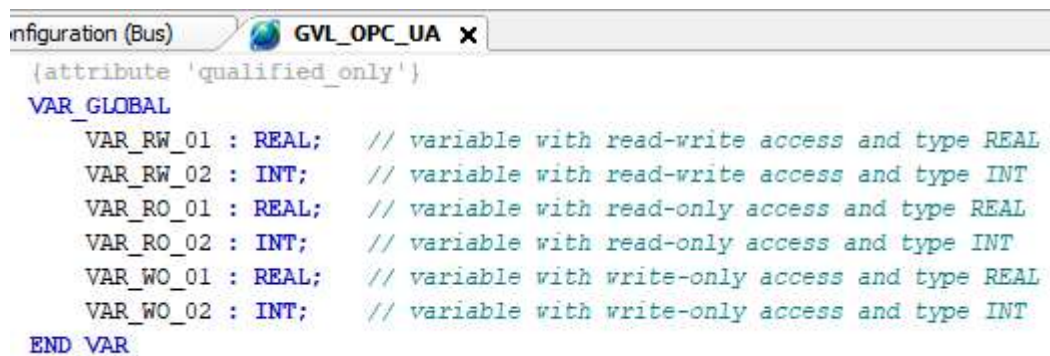
Depending on the user's needs, other sequences of steps can fit better. Some sections of this chapter describe some alternatives that fit better in other situations (for instance: add new variables to an existing configuration).

First Step - Create the Variables for OPC UA Communication

Different types of variables can be defined for OPC communication inside GVLs (Global Variables List) or POU's (Program Organization Units):

- Simple variables;
- Arrays of simple variables;
- Instance of function blocks;
- Instances of complex data structures (DUTs = Data Unit Types);
- Arrays of functions blocks;
- Arrays of DUTs.

The following example shows a GVL with 6 simple variables that will be used throughout this section to demonstrate the recommended steps for configuration of the OPC UA server in the PLC.



```

(configuration (Bus) GVL_OPC-UA x
{attribute 'qualified_only'}
VAR_GLOBAL
  VAR_RW_01 : REAL; // variable with read-write access and type REAL
  VAR_RW_02 : INT;  // variable with read-write access and type INT
  VAR_RO_01 : REAL; // variable with read-only access and type REAL
  VAR_RO_02 : INT;  // variable with read-only access and type INT
  VAR_WO_01 : REAL; // variable with write-only access and type REAL
  VAR_WO_02 : INT;  // variable with write-only access and type INT
END_VAR

```

Figure 4-1. GVL with variables for OPC UA communication

Note in the comments for each variable that the variables inside this GVL have three categories in terms of access rights:

- Read-write: OPC UA client must be able to read and write these variables.
- Read-only: OPC UA client must be able only to read these variables.
- Write-only: OPC UA client must be able only to write these variables.

Second Step - Insert the Objects "Communication Manager" and "OPC UA Server"

Right click on "Application", select "Add Object", and then select "Communication Manager".

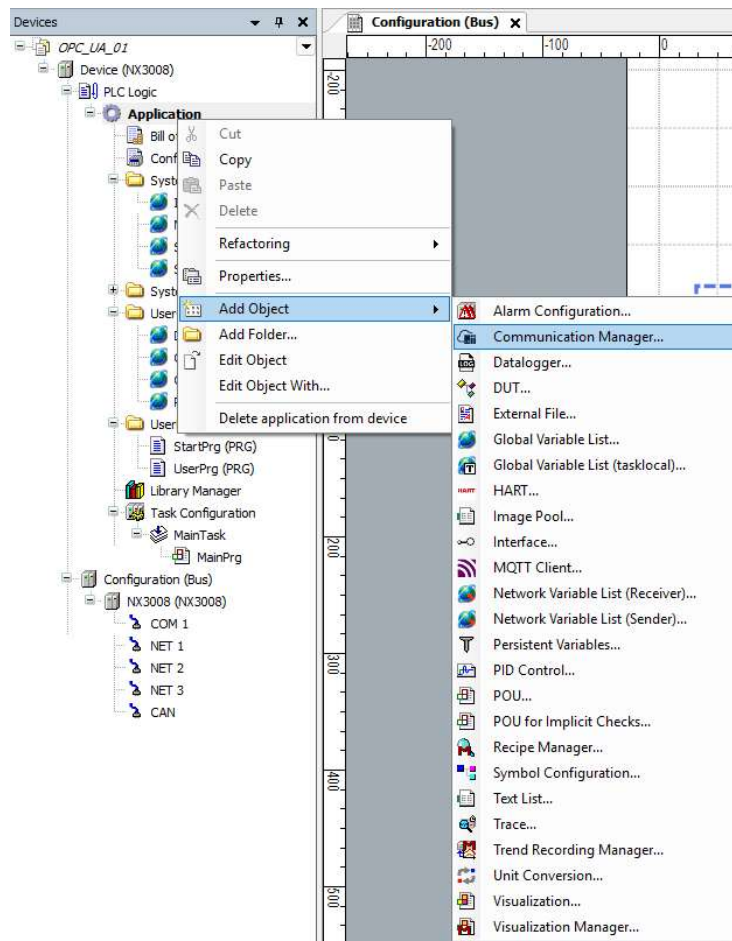


Figure 4-2. Selecting the object "Communication Manager"

In the following window, press the button "Add".

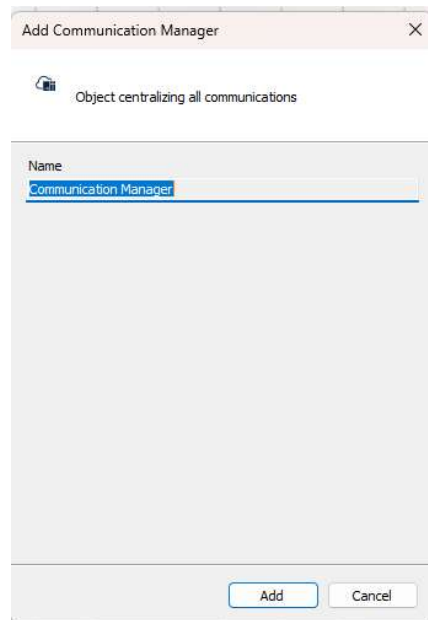


Figure 4-3. Adding the object "Communication Manager"

After this, the "Communication Manager" object appears in the device tree of the Mastertool project.

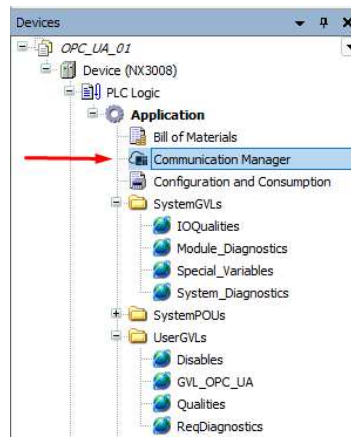


Figure 4-4. Object "Communication Manager" in the device tree

Right-click in the object "Communication Manager", select "Add Object", and then select "OPC UA Server...".

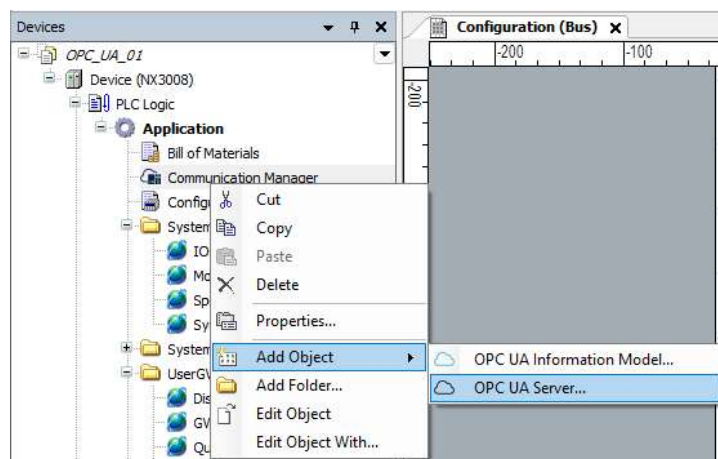


Figure 4-5. Selecting the object "OPC UA Server"

In the following window, press the button "Add".

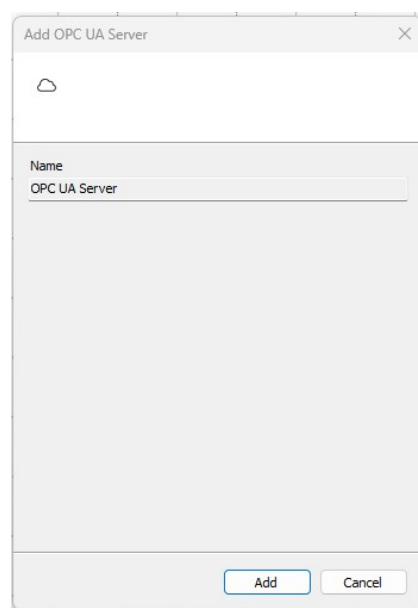


Figure 4-6. Adding the object "OPC UA Server"

After this, the "OPC UA Server" object appears in the device tree of the Mastertool project with another object "Symbol Set" below it.

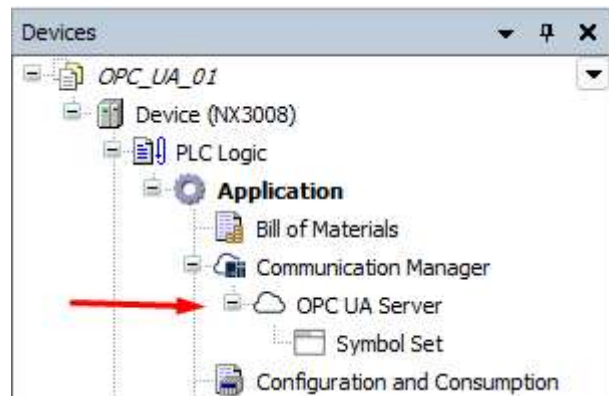


Figure 4-7. Object "OPC UA Server" in the device tree

Third Step - Selecting the OPC UA Variables and their Access Rights

Double click in the object "Symbol Set". A screen like the following appears.

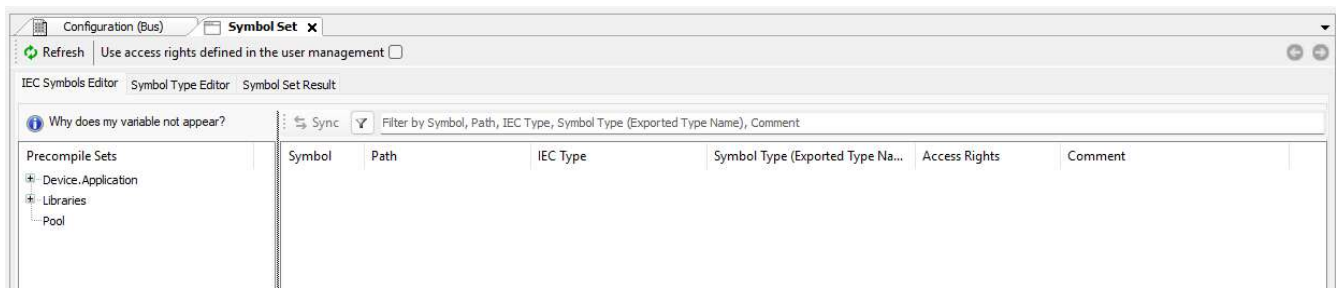


Figure 4-8. Screen for editing Symbol Set

Consider that we want to select the variables defined in GLV_OPC_UA (see the first step of this recommended sequence of steps for configuration).

In the left pane, expand "Device.Application" for finding OPC UA. Expanding one level below "Device.Application" the following screen appears:

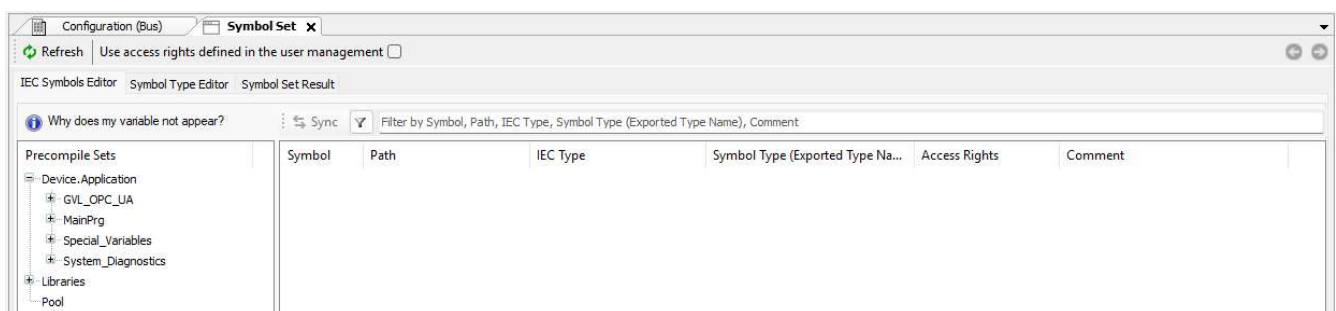


Figure 4-9. Screen for editing Symbol Set after expanding Device.Application

Now, it is already possible to drag the GVL_OPC_UA object as a whole to the right pane, producing the following screen:

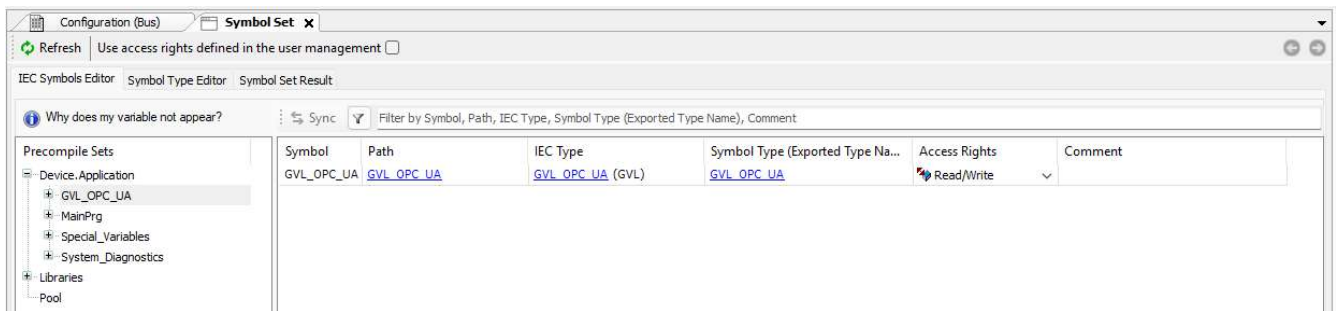


Figure 4-10. Screen for editing Symbol Set with GVL_OPC_UA as a whole

In the right pane, the column "Access Rights" can be adjusted among the following options:

- Read
- Write
- Read/Write

Because we dragged the object GVL_OPC_UA as a whole to the right pane it is not possible to configure different access rights for the different variables inside this GVL.

If you want different access rights for the variables inside this GVL, expand GVL_OPC_UA once more in the left pane, select the six internal variables, and after this drag them to the right pane.

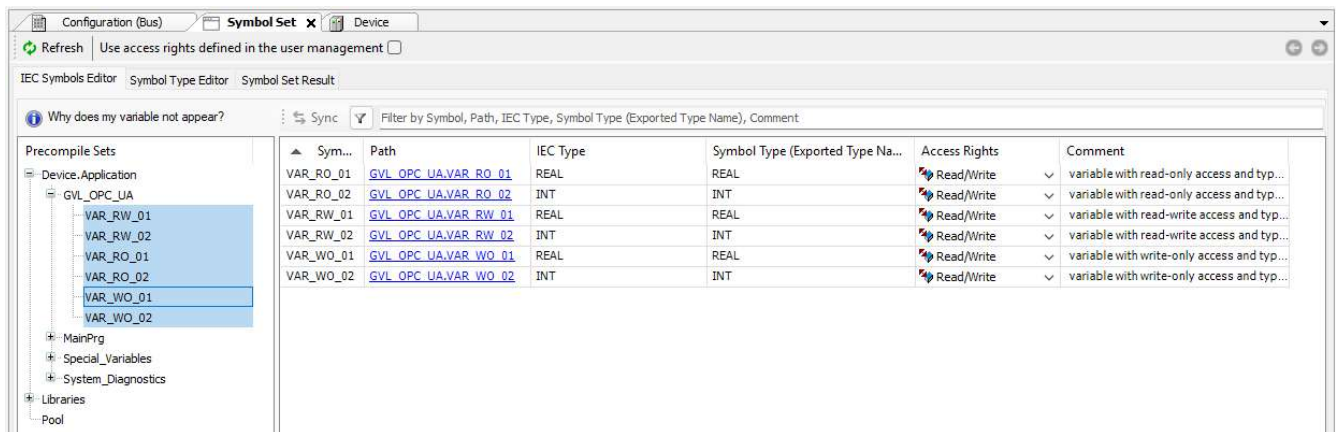


Figure 4-11. Screen for editing Symbol Set with individual variables of GVL_OPC_UA

After this, it is possible to adjust the access rights of each internal variable of GVL_OPC_UA as defined in the first step of this procedure.

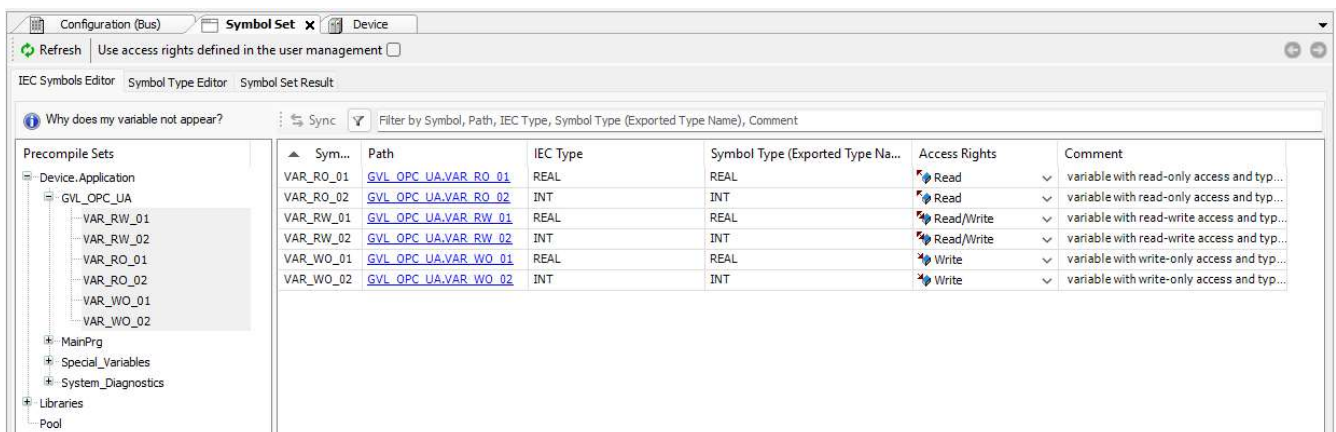


Figure 4-12. Screen for editing Symbol Set with adjusted access rights

Fourth Step - Download the Project

Download the project to the PLC and, after this, it is possible to connect an OPC UA client.

Add More Variables for OPC UA Communication Later

Sometimes it may be necessary to add more OPC variables to an existing project.

For instance, consider that we want to add more variables inside GVL_OPC_UA in the previous example (VAR_RW_NEW).



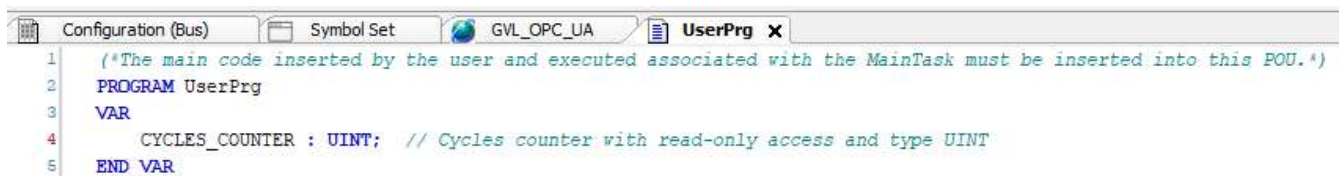
```

1 {attribute 'qualified_only'}
2 VAR_GLOBAL
3   VAR_RW_01 : REAL; // variable with read-write access and type REAL
4   VAR_RW_02 : INT;  // variable with read-write access and type INT
5   VAR_RO_01 : REAL; // variable with read-only access and type REAL
6   VAR_RO_02 : INT;  // variable with read-only access and type INT
7   VAR_WO_01 : REAL; // variable with write-only access and type REAL
8   VAR_WO_02 : INT;  // variable with write-only access and type INT
9   → VAR_RW_NEW : REAL; // new variable with read-write access and type REAL
10  END_VAR

```

Figure 4-13. Adding new OPC UA variables for communication in GVL_OPC_UA

In addition, we want to include the new variable CYCLES_COUNTER inside the POU UserPrg for OPC UA communication.



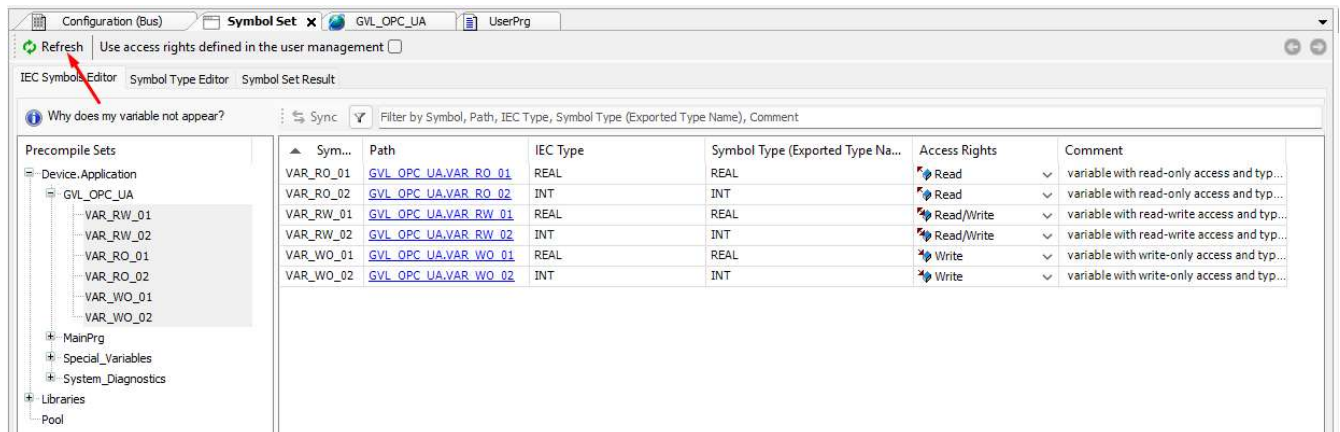
```

1 (*The main code inserted by the user and executed associated with the MainTask must be inserted into this POU.*)
2 PROGRAM UserPrg
3 VAR
4   CYCLES_COUNTER : UINT; // Cycles counter with read-only access and type UINT
5 END_VAR

```

Figure 4-14. Adding new OPC UA variables for communication in POU UserPrg

After this, open the "Symbol Set" folder again. Initially, it appears this way:



Refresh Use access rights defined in the user management ☐

IEC Symbol Editor Symbol Type Editor Symbol Set Result

Why does my variable not appear?

Precompile Sets

- Device, Application
 - GVL_OPC_UA
 - VAR_RW_01
 - VAR_RW_02
 - VAR_RO_01
 - VAR_RO_02
 - VAR_WO_01
 - VAR_WO_02
 - MainPrg
 - Special_Variables
 - System_Diagnostics
 - Libraries
 - Pool

Sym...	Path	IEC Type	Symbol Type (Exported Type Na...	Access Rights	Comment
VAR_RO_01	GVL_OPC_UA.VAR_RO_01	REAL	REAL	Read	variable with read-only access and typ...
VAR_RO_02	GVL_OPC_UA.VAR_RO_02	INT	INT	Read	variable with read-only access and typ...
VAR_RW_01	GVL_OPC_UA.VAR_RW_01	REAL	REAL	Read/Write	variable with read-write access and typ...
VAR_RW_02	GVL_OPC_UA.VAR_RW_02	INT	INT	Read/Write	variable with read-write access and typ...
VAR_WO_01	GVL_OPC_UA.VAR_WO_01	REAL	REAL	Write	variable with write-only access and typ...
VAR_WO_02	GVL_OPC_UA.VAR_WO_02	INT	INT	Write	variable with write-only access and typ...

Figure 4-15. Refresh is necessary to include newly created variables

Press the button "Refresh" to include the new variables recently added and expand "UserPrg" to find "CYCLES_COUNTER". After this, the left pane includes the newly created variables:

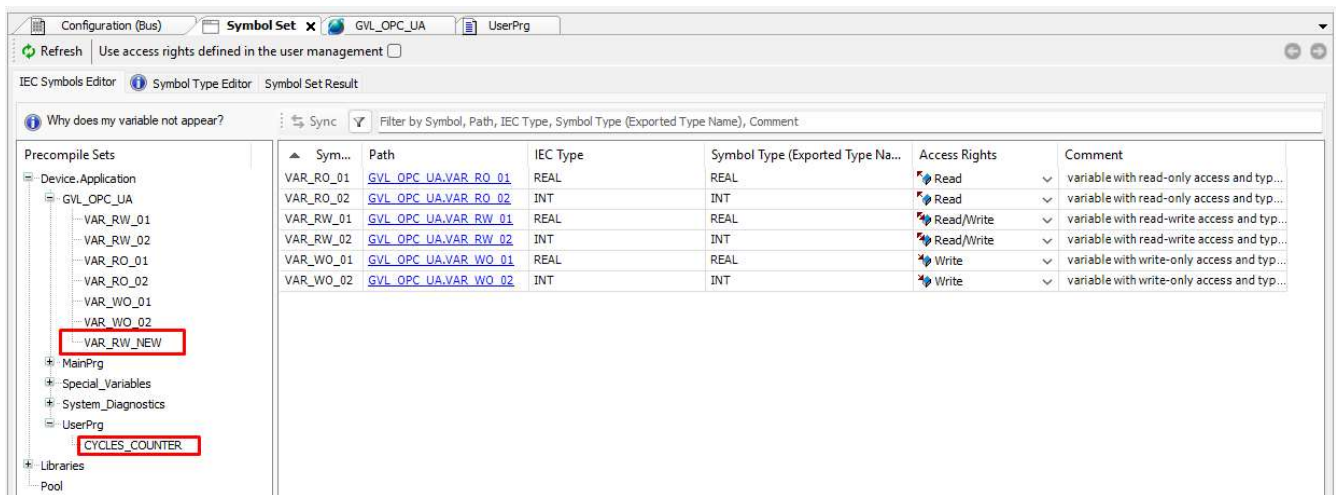


Figure 4-16. After refresh

After this, it is possible to drag these new variables to the right pane and adjust their access rights, resulting in the following final configuration.

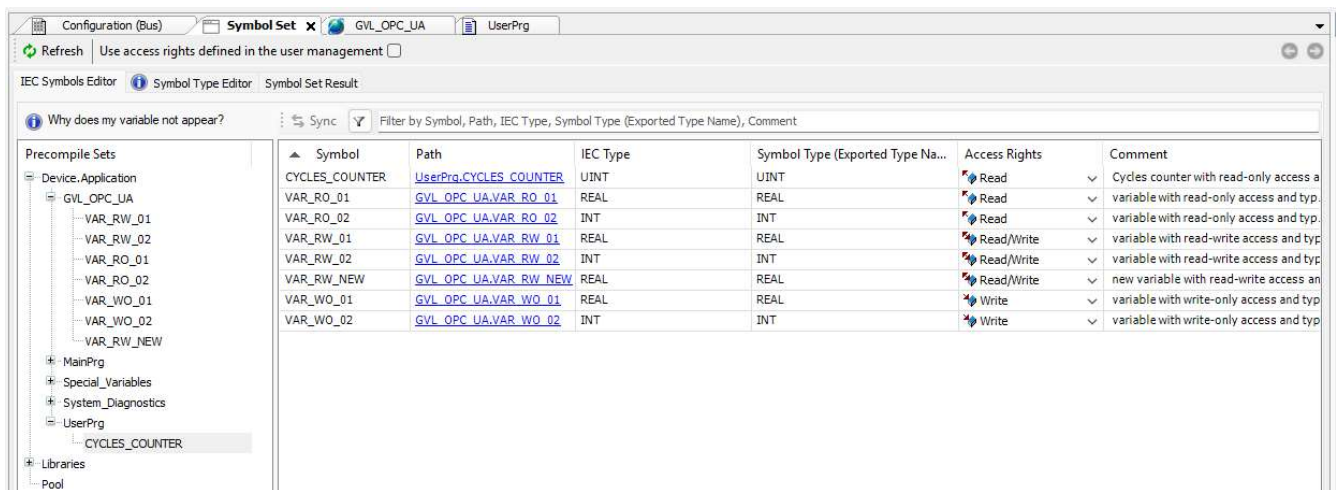


Figure 4-17. Configuration with new variables

Now the project is ready for download. After downloading it to the PLC, it is possible to connect the OPC UA client to add the new variables in the OPC UA communication.

5. Security Configuration

Before adjusting the security configurations for a PLC discussed in this chapter, it is necessary to select the network path to this PLC. This can be done using the following screen:

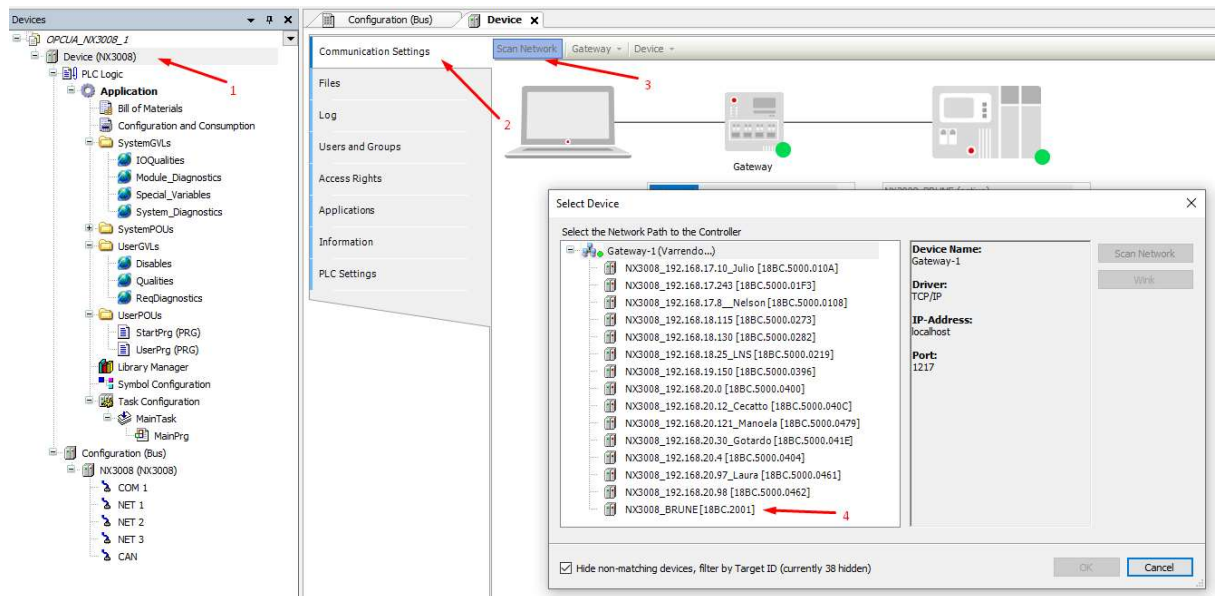


Figure 5-1. Selecting network path to the PLC

1. Click over "Device" in the device tree.
2. Select tab "Communication Settings" in the window "Device".
3. Press button "Scan Network".
4. Select the PLC double-clicking over it.

After this, the selected network path is shown:

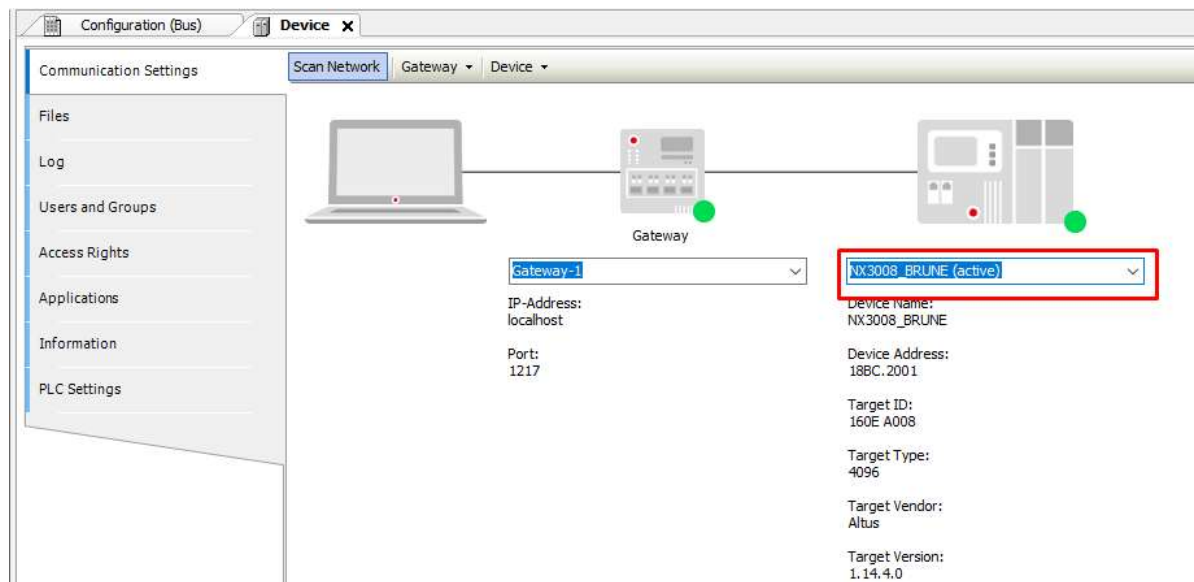


Figure 5-2. Selecting network path to the PLC

Please read section **OPC UA Security** for an introduction about the security features used in the OPC UA protocol (authentication, sign, and encrypt).

From the point of view of authentication, it is possible to activate or deactivate. Device users must be created and selected before activating authentication.

From the point of view of sign and encrypt, it is possible to choose between three options:

- None
- Sign
- Sign & Encrypt

Most security settings can be adjusted as explained in the following figures.

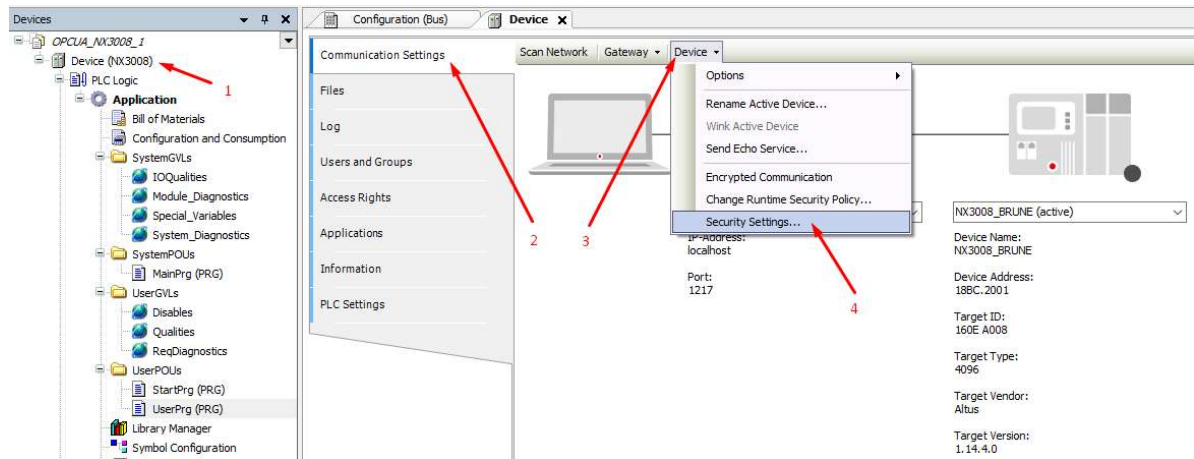


Figure 5-3. Opening window with security settings

1. Click on "Device" in the device tree. This will open the tab "Device".
2. Select "Communication Settings" in the tab "Device".
3. Select "Device".
4. Click over "Security Settings..."

After clicking over "Security Settings" the following window opens:

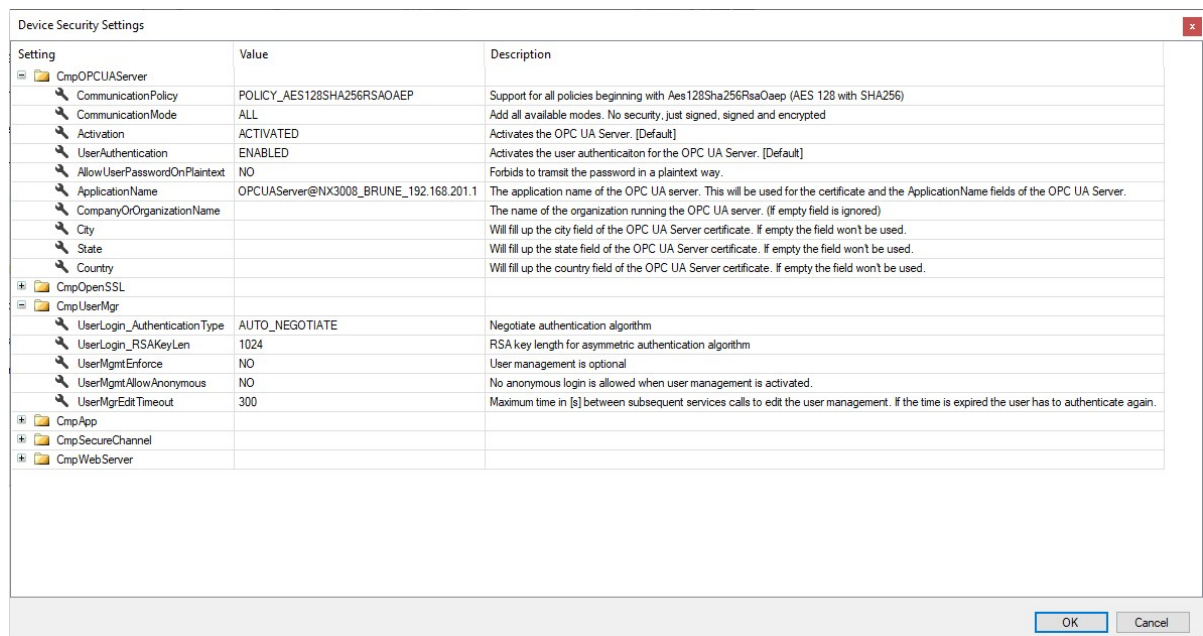


Figure 5-4. Window with security settings

In the previous windows, the security settings relevant for OPC UA can be seen and adjusted after expanding the folders "CmpOPCUAServer" and "CmpUserMgr".

ATTENTION:

After changing some of the security settings in the previous window it is necessary to power-off and power-on again the CPU before these settings become active. As a general recommendation, please power-off and power-on after changing any of the security settings in the previous window.

Anonymous Login (No Authentication)

If you want to connect without authentication, it is necessary to select anonymous login in the client, and configure the server for accepting anonymous login.

Considering the window in Figure 5-4, the following setting must be adjusted:

- CmpUserMgr / UserMgmtAllowAnonymous = YES

Authentication

Authentication consists in selecting a device user name and a password for establishing a session with the server.

Creating a New Device User and its Password

The first step for enabling authentication is to create a new device user and its password, and then select this new device user. This is explained in the following figures.

ATTENTION:

This is a very summarized description for device user management. More details can be found in the **Mastertool X User Manual (MU299611)**. This manual describes many other functions for device user management (editing passwords, deleting users, importing users, etc).

The following figures assume that:

- Anonymour login is not allowed. This means that, in Figure 5-4, the following setting must be adjusted: CmpUserMgr / UserMgmtAllowAnonymous = NO.
- There is a pre-existing user called "brune" which is member of group "Administrator".

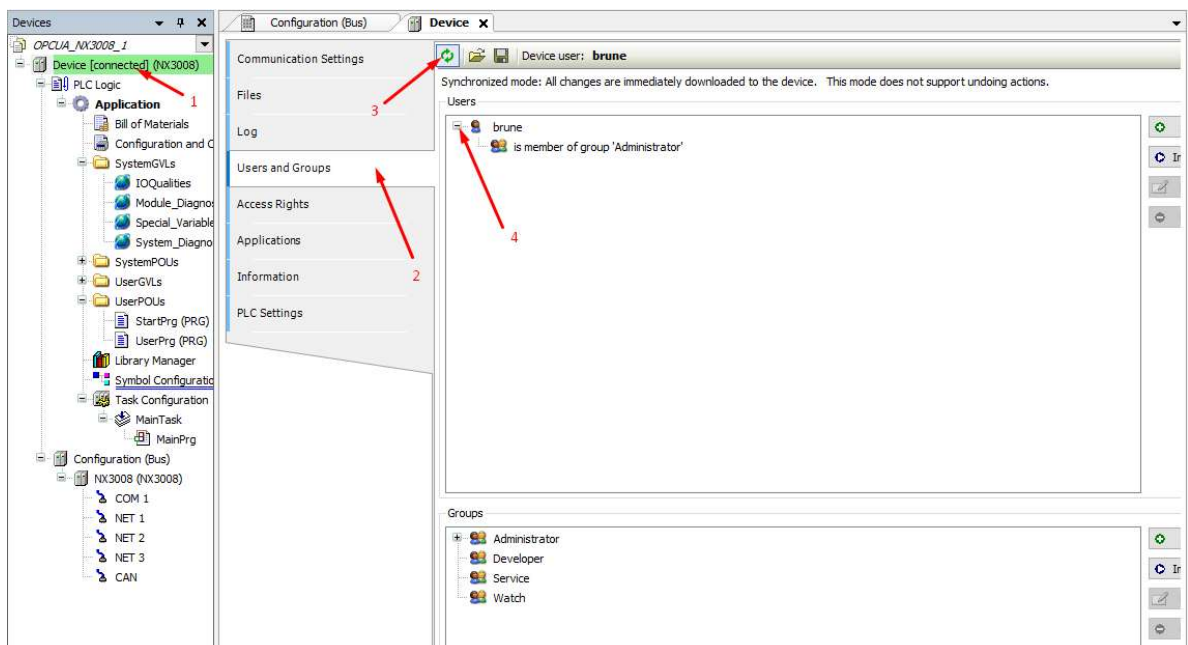


Figure 5-5. Opening screen for adding users

1. Click on "Device" in the device tree. This will open the tab "Device".

2. Select "Users and Groups" in the tab "Device".
3. Press the refresh button (🔄) for showing the current users. In this example, there is a pre-existing user called "brune" which is currently selected.
4. Expand the current users if you want to see their groups. In this example, the user "brune" is a member of group "Administrator".

After this, it is possible to create another user, its group and password by pressing the button "Add" in the left pane of the previous screen:

The 'Add User' dialog box is shown with the following fields and options:

- Name:** An empty text input field.
- Default group:** A dropdown menu showing '<please select>'.
- Password:** An empty password input field with a visibility toggle (eye icon).
- Confirm password:** An empty password input field.
- Password strength:** A red label indicating 'Very weak'.
- Options:**
 - ☒ Password can be changed by user
 - ☐ Password must be changed at first login
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

Figure 5-6. Window for adding a device user

The following figure shows this window filled in with a new user "roberto" of group "Administrator":

The 'Add User' dialog box is now filled with the following information:

- Name:** 'roberto'
- Default group:** 'Administrator' (selected in the dropdown)
- Password:** A masked password field (dots).
- Confirm password:** A masked password field (dots).
- Password strength:** A yellow label indicating 'Better'.
- Options:**
 - ☒ Password can be changed by user
 - ☐ Password must be changed at first login
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

Figure 5-7. Adding device user "roberto"

After creating the user, the following figure shows up:



Figure 5-8. Device user "roberto" added

For changing the to the new device user (roberto), first logoff the currently selected device user (brune). This can be done using the following command:

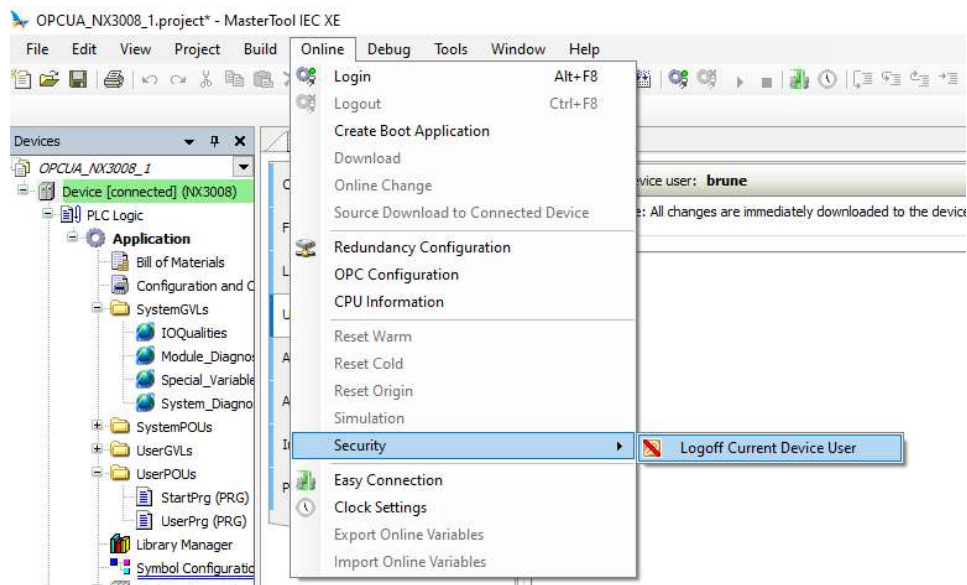


Figure 5-9. Logging off the current device user

After this, you get the following screen showing and "Anonymous" device user.



Figure 5-10. Anonymous device user

For selecting one of the registered users (brune or roberto), click over the refresh button (🔄) in the previous screen. This causes the following window to open:

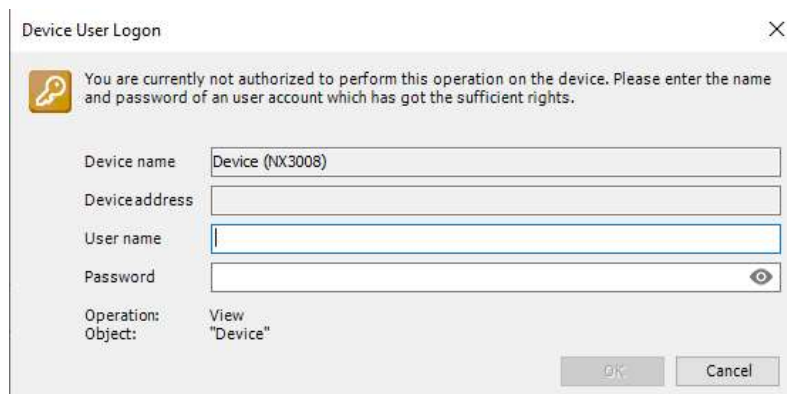


Figure 5-11. Logging on a device user

Finally, type the name of the device user and password for logging on.

Adjusting the OPC UA Settings for Authentication

In the screen of Figure 5-4, the following security setting must be adjusted for not accepting connection without authentication:

- CmpUserMgr / UserMgmtAllowAnonymous = NO

In addition, if you will not use cryptography together with authentication, the following security setting must be adjusted:

- CmpOPCUAServer / AllowUserPasswordsOnPlainText = YES

ATTENTION:

Remember to power-down and power-up again the PLC after changing these settings.

After this, the client will need to provide user and password for establishing an OPC UA connection.

The following figure shows an example of user and password configuration in the client UaExpert®:

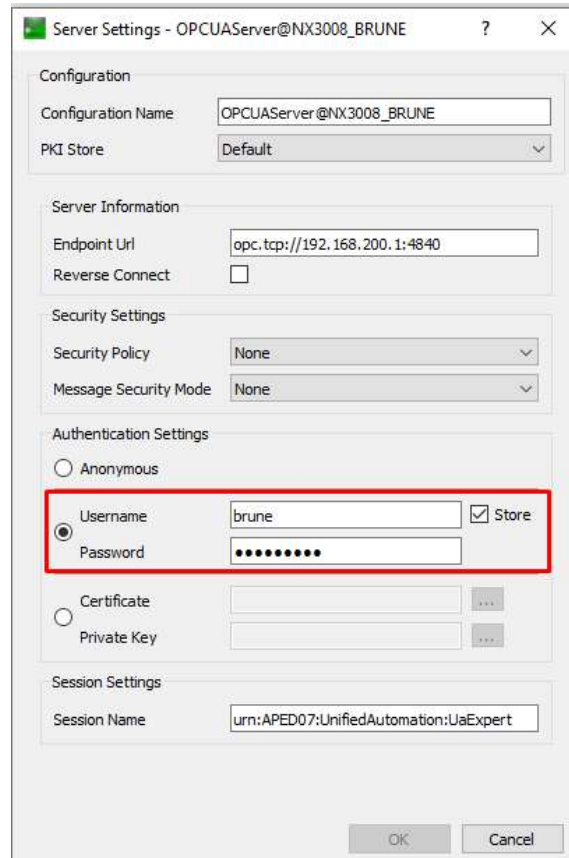


Figure 5-12. User and password in client UaExpert®

Sign and Encrypt

Configuration Options in Altus Servers

Regarding sign and encrypt, the user must adjust the two following parameters in the screen of Figure 5-4:

- Communication Policy
- Communication Mode

The following options of Communication Policy are available for Altus controllers supporting OPC UA server:

Setting	Value	Description
CmpOPCUAServer		
CommunicationPolicy	POLICY_AES128SHA256RSAOAEF	Support for all policies beginning with Aes128Sha256RsaOaep (AES 128 with SHA256)
CommunicationMode	POLICY_AES256SHA256RSAPSS	Support for all policies beginning with Aes256Sha256RsaPss (AES 256 with SHA256 and RSA with PSS/OAEP(SHA256))
Activation	POLICY_BASIC256SHA256	Support for all policies beginning with Basic256Sha256 (AES 256 with SHA256)
	POLICY_AES128SHA256RSAOAEF	Support for all policies beginning with Aes128Sha256RsaOaep (AES 128 with SHA256)

Figure 5-13. Options for Communication Policy

The following options of Communication Mode are available for Altus controllers supporting OPC UA server:

Setting	Value	Description
CmpOPCUAServer		
CommunicationPolicy	POLICY_AES128SHA256RSAOAEF	Support for all policies beginning with Aes128Sha256RsaOaep (AES 128 with SHA256)
CommunicationMode	ALL: Add all available modes. No sec	Add all available modes. No security, just signed, signed and encrypted
Activation	SIGNED_AND_ENCRYPTED: Only signed and encrypted communications profiles are added [HIGHEST_SECURITY_LEVEL]	
UserAuthentication	MIN_SIGNED: Enforce a signed communication. Encryption optional available.	
	ALL: Add all available modes. No security, just signed, signed and encrypted	
AllowUserPasswordO...	ONLY_PLAINTEXT: Support only plain-text communication. Default without security manager.	

Figure 5-14. Options for Communication Mode

Configuration Options in OPC UA Clients

In the client side, for instance, UaExpert®, it is possible to select:

- Security Policy, which offers the following encryption methods supported by Altus controllers:
 - Aes256Sha256RsaPss (encryption method with the highest security).
 - Basic256Sha256 (encryption method with the second highest security).
 - Aes128Sha256RsaOaep (encryption method with the third highest security).
 - None (no encryption - plain text).
- Message Security Mode, which offers the following options:
 - Sign & Encrypt (highest security level - ensures that the messages came from the intended source and were not modified in transit, and that messages cannot be read in transit).
 - Sign (second highest security level - ensures that the messages came from the intended source and were not modified in transit).
 - None (no security at all).

Altus Server Configurations and Allowed Configurations in OPC UA Clients

Depending on the security configurations selected for the Altus server (CommunicationPolicy and CommunicationMode), there is a list of supported configurations in the client side (Security Policy and Message Security Mode). This relationship appears in the following table:

- The two first columns of the following table shows all the 12 combinations of configurations in the Altus OPC UA server (3 options of CommunicationPolicy X 4 options of CommunicationMode).
- The two last columns show 37 possible configurations in a client like UaExpert®.
- Some additional notes about this table:
 - Rows 1, 2 and 3 have no security at all.
 - Row 4 has the highest security level.

Row	ALTUS OPC UA Server Configurations		Supported OPC UA Client Configurations	
	Communication Policy	Communication Mode	Security Policy	Message Security Mode
1	POLICY_AES256SHA256RSAPSS	ONLY_PLAIN_TEXT	None	None
2	POLICY_BASIC256SHA256	ONLY_PLAIN_TEXT	None	None
3	POLICY_AES128SHA256RSAAOEP	ONLY_PLAIN_TEXT	None	None
4	POLICY_AES256SHA256RSAPSS	SIGNED_AND_ENCRYPTED	Aes256Sha256RsaPss	Sign & Encrypt
5	POLICY_BASIC256SHA256	SIGNED_AND_ENCRYPTED	Aes256Sha256RsaPss	Sign & Encrypt
6			Basic256Sha256	Sign & Encrypt
7	POLICY_AES128SHA256RSAAOEP	SIGNED_AND_ENCRYPTED	Aes256Sha256RsaPss	Sign & Encrypt
8			Basic256Sha256	Sign & Encrypt
9			Aes128Sha256RsaOaep	Sign & Encrypt
10	POLICY_AES256SHA256RSAPSS	MIN_SIGNED	Aes256Sha256RsaPss	Sign & Encrypt
11			Aes256Sha256RsaPss	Sign
12	POLICY_BASIC256SHA256	MIN_SIGNED	Aes256Sha256RsaPss	Sign & Encrypt
13			Aes256Sha256RsaPss	Sign
14			Basic256Sha256	Sign & Encrypt
15			Basic256Sha256	Sign
16	POLICY_AES128SHA256RSAAOEP	MIN_SIGNED	Aes256Sha256RsaPss	Sign & Encrypt
17			Aes256Sha256RsaPss	Sign
18			Basic256Sha256	Sign & Encrypt
19			Basic256Sha256	Sign
20			Aes128Sha256RsaOaep	Sign & Encrypt
21			Aes128Sha256RsaOaep	Sign
22	POLICY_AES256SHA256RSAPSS	ALL	Aes256Sha256RsaPss	Sign & Encrypt
23			Aes256Sha256RsaPss	Sign
24			None	None
25	POLICY_BASIC256SHA256	ALL	Aes256Sha256RsaPss	Sign & Encrypt
26			Aes256Sha256RsaPss	Sign
27			Basic256Sha256	Sign & Encrypt
28			Basic256Sha256	Sign
30			None	None
31	POLICY_AES128SHA256RSAAOEP	ALL	Aes256Sha256RsaPss	Sign & Encrypt
32			Aes256Sha256RsaPss	Sign
33			Basic256Sha256	Sign & Encrypt
34			Basic256Sha256	Sign
35			Aes128Sha256RsaOaep	Sign & Encrypt
36			Aes128Sha256RsaOaep	Sign
37			None	None

Table 5-1 -Relation between security settings in server and client

Creating or Importing Certificates

Before using the message security modes "Sign" or "Sign & Encrypt", it is necessary to create a certificate. As already described in section **Certificates**, there are two types of certificates:

- Self-signed certificate
- CA certificate (CA = Certificate Authority)

The section **Certificates** also explains the criteria for selecting one of these two types. The following subsections describe how to create a self-signed certificate or how to import a CA certificate, depending on the type of certificate selected by the user.

Creating a Self-Signed Certificate

Initially use the menu **View / Security Screen** for opening the window shown in the following figure, and then select the tab **Devices**:

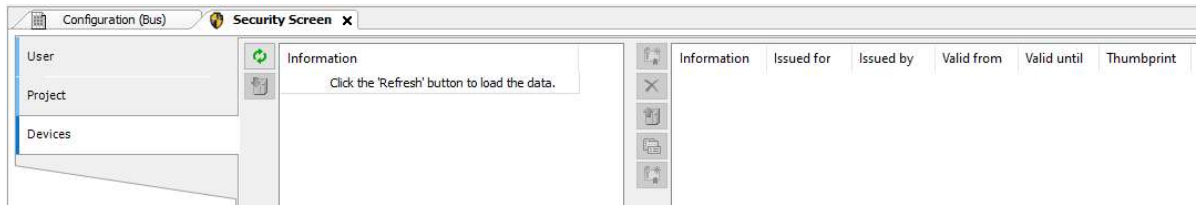


Figure 5-15. Security Screen / tab Devices

Click in the refresh button (1), and then click over Device (2) for obtaining the following screen:

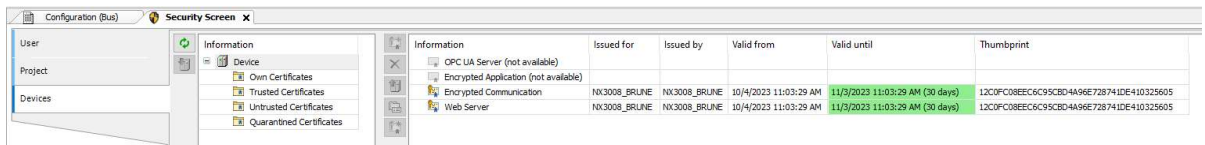


Figure 5-16. Security Screen / tab Devices - refreshed

Click over "OPC UA Server (not available)" (3) in the Information column, and then in the button for generating a certificate (4):

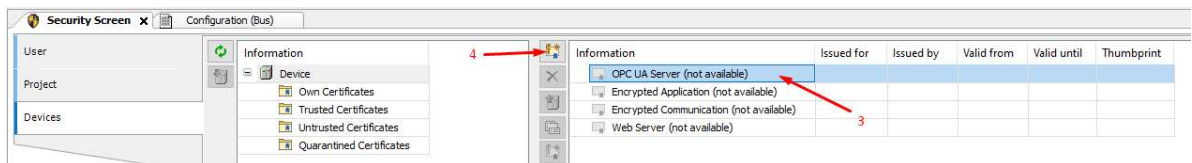


Figure 5-17. Security Screen / tab Devices - starting creation of certificate

After this, the following message box appears for adjusting the certificate settings:

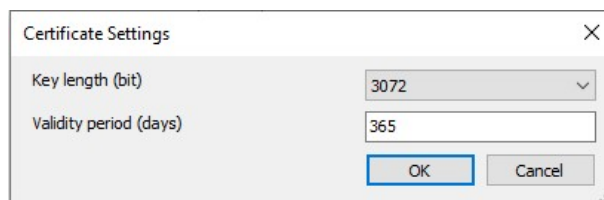


Figure 5-18. Security Screen / tab Devices - certificate settings

You can use these default setting or chose between the following options:

- Key length (bit): 2048, 3072, or 4096. The bigger the key length, more time it takes for creating the certificate, but hackers will also need more time for breaking this certificate.
- Validity period (days). The bigger the validity period, more time hackers have for trying to break this certificate. So, it is not recommended to select a huge validity period.

After clicking OK in the previous message box, it takes some seconds (sometimes minutes, depending on the CPU power) for creating the certificate. After the certificate was calculated, the following screen appears:

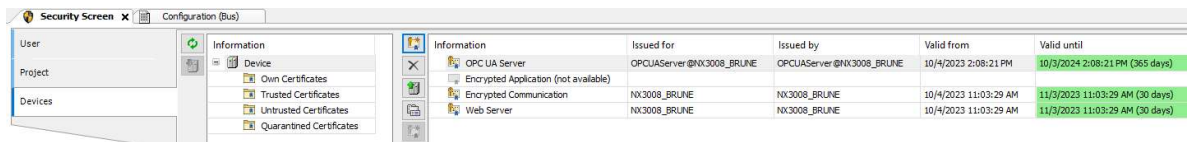


Figure 5-19. Security Screen / tab Devices - certificate created

ATTENTION:

Power-off and power-on again the PLC for activating the created certificate.

Importing a CA Certificate

For now, Altus PLCs cannot use CA certificates.

Negotiating the Certificate with the OPC UA Client

Self-Signed Certificate

After the self-signed certificate was created in the server, configure the client with one of the allowed configurations for parameters Security Policy and Message Security Mode, according to the description in section **Altus Server Configurations and Allowed Configurations in OPC UA Clients**.

The first attempt to connect will fail, but will leave a certificate in the folder "Quarantined Certificates". The following figure shows an example after the first attempt to connect using the client UaExpert®. Before checking the folder "Quarantined Certificates", press the refresh button.

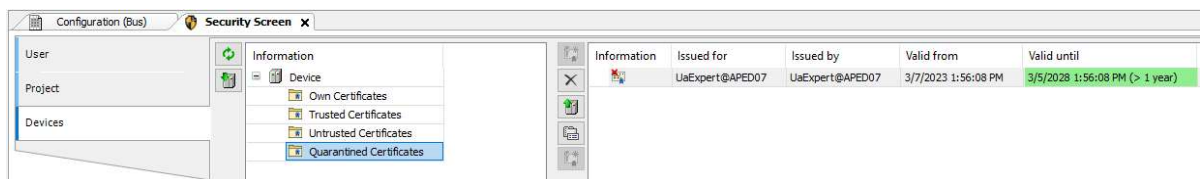


Figure 5-20. Security Screen / tab Devices - quarantined certificate for client

Drag this certificate from the folder "Quarantined Certificates" to the folder "Trusted Certificates".

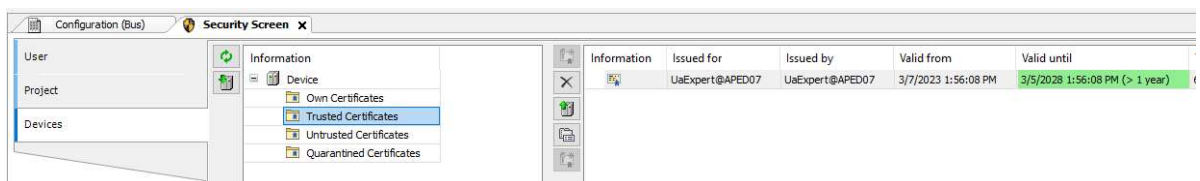


Figure 5-21. Security Screen / tab Devices - trusted certificate for client

After this, the next attempt of client connection must work.

CA Certificate

For now, Altus PLCs cannot use CA certificates.

6. Recommendations for OPC UA Clients Configuration

The configuration of Nexto Series CPUs as OPC UA servers, described in chapter 4, is quite simple. The configuration of OPC UA clients is a little bit more complicated, and requires some knowledge about the OPC UA protocol. For this purpose, it is important to read chapter 3.

Of course, the user also needs to read the documentation of the OPC UA client.

This chapter gives some general hints related to configuration of OPC UA clients connected to a Nexto Series CPU, for achieving the following objectives:

- Avoid to work with unintended parameters for subscriptions and monitored items;
- Avoid interoperability issues;
- Avoid performance issues. Depending on how the client is configured, the OPC UA server tasks can have a high CPU consumption.

Configure only Supported Values of Parameters `PublishingInterval` and `SamplingInterval`

According to section **Revised Parameters for Subscriptions and Monitored Items**, the following parameters configured in the client may be revised by the server:

- `PublishingInterval`
- `SamplingInterval`

Sometimes the client may not notify the user about such a revision made by the server. Therefore, the user configuration may be changed, and the user may not be warned about this change.

For avoiding these revisions, the user should use one of the values supported by the server.

Values supported for the parameter `PublishingInterval`:

- Any multiple of 100 ms.

Values supported for the parameter `SamplingInterval`:

- 100 ms
- 300 ms
- 500 ms
- 1000 ms
- 2500 ms
- 5000 ms

Observe the Operational Limits

The meaning of operational limits is described in section **Operational Limits**, and their values are defined in section **Values of Operational Limits**. When a client exceeds an operational limit of a server, some OPC UA services will fail, causing interoperability issues.

A really good client would never exceed an operational limit, because it can read the operational limits from the server, and do not call services exceeding any operational limit.

If a client ignores and exceeds an operational limit, some OPC UA services will fail. In this case, at least the client should report this failure in a log. After observing such a log, the user has two options:

- The user can change a parameter in the client that avoids to exceed an operational limit. For instance, in the client Kepware®, it is possible to adjust the communication parameter "Max. Items per Read" to be smaller or equal to the operational limit "MaxNodesPerRead".

- If the client does not offer a configurable parameter that avoids to exceed a specific operational limit, the solution for the problem is more difficult, and sometimes not possible. Sometimes the solution can be splitting the communication in many sessions or subscriptions with a limited number of variables.

Avoid Small Values for `SamplingInterval` and `PublishingInterval`

The smaller the values of these two parameters for a subscription, the bigger will be the CPU consumption in the server.

It is important to evaluate carefully how small must be these parameters, and avoid to use unnecessarily low values for them. During this evaluation, the user must answer some questions:

- Which is the update rate needed for a variable in the client? This should be the value for `PublishingInterval`.
- Which is the response time needed for a variable in the client (time since variable changed in the server until the new value arrives at the client)? This should be the value for `PublishingInterval` + `SamplingInterval`.

Split a Session in Several Subscriptions

Some variables must be updated more frequently than other variables. For instance, a slow variable like a temperature does not need to be updated very frequently.

It is recommended to categorize the variables in several groups according to the expected update rate, and use a different subscription for each category.

If all the variables belong to the same subscription, this single subscription should have small values for `SamplingInterval` and `PublishingInterval` suitable for fast variables, causing an unnecessary CPU consumption for reporting slow variables.

Most clients allow to create many subscriptions for a session.

Split the Communication in Several Sessions

This measure normally helps to increase the communication performance in the client side. Typically the operating system of a client (e.g: Windows®) allocates a thread for each session. Processing several sessions in parallel by different threads generally optimizes the communication performance.

Note that there is a limit for the number of sessions in the server (see section **Maximum Number of Sessions**).

7. Diagnostics

Using OPC UA communication, it is possible to get diagnostic and other types of information about the server.

This chapter shows how to get this information using the OPC UA client UaExpert®.

Additional diagnostics, like failures to connect to a server, must be provided by the OPC UA client.

Server Diagnostics

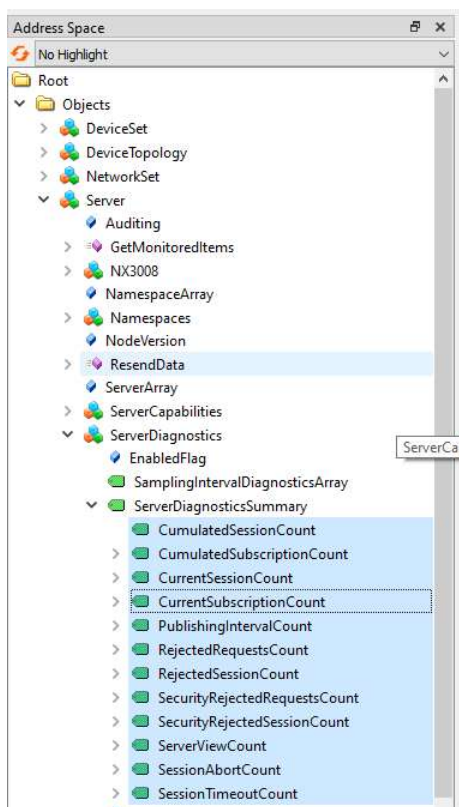


Figure 7-1. Browsing server diagnostics

Data Access View								
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	OPCUAServer@...	NS0 Numeric ...	CumulatedSessionCount	1	UInt32	10:53:32.667	10:53:32.667	Good
2	OPCUAServer@...	NS0 Numeric ...	CumulatedSubscriptionCount	5	UInt32	10:53:32.667	10:53:32.667	Good
3	OPCUAServer@...	NS0 Numeric ...	CurrentSessionCount	1	UInt32	10:53:32.667	10:53:32.667	Good
4	OPCUAServer@...	NS0 Numeric ...	CurrentSubscriptionCount	2	UInt32	10:53:32.667	10:53:32.667	Good
5	OPCUAServer@...	NS0 Numeric ...	PublishingIntervalCount	6	UInt32	10:53:32.667	10:53:32.667	Good
6	OPCUAServer@...	NS0 Numeric ...	RejectedRequestsCount	0	UInt32	10:53:32.667	10:53:32.667	Good
7	OPCUAServer@...	NS0 Numeric ...	RejectedSessionCount	0	UInt32	10:53:32.667	10:53:32.667	Good
8	OPCUAServer@...	NS0 Numeric ...	SecurityRejectedRequestsCount	0	UInt32	10:53:32.667	10:53:32.667	Good
9	OPCUAServer@...	NS0 Numeric ...	SecurityRejectedSessionCount	0	UInt32	10:53:32.667	10:53:32.667	Good
10	OPCUAServer@...	NS0 Numeric ...	ServerViewCount	0	UInt32	10:53:32.667	10:53:32.667	Good
11	OPCUAServer@...	NS0 Numeric ...	SessionAbortCount	0	UInt32	10:53:32.667	10:53:32.667	Good
12	OPCUAServer@...	NS0 Numeric ...	SessionTimeoutCount	0	UInt32	10:53:32.667	10:53:32.667	Good

Figure 7-2. Showing server diagnostics

Operational Limits

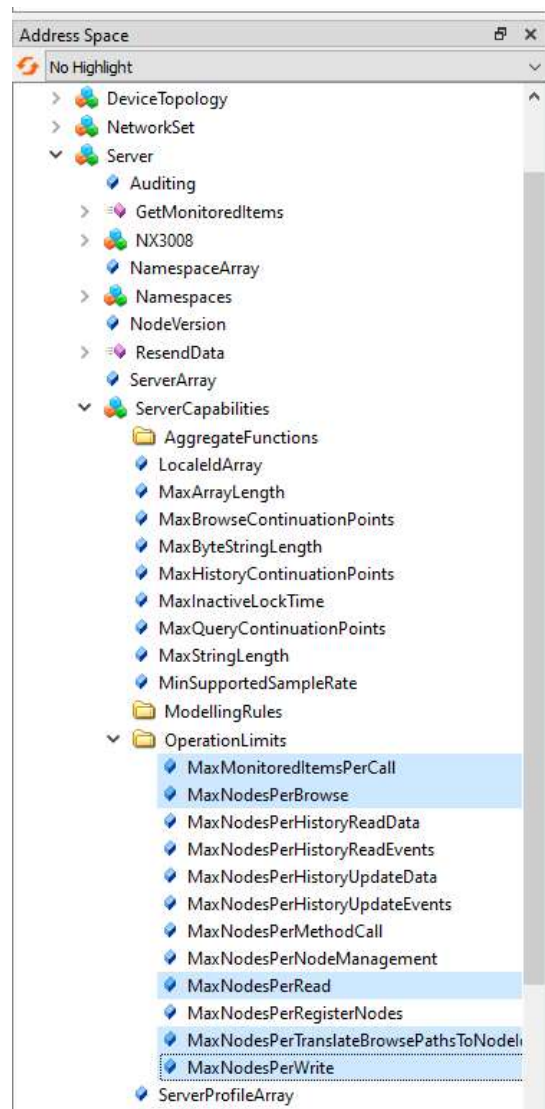


Figure 7-3. Browsing operational limits

Data Access View								
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	OPCUAServer@...	NS0[Numeric]...	MaxMonitoredItemsPerCall	1000	UInt32	10:40:20.780	10:40:20.780	Good
2	OPCUAServer@...	NS0[Numeric]...	MaxNodesPerBrowse	1000	UInt32	10:40:20.780	10:40:20.780	Good
3	OPCUAServer@...	NS0[Numeric]...	MaxNodesPerRead	1000	UInt32	10:40:20.780	10:40:20.780	Good
4	OPCUAServer@...	NS0[Numeric]...	MaxNodesPerTranslateBrowsePathsToNodeIds	100	UInt32	10:40:20.790	10:40:20.790	Good
5	OPCUAServer@...	NS0[Numeric]...	MaxNodesPerWrite	1000	UInt32	10:40:20.790	10:40:20.790	Good

Figure 7-4. Showing operational limits

8. Performance Analysis

This chapter analyzes the performance of OPC UA communication using several types of Nexto PLCs. The idea is to give information about CPU load, initialization time, and the number of variables (monitored items) that can be used in OPC UA communication.

This information depends on many factors:

- CPU processing power
- OPC UA communication parameters, mainly Publishing Interval and Sampling Interval.
- Number of sessions.

General Test Conditions

All the test results reported in this chapter used the following fixed test conditions:

- The OPC UA client was KEPServerEX® V6.14.263.0 from Kepware®.
- The maximum number of variables per session was 5000. For 5000 variables or less, a single session was used.
- Security was not used (authentication, sign, encrypt).
- PLC project features:
 - Project profile: machine profile.
 - MainTask interval: 20 ms.
 - All the OPC UA variables have type UDINT and are incremented in each MainTask interval.
 - MainTask cycle time: near 0 ms (only the code necessary for incrementing all the OPC UA variables).
 - No local I/O modules and no remote I/O.
 - No other types of communication (only Mastertool logged in).

Test Scenarios

The test scenarios are reported in a table.

The following leftmost columns define the test condition:

- CPU model and firmware version
- Number of variables
- Publishing Interval
- Sampling Interval

The following rightmost columns define the test results:

- Initialization time: time since client was started until variables started to be reported.
- Total CPU load: total CPU consumption (%).

Test #	Test Conditions				Test Results	
	CPU Model (firmware version)	Number of Variables	Publishing Interval (ms)	Sampling Interval (ms)	Initialization Time (s)	CPU Load (%)
1	NX3008 (1.12.32.0)	0				31%
2		1000	100	100	0,32	51%
3		1000	1000	1000	1,11	37%
4		5000	1000	1000	1,17	52%
5		10000	1000	1000	2,22	65%
6		15000	1000	1000	4,43	74%
7		20000	1000	1000	5,69	83%
8		25000	1000	1000	8,86	90%
9		30000	1000	1000	13,73	96%
10	XP340 (1.12.32.0)	0				47%
11		500	100	100	1,02	85%
12		500	1000	1000	1,15	59%
13		1000	1000	1000	1,48	65%
14		2000	1000	1000	3,18	75%
15		3000	1000	1000	5,57	83%
16		4000	1000	1000	10,11	89%
17		5000	1000	1000	12,85	95%
18	NX3010 (1.10.12.0)	0				27%
19		500	100	100	1,15	88%
20		500	1000	1000	1,12	38%
21		1000	1000	1000	1,60	45%
22		2000	1000	1000	3,38	59%
23		3000	1000	1000	5,71	72%
24		4000	1000	1000	8,73	85%
25		5000	1000	1000	12,48	98%
26	NX3003 (1.12.32.0)	0				42%
27		250	100	100	1,50	98%
28		500	1000	1000	2,19	62%
29		1000	1000	1000	4,56	75%
30		2000	1000	1000	12,72	99%

Table 8-1 -Test scenarios

The previous table shows some situations where the Publishing Interval and Sampling Interval are very low (100 ms). Note that this parameterization causes a high CPU consumption.