OPC UA
Modbus Server

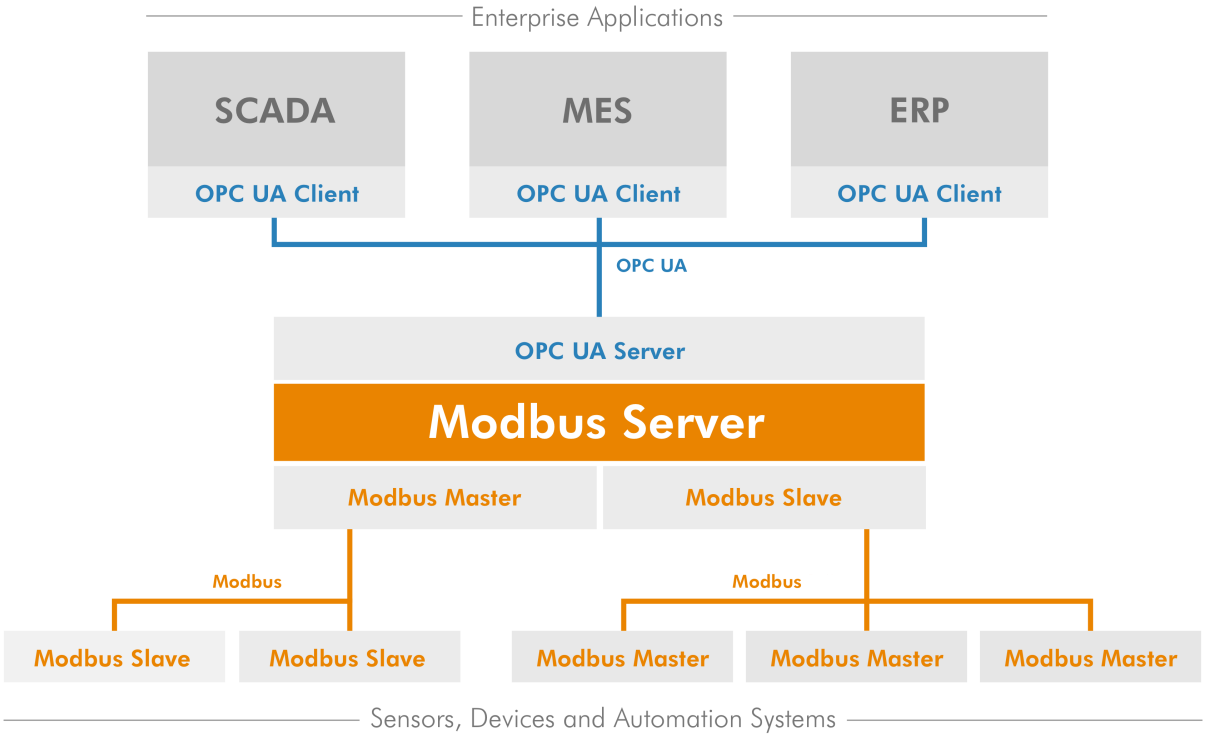
User Manual

Version 1.4

Table of Contents

Introduction

Prosys OPC UA Modbus Server is an application, which connects to Modbus devices and converts Modbus communication to OPC UA communication. It can be used to provide real-time sensor, device and automation system data from different Modbus devices to OPC UA client applications - or the other way around - in a generic way. Information security comes built-in with the OPC UA communications, making Prosys OPC UA Modbus Server a secure gateway to all your existing Modbus devices.



Prosys OPC UA Modbus Server can connect to any number of Modbus devices as a Modbus Master. You can also enable the Modbus Slave functionality, in order to let other Modbus Master devices to access data in the Prosys OPC UA Modbus Server. All defined variables (in Modbus Devices and in the internal Modbus Slave) are exposed via an OPC UA server to external OPC UA clients.

Prosys OPC UA Modbus Server supports both TCP/IP (Modbus TCP and Modbus RTU over TCP) and serial (RTU and ASCII) options for Modbus communication.

Should you consider opening communication to any publicly available network, consider disabling None Security Mode to disable insecure connections. Please, refer to Section 5.5, “Endpoints View” for details about this.

1. Installation

1.1. Choosing the Installation

Prosys OPC UA Modbus Server is available for download at http://www.prosysopc.com upon request. After a successful request, you are directed to the product download page, from which you can select the correct installation package for your target environment.

You have three alternative ways to install the application:

- Section 1.2, "Standard Installation" from an installation package (available for Windows and Linux). The installation package includes a private Java Runtime Environment (JRE), which is installed for this application only. This is the easiest option as it includes everything you need to run the application. On the other hand, it requires **administrator privileges**.
- **Section 1.3, "Portable Installation"**, which is in practice a ZIP package that you can extract to any location in your computer. The Portable Installation can be used in any operating system that has Java Runtime Environment (JRE) already available. This option provides the most flexibility, but requires that you **pre-install a suitable JRE** that will then be shared among other Java applications as well. Also you will have to setup it yourself for unattended operation. On the other hand, you can install and run the application without any administrator privileges.
- **Section 1.4, "Docker Installation"**, allows you to run the Prosys OPC UA Modbus Server in a Docker container. All you need is a Docker Engine, which is available on almost every platform.

1.2. Standard Installation

If you are upgrading from a previous version (pre-1.4.0) of the application, you must first uninstall the previous version before installing the new version. See Section 2, "Uninstallation".

1.2.1. Standard Windows Installation

On Windows, run the installer executable prosys-opc-ua-modbus-server-windows-x64-x.y.z-b.exe and follow the instructions. By default, the application is installed in the folder *Program Files/ProsysOPC/Prosys OPC UA Modbus Server*.

The installer also has an option to make desktop shortcut for running the application in Configuration Mode (see Section 3, "Running the Application" for more about the modes).

Additionally a Windows service is created, which gives an option to run the application in unattended operation mode. The service is started automatically, if you accept that when closing the application in Configuration Mode.

1.2.2. Standard Linux Installation

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

On Linux, first open the terminal and navigate to the directory of the downloaded *.sh* file. Then add a file permission to make the installation shell script executable with the command

```
sudo chmod u=x prosys-opc-ua-modbus-server-linux-x.y.z-b.sh
```

Then run the installation shell script with the command

```
sudo ./prosys-opc-ua-modbus-server-linux-x.y.z-b.sh
```

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

The installer also installs the application as a Linux service, making it available in unattended operation. If you do not have a graphical user interface available, this unattended operation mode is the only one you can run (which means you have to create the configuration elsewhere and transfer it).

1.3. Portable Installation

If the Standard Installation options described above are not suitable for your environment, you can install the application from the Portable Installation, which is available as a ZIP package. You can extract the package into any folder of your choice and run the application from there.

The pre-requisite is that you have Java 11 Runtime already installed into the operating system and be available from the PATH.

The Portable Installation enables installation with standard user privileges, since you can extract it also in a user directory.

1.4. Docker Installation

If you don't have a Docker Engine installed, you need to do that first. Please, follow the official Docker Installation instructions.

If you are running the Docker on Windows, you need to use Linux containers.

Next you need to download the ZIP package, which includes the Docker image and the configuration files for the Modbus Server.

Extract the ZIP file to a location of your choice. In the next steps (Section 3.3, "Running the Docker Application") the path to this location will be referred to as <installation_folder>.

Be aware that <installation_folder> will be used as a mounting point to the Docker container. The Prosys OPC UA Modbus Server that runs inside the container will find its configuration files, respectively, under the folder <installation_folder>/conf.

2. Uninstallation

Uninstallation depends on the way you installed the application. If you used the operating system specific installer, you can use the respective installation system to uninstall it.

The uninstallation will not remove the settings and log files that were created. See Section 7.1, "Configuration Folder" for details about file locations.

2.1. Standard Windows Uninstallation

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

2.2. Standard Linux Uninstallation

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

2.3. Portable Uninstallation

If you installed the Portable version of the application from the ZIP package, you can simply remove the installation directory with all sub-directories.

All the settings and log files are stored inside the installation directory, so they will be removed as well in this case.

2.4. Docker Uninstallation

On Linux distributions, please follow the official Docker Uninstallation instructions (CentOS, Debian, Fedora and Ubuntu).

On Windows the Docker Desktop can be uninstalled through the Control Panel.

On Mac the Docker Desktop can be uninstalled from the Docker menu by selecting **Troubleshoot** and then **Uninstall**.

3. Running the Application

You can run the application in two alternative modes:

- Configuration Mode
- Server Mode

Configuration Mode provides a user interface for configuring the Modbus device and data connections. In the Server Mode, the application is run in the background. The OPC UA server is available in both modes for OPC UA client connections.

If you installed the application from a Standard Installation, it should have been installed also as a Service application into the operating system. The service application will run the application in the Server Mode and enables it to start automatically when the operating system starts.

You cannot run the application in both modes at the same time. In Standard Installation the application will stop the Server Mode automatically, if you run it in Configuration Mode. In Portable Installation you must stop the Server Mode yourself, if you wish to run it in Configuration Mode.

3.1. Running the Standard Application

To prepare the necessary Modbus device and data configuration, run the application in Configuration Mode.

On Windows, you can start the application in Configuration Mode from the Desktop shortcut or Windows Start Menu.

On Linux, you can start the application from the Start Menu of your Window Manager. Alternatively, you can start it from the command line as:

```
sudo /opt/prosys-opc-ua-modbus-server/prosys-opc-ua-modbus-server --config
```

In the Standard Installation the application requires Administrator privileges, so you have to run it with sudo.

When you close the application in Configuration Mode, it will ask if you wish to start it in the Server Mode automatically. If you answer yes, the respective Service application will be started and it will also start automatically at system start up.

Alternatively, you can use the Service Manager of your operating system to start and stop the application in Server Mode. The name of the service is 'prosys-opc-ua-modbus-server'.

3.2. Running the Portable Application

If you installed the application from the Portable Installation, you can use the start-up scripts that are available in the bin folder under the installation folder.

- Use modbusserver-config.bat or modbusserver-config.sh to start the application in **Configuration Mode** in Windows or Unix-based systems, respectively.

- Use modbusserver.bat or modbusserver.sh to start the application in **Server Mode** in Windows or Unix-based systems, respectively.

The Configuration Mode uses JavaFX for the graphical user interface. JavaFX runtime is included in the distribution, but it works only on mainstream operating systems, such as standard Windows, Linux and macOS. The Java used to run the application should not contain a JavaFX runtime, and typically they do not.

You can run the Portable Application without any administrator privileges also in the Server Mode.

If you wish to make the application to start automatically in unattended mode, you must configure it in the system start up. For example by adding <installation_folder>/bin/modbus-server.sh into /etc/rc.local (or /etc/rc.d/rc.local) in Linux systems.

3.3. Running the Docker Application

You can run the Docker Application after you have performed the Section 1.4, “Docker Installation”.

The Docker Application can only be run in Server Mode, which means you need to setup the configuration with the Standard or Portable Installation (in Configuration Mode). The easiest is to extract the Portable Installation to the same folder as the Docker Installation and run bin/modbusserver-config.bat or bin/modbusserver-config.sh. Then you can use the same configuration in both of them directly. Otherwise you will have to copy the configuration between the different installations. See Section 7.1, “Configuration Folder” about the configuration options for installations without a graphical user interface.

You can use the start-up script that is available under the <installation_folder>/bin. The script can perform five different actions:

- start: Start the Modbus Server in the Docker container. Loads the image first, if it is not loaded yet.
- stop: Stop and remove the Docker container.
- load: Load the Docker image.
- unload: Unload the Docker image. Stops the container first.
- status: Current status of the Docker image, Docker container and OPC UA server.

Use modbusserver-docker.ps1 <parameter> or modbusserver-docker.sh <parameter> to perform the action in the Docker Application in Windows or Unix-based systems, respectively.

The Docker requires Administrator privileges in Linux, so you have to run the commands with sudo.

Useful commands to operate Docker:

- See all the loaded Docker images, use docker images.
- See all the running Docker containers, use docker ps.

3.3.1. Running the Docker Application manually

To load the Docker image

```
docker load -i <installation_folder>/docker-image/prosys-opc-ua-modbus-server-X.X.X-X-image.tar
```

And then you can run the Docker image as a container:

On Unix-based systems use

```
docker run --publish 53510:53510 --mount type=volume,dst=/data/,volume-driver=local,volume-opt=type=none,volume-opt=o=bind,volume-opt=device=<installation_folder> -e "JAVA_TOOL_OPTIONS=-Dapp.home=/data/ -Dlog4j.configurationFile=/data/conf/log4j2.xml -Dloggingoutputpath=/data/log/" -h $HOSTNAME --name modbus-server  prosys-opc-ua-modbus-server:X.X.X-X
```

On Windows use

```
docker run --publish 53510:53510 --mount type=volume,dst=/data/,volume-driver=local,volume-opt=type=none,volume-opt=o=bind,volume-opt=device=<installation_folder> -e "JAVA_TOOL_OPTIONS=-Dapp.home=/data/ -Dlog4j.configurationFile=/data/conf/log4j2.xml -Dloggingoutputpath=/data/log/" -h %COMPUTERNAME% --name modbus-server  prosys-opc-ua-modbus-server:X.X.X-X
```

This maps the <installation_folder> into /data folder inside the container, and also maps the access to the configuration files, respectively.

You can modify following docker run options:

- --publish <UA_TCP_PORT>:<UA_TCP_PORT>, The UA TCP port must correspond to the UA TCP port in the Section 5.5, “Endpoints View”.
- In addition you can also use the standard Docker options.

4. Application License

Prosys OPC UA Modbus Server is a commercial application and requires a license to be purchased for continuous operation. The application may be evaluated in Demonstration Mode for 60 days to validate the feasibility for the installation in question. In Demonstration Mode, it will run 120 minutes, after which it will be stopped. The commercial license is required to let the application run continuously.

The Application is licensed per installation and tied to the hostname of the computer in which it is being run.

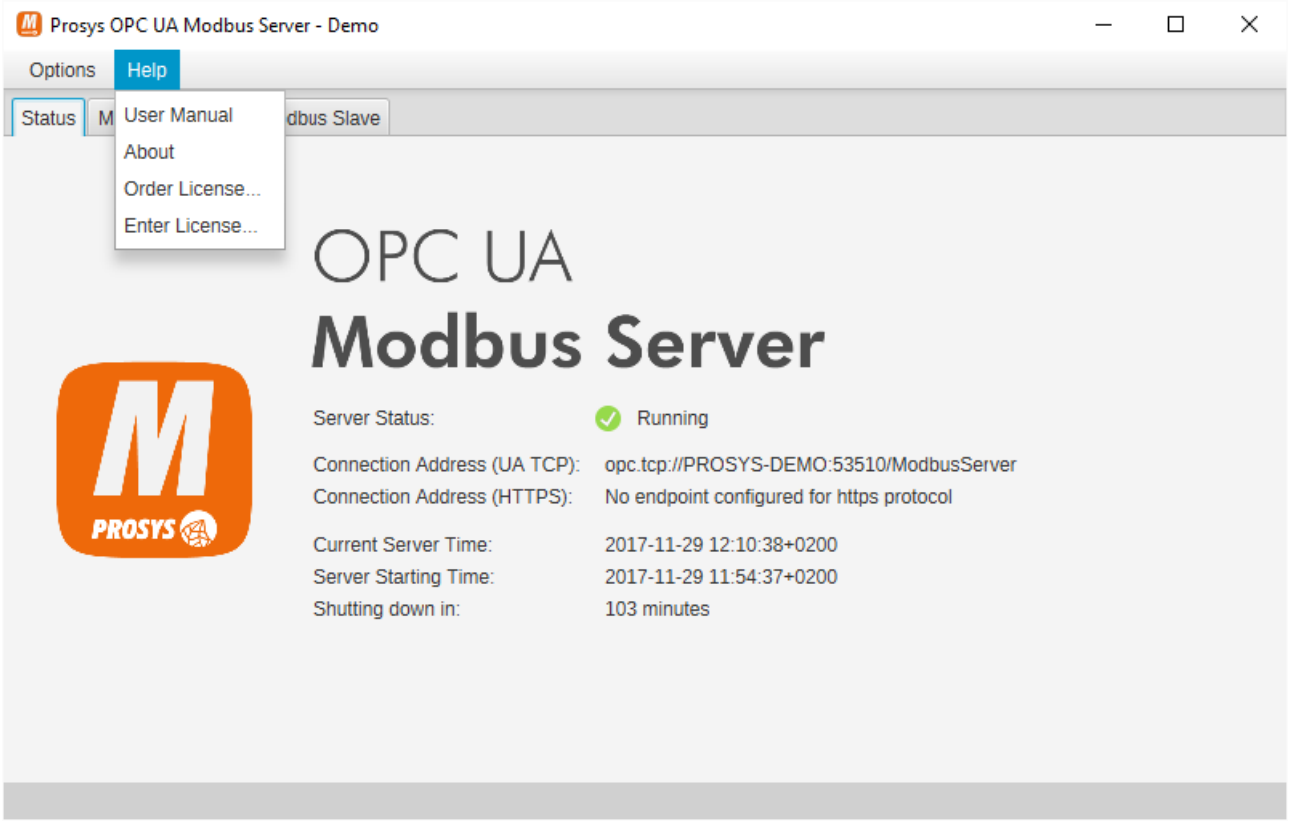


Figure 1. Order License and Enter License are available from the Help menu.

When you are ready to order the license, you will need to find the hostname of the target computer and the name of the Licensee (Person or Organization). The best way to ensure that the information is correct is to use the Order License Dialog that is available from the Help menu (Figure 1, “Order License and Enter License are available from the Help menu.”). You can generate a License Request File from the dialog that opens (Figure 2, “Order License Dialog.”).

If you are using the Portable Installation, make sure that you record the correct hostname of the target computer. The license cannot be changed to a new hostname later on.

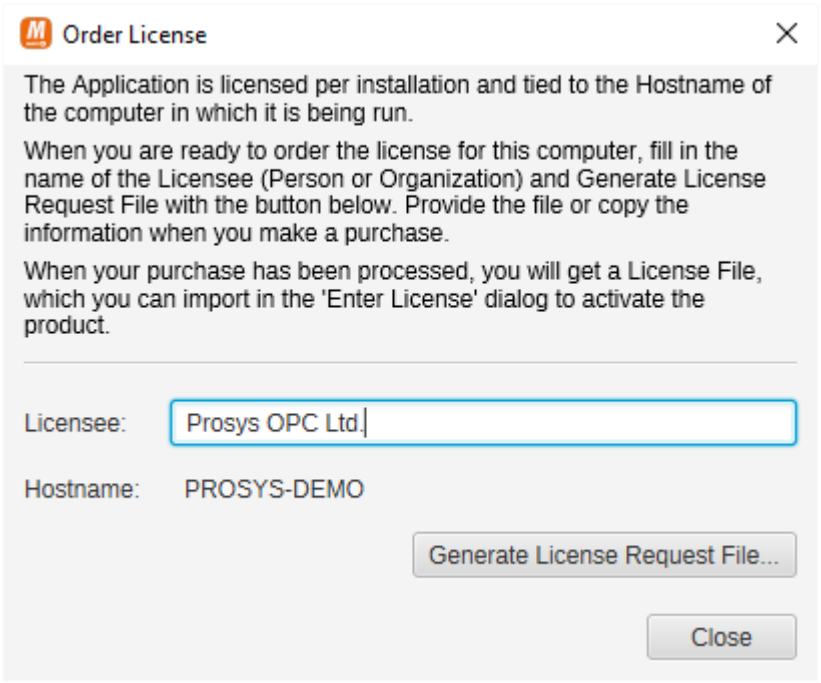


Figure 2. Order License Dialog.

Send the License Request File or the respective information to sales@prosysopc.com when you make a purchase.

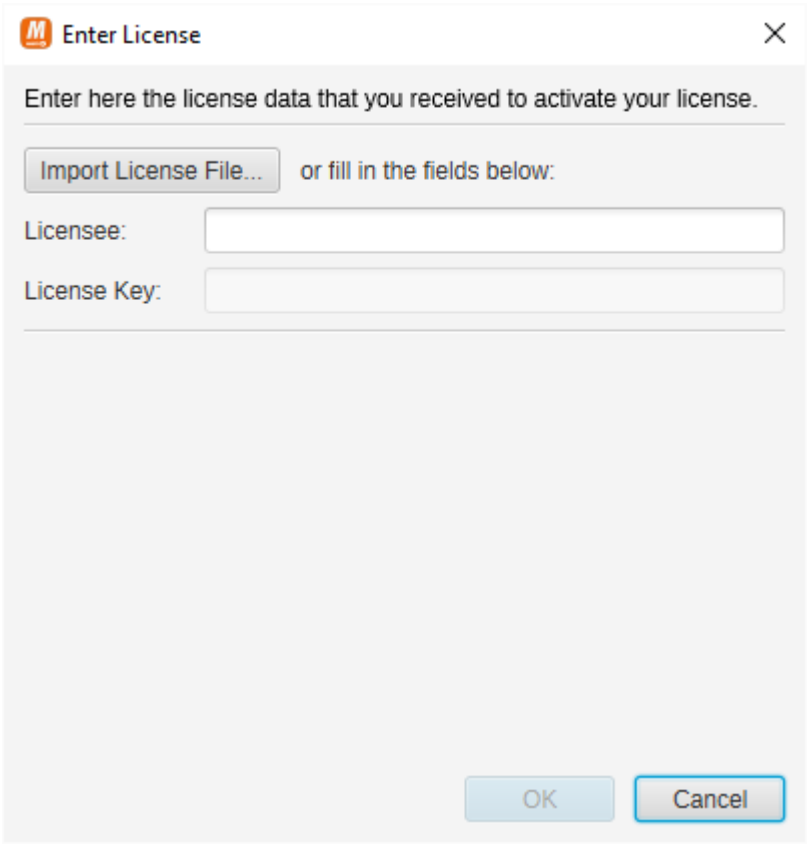


Figure 3. Enter License Dialog.

When your purchase has been processed, you will get a License File, which you can import in the Enter License Dialog (Figure 3, "Enter License Dialog.", also available from the Help menu) to activate the product.

If you are using the Portable Installation, you must copy the License File manually as instructed in Section 7.3, “License File”. After copying, run the application once and validate that the license is correctly in place from the console output.

5. Configuration Mode

When you run the application in the Configuration Mode, you can define the OPC UA and Modbus connections.

The Configuration Mode requires graphical user interface from the computer where it is run.

In Configuration Mode, you can access different main views that provide different parts of the configuration.

The default main views are:

- Status
- Modbus Devices
- Modbus Slave

You can also enable more configuration options by selecting the Expert Mode from the Options menu.

The Expert Mode contains the following additional main views:

- Endpoints
- Users
- Sessions
- Certificates
- Address Space

5.1. Status View

The *Status View* (Figure 4, “The Status View displays current status information about the OPC UA server.”) displays information about the current Status of the integrated OPC UA server of the application. If everything is fine, the *Server Status* is “Running”. During startup, the status will show "Initializing" with specific information about the exact phase. In case of errors, it may turn to "Failed" with additional information about the exact problem. The application can fail at start up, for example when another instance of the application is already running (either in Configuration Mode or Server Mode).

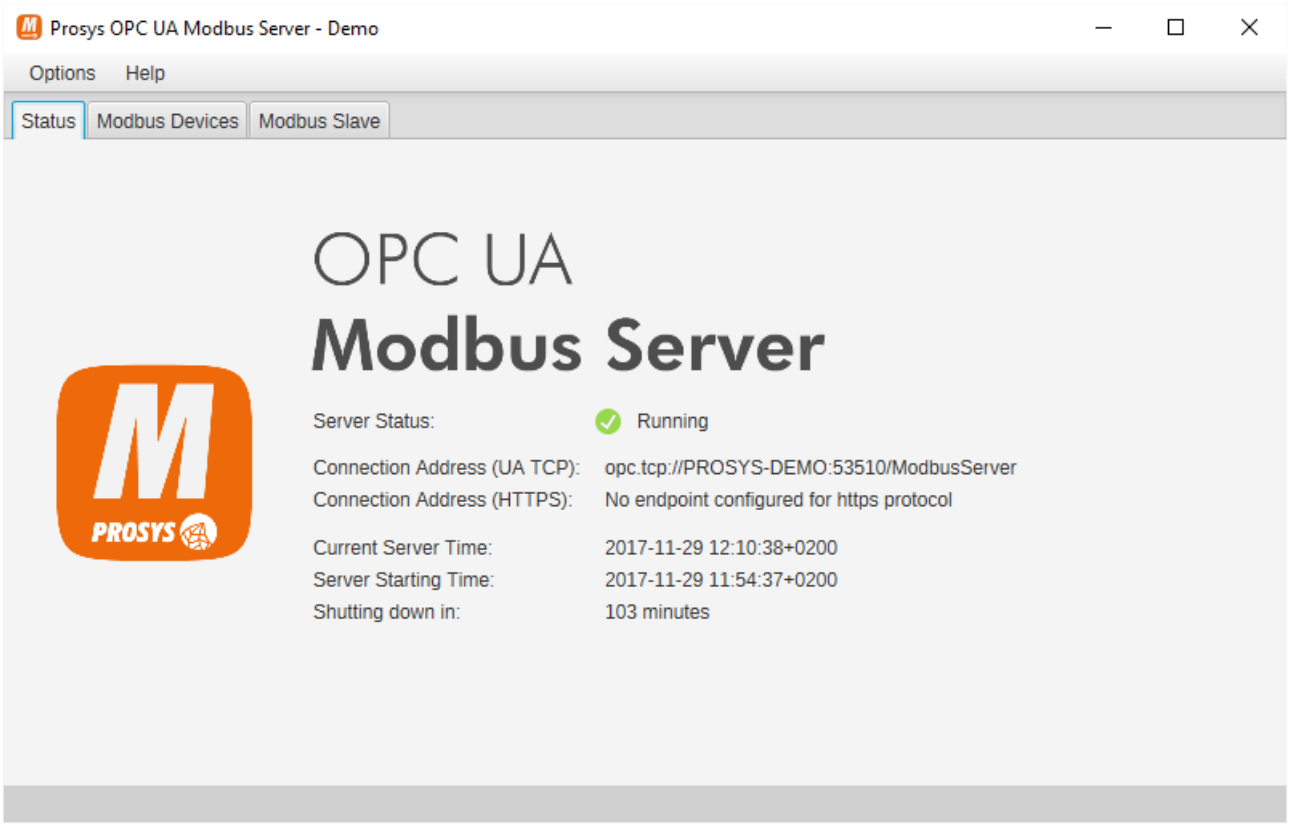


Figure 4. The Status View displays current status information about the OPC UA server.

The Status View also displays the current and starting time of the server. If the server is connected by OPC UA client applications over network, you should ensure that the computers are running with synchronized clocks. Especially the secure connections will not work if the clocks between the computers are too much out of sync. The status and time values are also available to OPC UA client applications through the *ServerStatus* Variable of the OPC UA *Server* Object (see Section 6, “OPC UA Server”).

The Connection Addresses show the OPC UA addresses, which the OPC UA client applications can use to connect to the OPC UA server.

Prosys OPC UA Modbus Server supports both UA TCP and UA HTTPS, which are defined as alternative *Transport Protocols* in the OPC UA specification. Most client applications will use UA TCP, but UA HTTPS may provide an alternative for some specific client applications. You can define the available addresses in the Section 5.5, “Endpoints View”.

5.2. Modbus Devices View

The *Modbus Devices* view (Figure 5, "Modbus Devices view showing the device configuration information.") is used to configure the external Modbus devices that the application will be connecting to as a Modbus master.

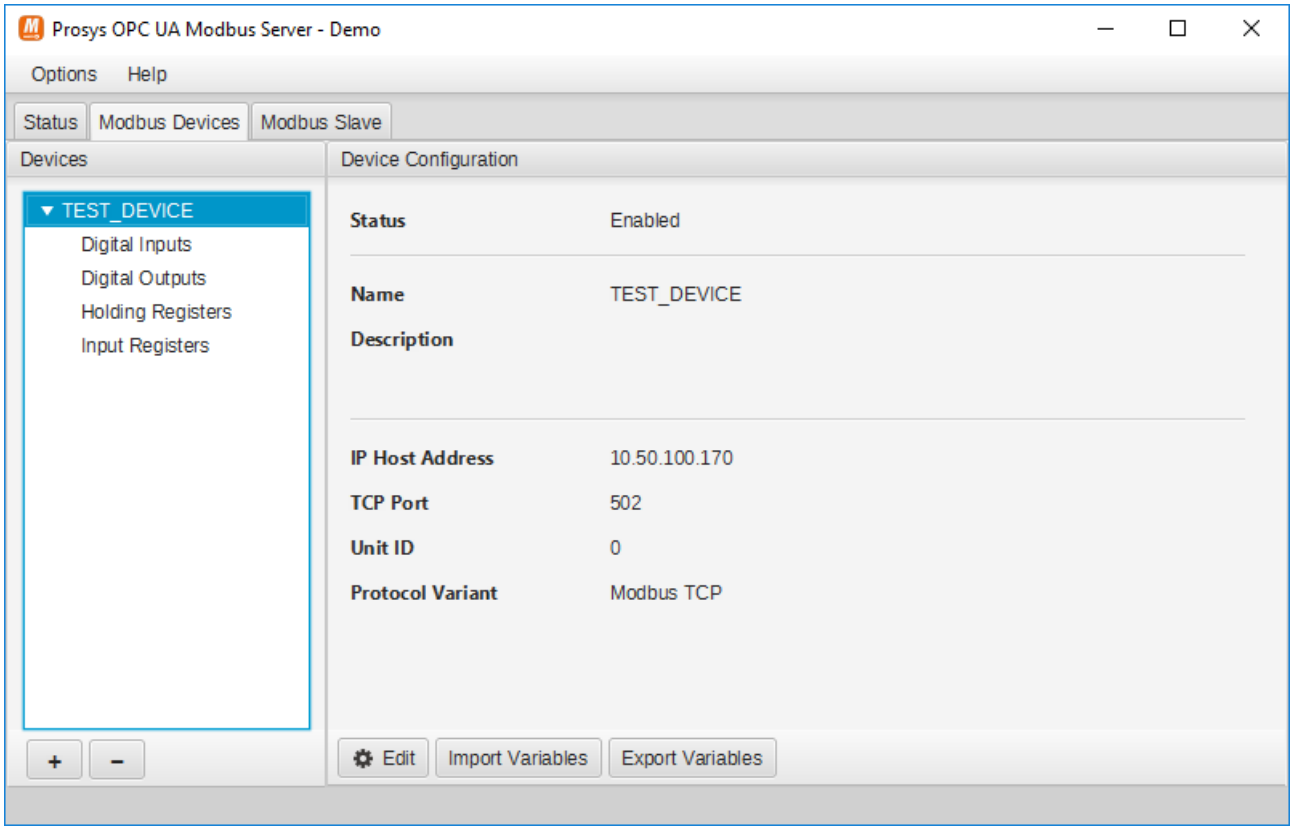


Figure 5. Modbus Devices view showing the device configuration information.

5.2.1. Device Configuration

Use the + button at the bottom of the *Devices* list (on the left), to add a new external Modbus device. Use the - button to remove the currently selected device from the list.

To see the configuration of a specific Modbus device, select it from the *Devices* list. The current device configuration information is displayed on the panel at the right hand side. To change the configuration, click Edit at the bottom of the view to open the *Device Configuration* pop-up window.

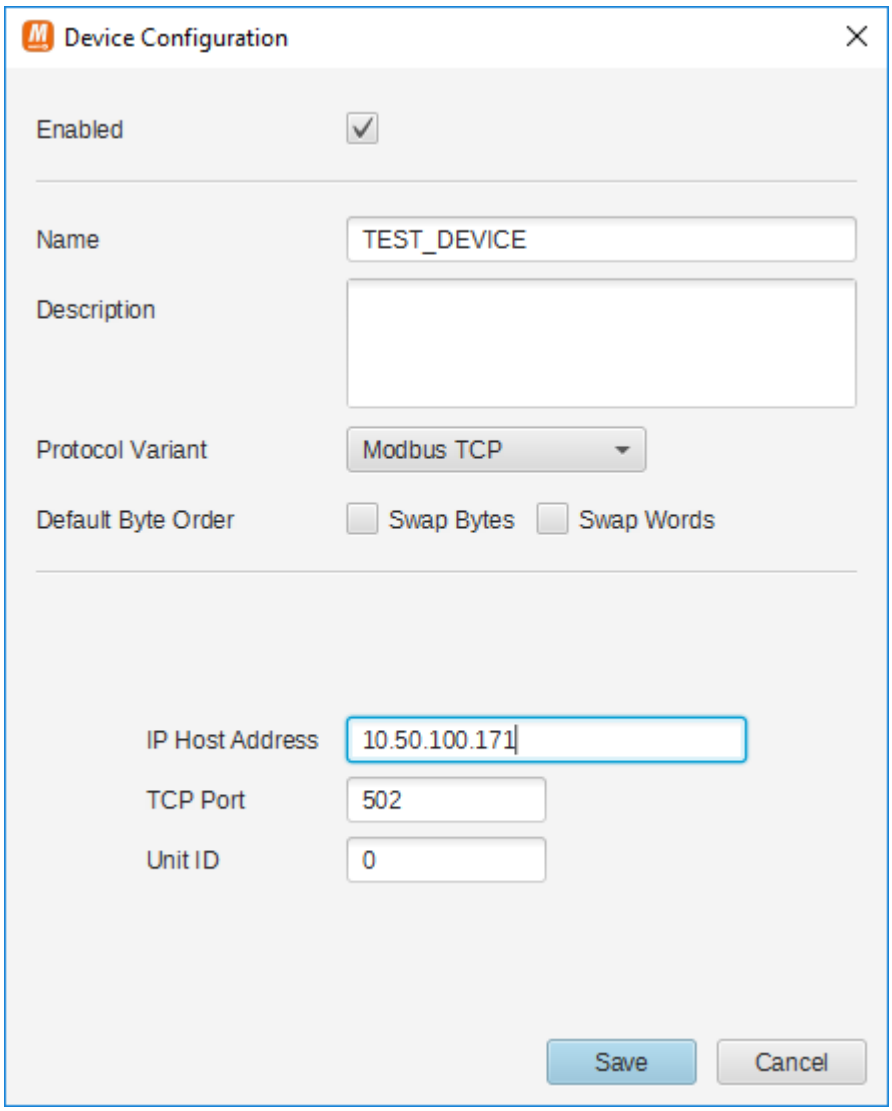


Figure 6. Device Configuration pop-up window.

The device configuration consists of the following parameters:

Table 1. Modbus Device Parameters

Parameter	Description
Enabled	You can enable or disable communication to the device temporarily without removing it from the configuration. Only devices that are enabled will show up in the OPC UA server address space (Section 6.1, "OPC UA Server Address Space").
Name	The Name of the device is used to identify the device in the OPC UA server address space. The name is used to define the NodeId, BrowseName and DisplayName attributes of the respective OPC UA object. It is also used for all NodeIds of the variables of the device. NOTE: If you change the name of the device, it will also affect all its variables by changing the identifiers (NodeIds) in the OPC UA address space.
Description	The description parameter can provide more information about the device in addition to the Name. The Description is available as the Description attribute of the respective OPC UA object.

Unit ID	Unit ID of the device. This is usually 0, but for example if you are using Modbus RTU over TCP, the Unit ID is the RTU node address of the target device. Also in some rare cases, devices use a different Unit ID with Modbus TCP, in which case you will need to configure the correct ID here.
Protocol Variant	Modbus protocol variant used for the device: Modbus TCP, Modbus RTU over TCP, Modbus RTU or Modbus ASCII. Modbus TCP and Modbus RTU over TCP utilise TCP/IP communication while Modbus RTU and Modbus ASCII are used for serial port communication.
Default Bit Order	Defines the default values for the data format settings of all new variables added to the device. It covers two options that are described in detail below.
Swap Bytes	Defines that the order of the two bytes inside a register should be swapped from the default Modbus interpretation (only applies to data types with a length of 16 bits or more). Default mode interprets the first byte as the most significant byte (MSB first). Setting Swap Bytes to true will change the mode to interpret the first byte as the least significant byte (LSB first).
Swap Words	Defines that the order of two words in two subsequent registers should be swapped from the default Modbus interpretation (only applies to data types with a length of 32 bits or more). Default mode interprets the first word as the most significant (high).

There are also a number of parameters that are specific to the selected transfer protocol: either TCP/IP (Modbus TCP and Modbus RTU over TCP) or serial port (Modbus RTU and Modbus ASCII). These parameters are described below.

Table 2. Device parameters specific to TCP/IP communication

IP Host Address	The IP address or hostname of the Modbus device to connect to.
TCP Port	The TCP port number of the device. The standard port number for Modbus protocol is 502, so that will be used by default.

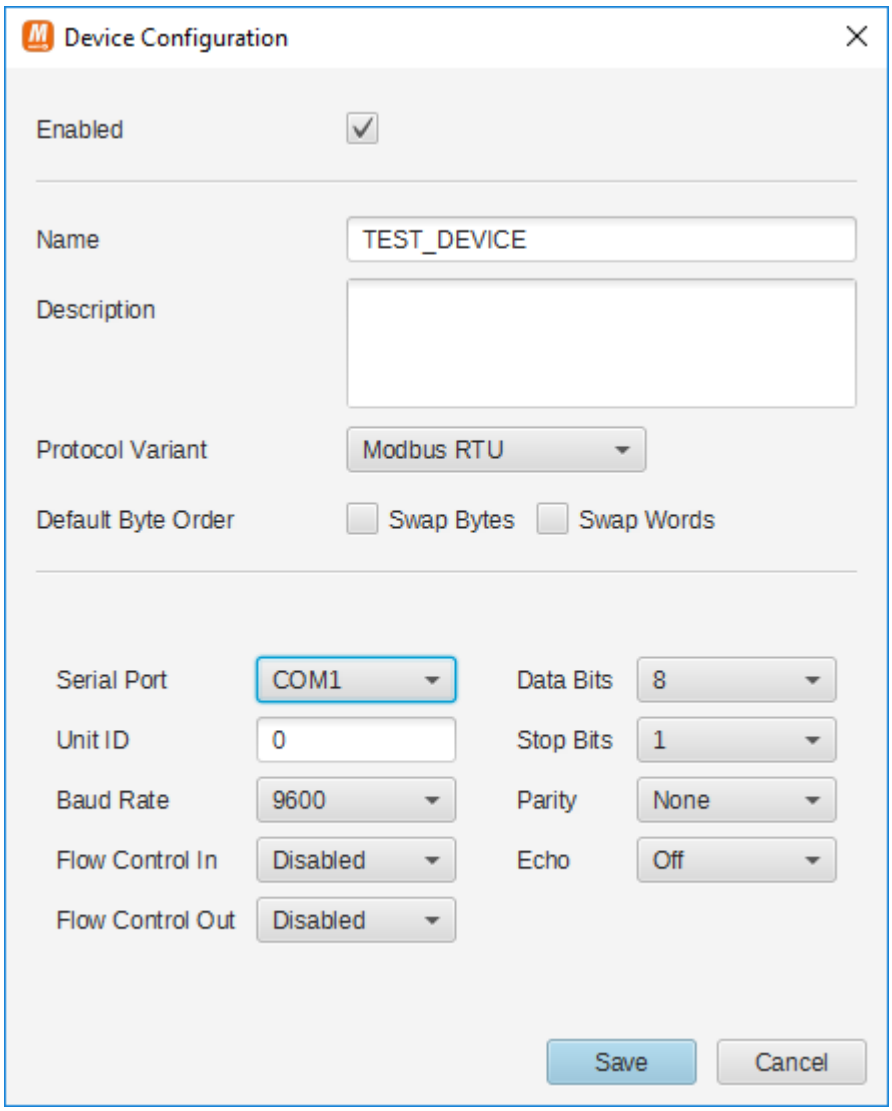


Figure 7. Device Configuration window for serial port communication.

Table 3. Device parameters specific to serial port communication

Serial Port	Name of the serial port that the Modbus device is connected to.
Baud Rate	Serial port speed, i.e., bit rate (bits/s).
Flow Control In	Inward data flow control mode.
Flow Control Out	Outward data flow control mode.
Data Bits	The number of data bits in each character.
Stop Bits	The number of stop bits sent at the end of every character.
Parity	Parity bit mode used for error-checking with every character.
Echo	Enable/disable echoing.

5.2.2. Right-click Functions for Devices List View

By right-clicking a device in the *Devices* list, you can access the menu with a set of convenient functionalities for managing the Modbus devices. The available functions are listed below with short descriptions:

- Clear all variables, removes all existing variables from the device.
- Delete device, deletes the device and all its variables.
- Duplicate device, creates a new device that has the same device configuration and the same variables as the selected device.
- Disable device or Enable device, disables or enables the Modbus device. Disabling a device means that the it is disconnected and removed from the address space of the OPC UA server.
- Edit device..., opens the *Device Configuration* pop-up window for changing device settings.

5.2.3. Modbus Tables

Modbus devices use four primary tables to store data: *Digital Inputs*, *Digital Outputs*, *Input Registers* and *Holding Registers*. Digital Inputs and Digital Outputs contain *discrete* values (single bit) in contrast to the other ones containing *registers* (16 bit words). Digital Inputs and Input Registers are read-only, whereas Digital Outputs and Holding Registers can also be written to.

Modbus devices are free to use the different tables to access the same physical memory areas in the device. So the same data may be available from different tables.

Each table can hold a maximum of 65535 addresses, although the actual range of valid addresses depends on the device. Each address in the discrete tables contains 1 bit of information, whereas each address in the register tables contains 1 word = 2 bytes = 16 bits of information.

Prosys OPC UA Modbus Server requires that you predefine the variables that will be available from the OPC UA server for each device. Each variable will map to a certain Modbus address and data type. Each variable defined for a discrete table will have its DataType defined as Boolean. For a register variable, you can choose from a range of different data types.

In the *Modbus Devices* view, you will find the tables under each device in the *Devices* list on the left. When you select a table from the *Devices* list, you will also see the respective variables defined for the table.

5.2.4. Data Types

Digital Input and Digital Output tables of the Modbus protocol define a 1-bit value for each address inside the tables. These values are always interpreted as BIT data type, which corresponds to the Boolean DataType in OPC UA.

On the other hand, Holding Register and Input Register tables hold 16 bits (= 2 bytes) of data in each each address. It is also possible to combine subsequent addresses to provide 32 bits (= 4 bytes) or 64 bits (= 8 bytes) of data. These bits can be interpreted in various ways to define variables whose values correspond to various different OPC UA DataTypes.

The Prosys OPC UA Modbus Server supports the interpretation of values in the Modbus tables

according to various IEC 61131-3 data types that are then mapped to their counterpart OPC UA DataTypes in the address space of the Modbus OPC UA Server. The currently supported data types for registers are listed in the table below.

Table 4. Supported Data Types

Modbus data type	Number of bits	Number of registers	OPC UA DataType	Description
BIT	1	1/16	Boolean	Binary value
SINT	8	1/2	SByte	Short integer number
USINT	8	1/2	Byte	Unsigned short integer number
INT	16	1	Int16	Integer number
UINT	16	1	UInt16	Unsigned integer number
DINT	32	2	Int32	Double integer number
UDINT	32	2	UInt32	Unsigned double integer number
LINT	64	4	Int64	Long integer number
ULINT	64	4	UInt64	Unsigned long integer number
REAL	32	2	Float	Floating-point decimal number
LREAL	64	4	Double	Double-precision floating-point decimal number
CHAR	8	1/2	String	ISO 8859-1 encoded text
WCHAR	16	1	String	UTF-16 encoded text

USINT, UINT and UDINT, ULINT are unsigned types, whereas SINT, INT, DINT and LINT are their counterpart signed data types (may also contain negative values).

5.2.5. Variable Configuration

Variable configuration functionality can be accessed by clicking any table of any device in the *Devices* list. This will open the *Variables* view for the specific table on the right-hand side of the application window. To add or remove variables from a table, use the + and - buttons, respectively. (See Figure 8, “*Variables* view.”)

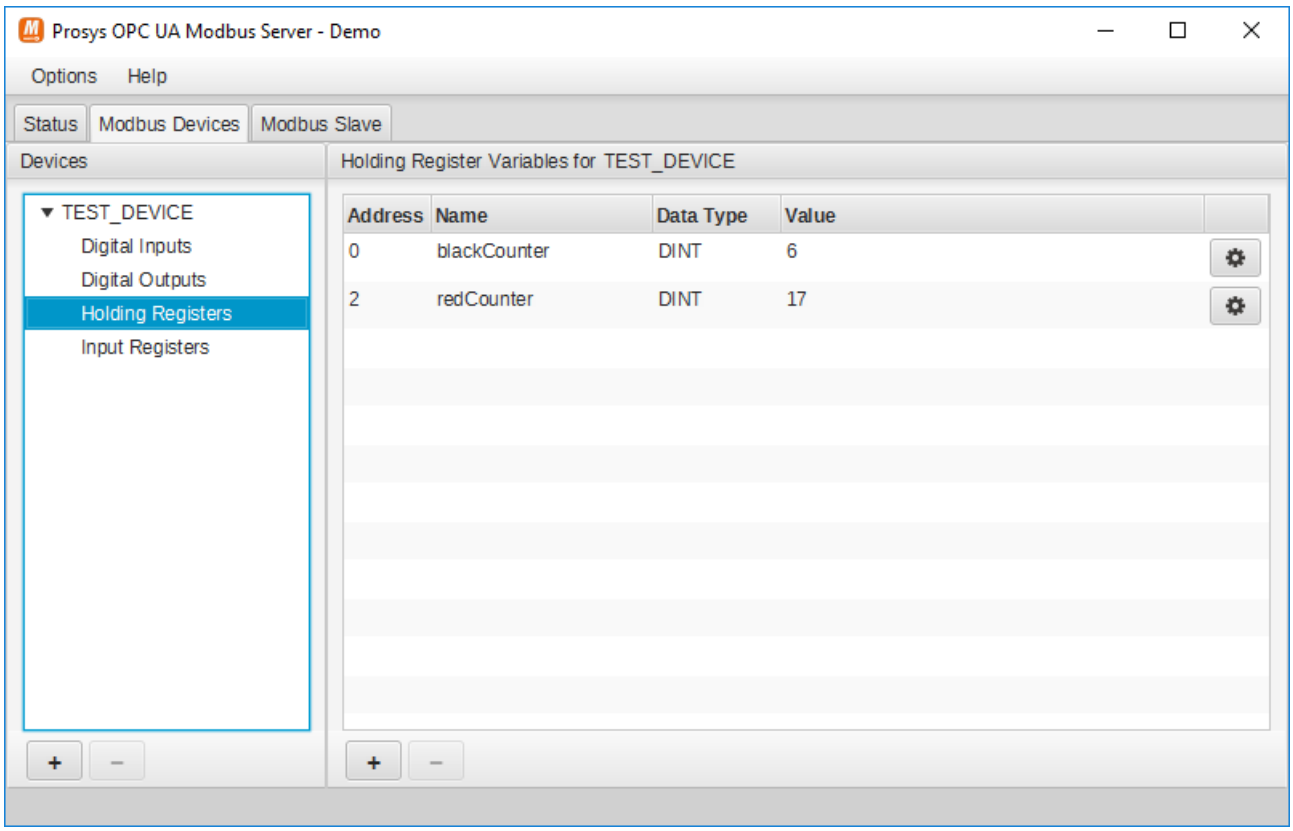


Figure 8. Variables view.

The variables are used to define the OPC UA model for your device. Each variable is mapped to one or more Modbus addresses inside the chosen Modbus table. For register variables, you can also define the data type and an optional scaling configuration to provide the data in the desired format, even in case the original measurement values in the device are not.

For each variable you can configure the following properties:

Table 5. Variable properties

Property	Description
Address	The starting address of the variable inside the Modbus table. The address range starts from 0 for each table and ends in address 65535 (in contrast to the Modbus Coil/Register numbering scheme that starts from 1 for Digital Outputs, 10001 for Digital Inputs, etc.)
Name	Name to identify the variable. The name is also used for the BrowseName and DisplayName of the OPC UA Variable, but not for the NodeId. The NodeId will comprise of the Device Name, the address and the bit offset inside the address, for example 'DEMO:HoldingRegisters:0:0'.
Description	Additional description to be used for the OPC UA Variable.
Array Length	Defines the number of sequential Modbus data items of the chosen data type that will be presented as an array of the given length (value of 1 means that the variable is a scalar, i.e., not an array). Maximum size for the data in an array is 2000 bits, which means that the maximum length of an array depends on the length of the data type. For string data types, the array length defines the maximum length of bytes (CHAR) or words (WCHAR) that the value can hold (this can be different from the actual number of characters in the string).

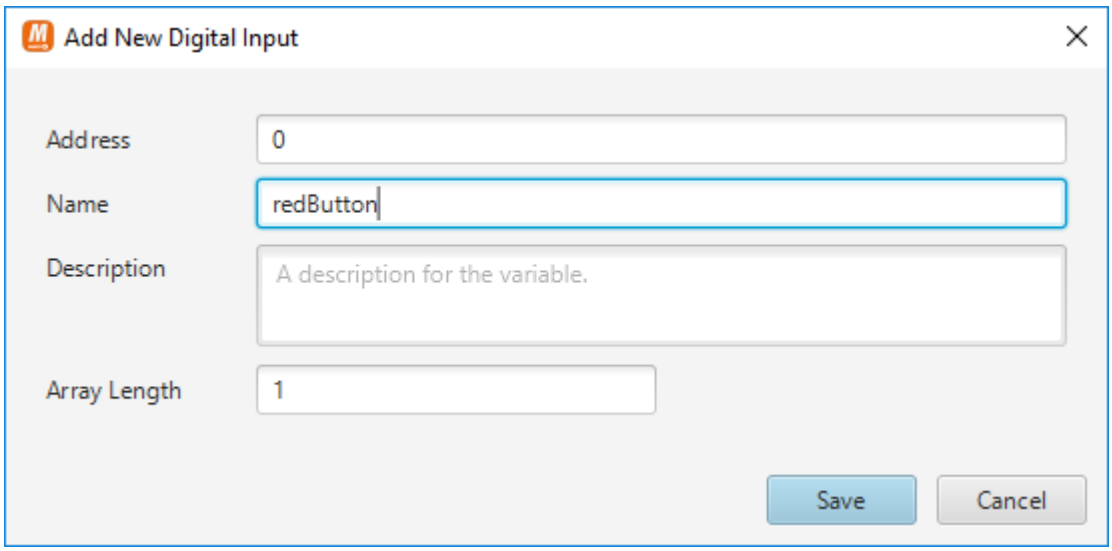


Figure 9. Adding a new discrete variable.

Additionally, the following properties are available for register variables (Holding Registers and Input Registers):

Property	Description
Data Type	Defines the IEC-61131-3 data type used to interpret the value of the register.
Bit Offset	Defines the bit offset (between 0 and 15) inside the specified register for the starting point of the variable (only applies to data types with a length of 8 bits or less). For example, a bit offset value of 3 means that the variable value starts from the 4th bit of the register.
Swap Bytes	Defines that the order of the 2 bytes in a register should be swapped from the default Modbus interpretation (only applies to data types with a length of 16 bits or more). The default mode interprets the first byte as the most significant byte (MSB first). Setting Swap Bytes to true will change the mode to interpret the first byte of a word as the least significant byte (LSB first).
Swap Words	Defines that the order of the 2 words in the 2 subsequent registers should be swapped from the default Modbus interpretation (only applies to data types with a length of 32 bits or more). Default mode interprets the first word as the most significant (high).
Scaling	You can configure a custom scaling for the values. Currently the alternatives are No Scaling and Linear Scaling. The formula for the Linear Scaling is Factor * Value + Offset. For arrays, the scaling applies to each element.

If Linear Scaling is applied, the OPC UA DataType for the scaled value will always be Float (for data types with length of 2 words or less) or Double (for data types with length of 4 words), independent of the unscaled Modbus data type. See more about the data types in the previous chapter (Section 5.2.4, “Data Types”).

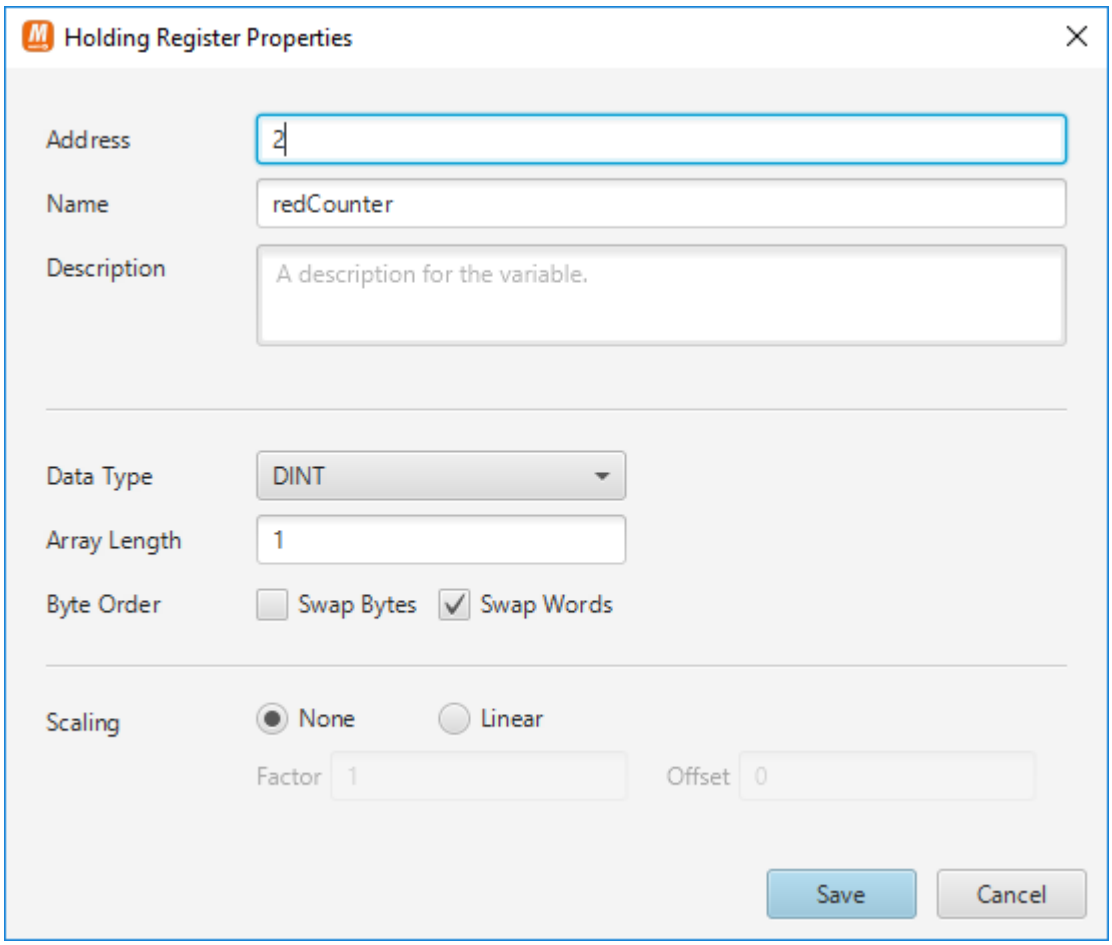


Figure 10. Configuring a register variable.

5.2.6. Keyboard Shortcuts for Variables View

A set of keyboard shortcuts are enabled to provide convenience to the editing of variables in the *Variables* view. The shortcuts include:

- CTRL + N, opens a pop-up for creating a new variable.
- CTRL + E, opens a pop-up for editing a selected existing variable.
- CTRL + A, selects all existing variables.
- DELETE, removes all selected variables.

5.2.7. Importing Variable Configurations from CSV File

It is also possible to import variables for a selected Modbus device by clicking the Import Variables button. You are then prompted to select a CSV file containing the variable configurations you wish to add. You can create the configuration file yourself or use a file that was created by the OPC UA Modbus Server, see Section 5.2.8, “Exporting Variable Configurations to CSV File”.

In case you have already defined variables for the device but still wish to import variable configurations from a CSV file, it is possible that the existing variables and the imported variables occupy the same Modbus addresses, which is not allowed. In this case, the two variables are conflicting. Therefore, when importing the variables you are prompted to choose an import mode that handles conflicting variables in the desired way. Three modes are available. They either remove all existing variables or, in case of

conflicting variables, give priority to either the existing variable or the new variable and ignore the other.

Depending on the chosen import mode, the importing of variables from a CSV file can remove all existing variables that were configured for the device before the import.

The CSV (comma-separated values) configuration file must conform to the format that is specified in the following table. You can, for example, use Excel (or any other spreadsheet tool allowing the export of CSV files) to create the desired variable configurations. The first row of the CSV must contain all the header fields (separated by commas) in the order that corresponds to the provided data. A CSV file containing a correct header format is also included in the installation location of the application, in the folder named "csv".

Table 6. Format used for the variable configurations CSV file

Header field name	Description	Allowed values	Mandatory
table	Modbus table where the variable should be added, see Section 5.2.3, "Modbus Tables"	di, do, ir, hr	yes
address	Address, see Table 5, "Variable properties"	0-65535	yes
dataType	Data Type, see Table 4, "Supported Data Types"	bit, sint, usint, int, uint, dint, udint, lint, ulint, real, lreal, char, wchar	yes
bitOffset	Bit Offset, see Table 5, "Variable properties"	0-15 (depends on data type)	no
arrayLength	Array Length, see Table 5, "Variable properties"	1-2000 (depends on data type)	no
name	Name, see Table 5, "Variable properties"	any text	no
description	Description, see Table 5, "Variable properties"	any text	no
swapBytes	Swap Bytes, see Table 5, "Variable properties"	true, false	no
swapWords	Swap Words, see Table 5, "Variable properties"	true, false	no
scalingOption	Scaling Option, see Table 5, "Variable properties"	none, linear	no
scalingFactor	Scaling Factor, see Table 5, "Variable properties"	any decimal or whole number	no
scalingOffset	Scaling Offset, see Table 5, "Variable properties"	any decimal or whole number	no

5.2.8. Exporting Variable Configurations to CSV File

You can also export all the variable configurations of a specific Modbus device by clicking the Export Variables button. The configuration is saved in a CSV file according to the format specified in Section 5.2.7, “Importing Variable Configurations from CSV File” to a folder that can be freely chosen by the user.

5.3. Modbus Slave View

The *Modbus Slave* view (Figure 11, "Modbus Slave view.") is used to configure an internal Modbus slave device that the Prosys OPC UA Modbus Server will host.

The Modbus slave is enabled by clicking the + button at the bottom of the panel on the left. This adds the slave into the configuration. You can then disable it from the Slave Configuration View (Available from Edit) or with the - button. Prosys OPC UA Modbus Server supports only one Modbus slave device.

In general, the configuration of the Modbus slave is similar, although more limited, to the configuration of the Modbus devices in the *Modbus Devices* view (Section 5.2, "Modbus Devices View"). For TCP/IP communication the slave is bound to the address 0.0.0.0, which means all local interfaces in practice.

You must define the variables that are available in the OPC UA address space for the slave - this will also define the valid Modbus addresses that can be accessed by Modbus master devices connected to the slave.

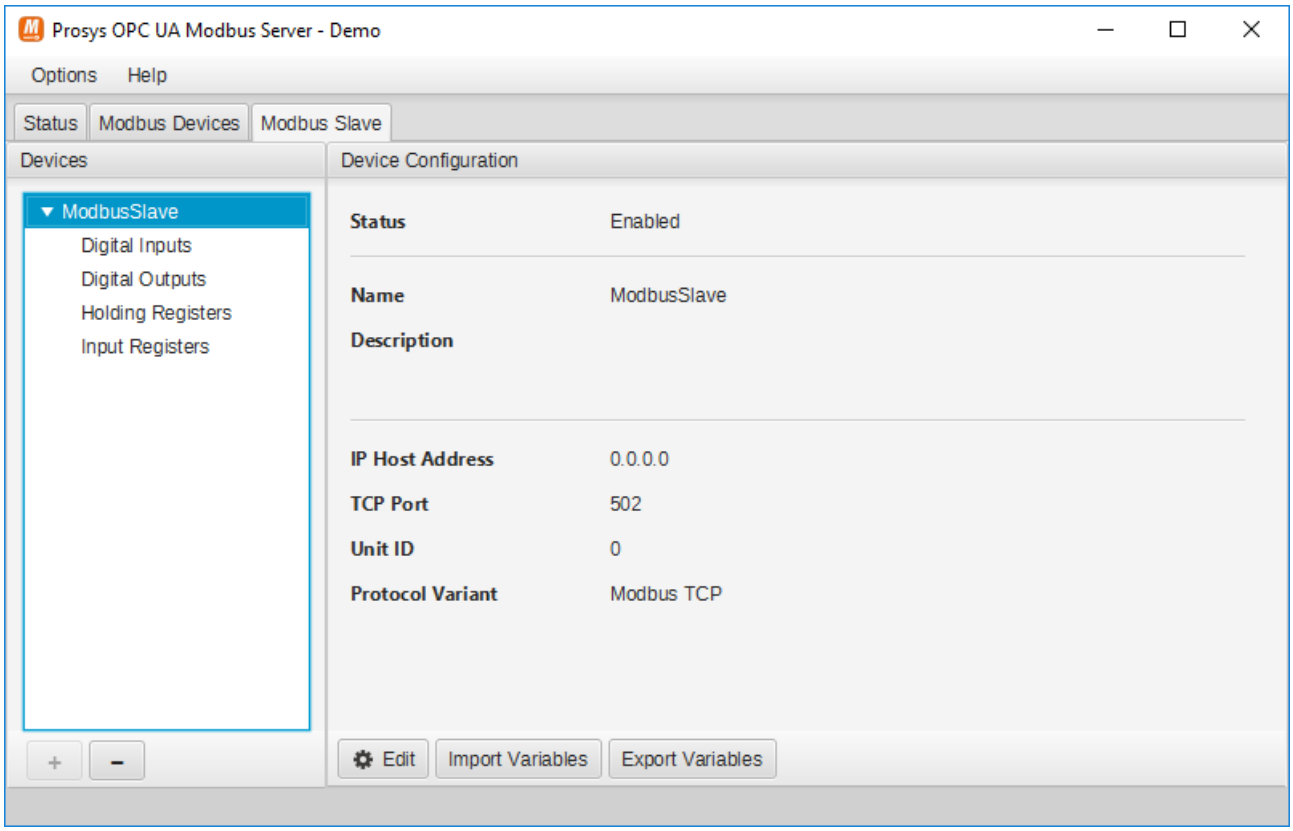


Figure 11. Modbus Slave view.

5.4. Expert Mode

By default, the *Configuration Mode* UI displays only the settings specific to configuring the Modbus devices. In order to change the OPC UA server configuration, you must enable the *Expert Mode* from the Options menu.

The Expert Mode makes the following additional views visible:

- Endpoints
- Users
- Sessions
- Certificates
- Address Space

5.5. Endpoints View

The *Endpoints View* is only available in the Expert Mode (Section 5.4, “Expert Mode”).

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

Should you consider opening communication to any publicly available network, consider disabling None from the *Security Modes* to disable insecure connections.

The Endpoints View (Figure 12, “The Endpoints View allows you to configure the connection addresses and security options available for OPC UA clients to connect to the OPC UA server.”) allows you to configure these settings and verify which endpoints the server is exposing.

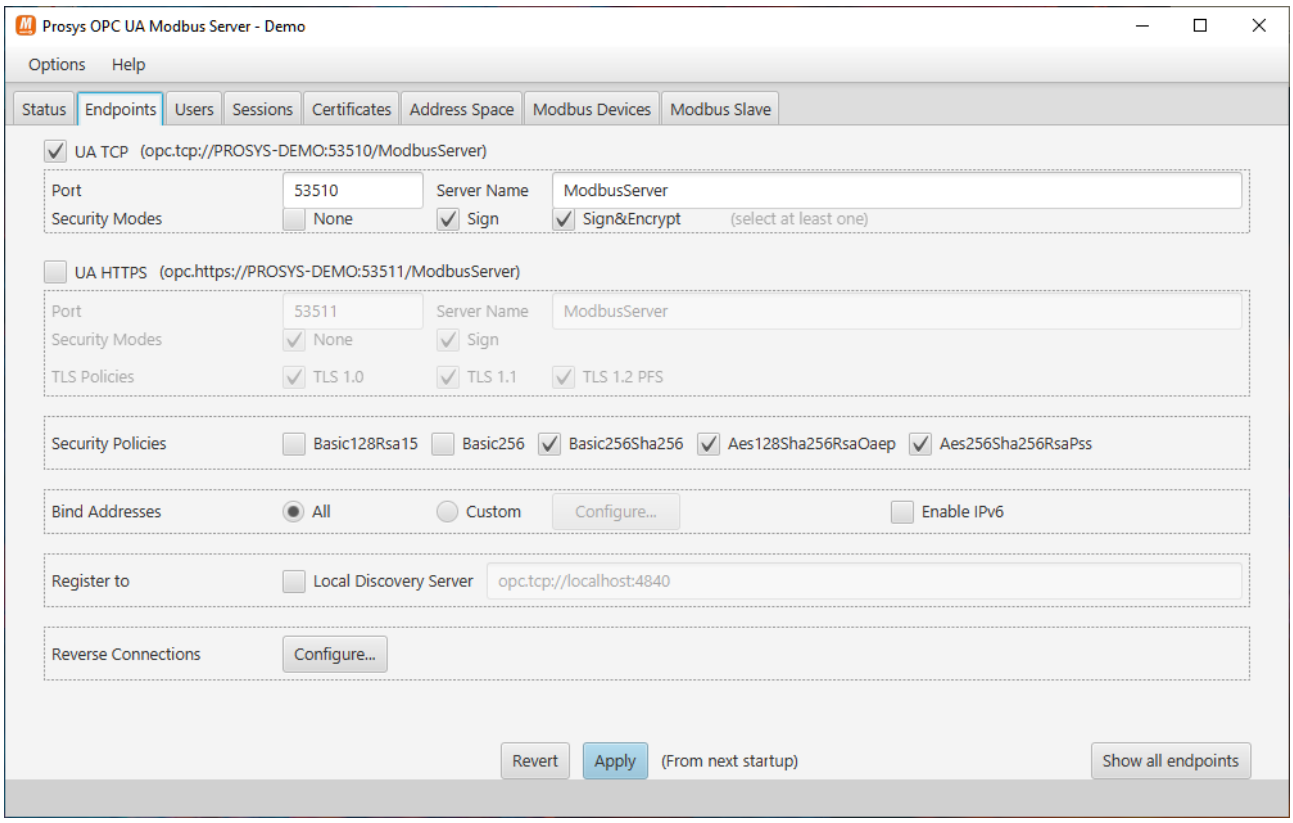


Figure 12. The Endpoints View allows you to configure the connection addresses and security options available for OPC UA clients to connect to the OPC UA server.

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

- UA TCP
- UA HTTPS

UA TCP is the usual transport protocol and it should be supported by all client applications. UA HTTPS is disabled by default, but you can enable easily from these settings, if you require that.

5.5.1. UA TCP Transport Protocol

UA TCP is an OPC UA specific binary communication, including full OPC UA specific security implementation. The *Port* and *Server Name* define the exact connection address, which is displayed at the top (opc.tcp://<hostname>:53510/ModbusServer).

In addition to the connection address, you can define the security modes that the server accepts. The client applications will decide which mode they wish to use, so the server can only configure which options are available.

Security Mode Sign will ensure that all traffic can be validated by the client and server application and may not be modified during transfer. Security Mode Sign&Encrypt will also make all communication between the client and server encrypted, which means that it cannot be seen by any third party that might be monitoring the network traffic. If the client decides to use one of these modes, the client and server application will also need to trust to the certificates of each other. Please refer to the Certificates View (Section 5.8, “Certificates View”) for details about creating the trust between the applications.

If you wish to enable insecure connections for testing the server, you can select the None option from *Security Modes*. Using this option in publicly available networks is not recommended.

Security Policies define alternative algorithms that the client applications may choose from. It is important to enable algorithms that the client applications support.

5.5.2. UA HTTPS Transport Protocol

UA HTTPS is an alternative transport protocol, which is not required by OPC UA, but which can enable an alternate communication pathway for some installations.

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

Once you have changed the endpoints, you must click Apply to save the settings and restart the server before they come to effect. If you don't apply the settings, you can use Revert button to restore the previously stored settings.

5.6. Users View

The *Users View* is only available in the Expert Mode (Section 5.4, "Expert Mode").

The Users View (Figure 13, "You can use the Users View to add or remove users that may access the OPC UA server.") lets you configure the *User Authentication Methods* as well as the user accounts that can be used to access the OPC UA server.

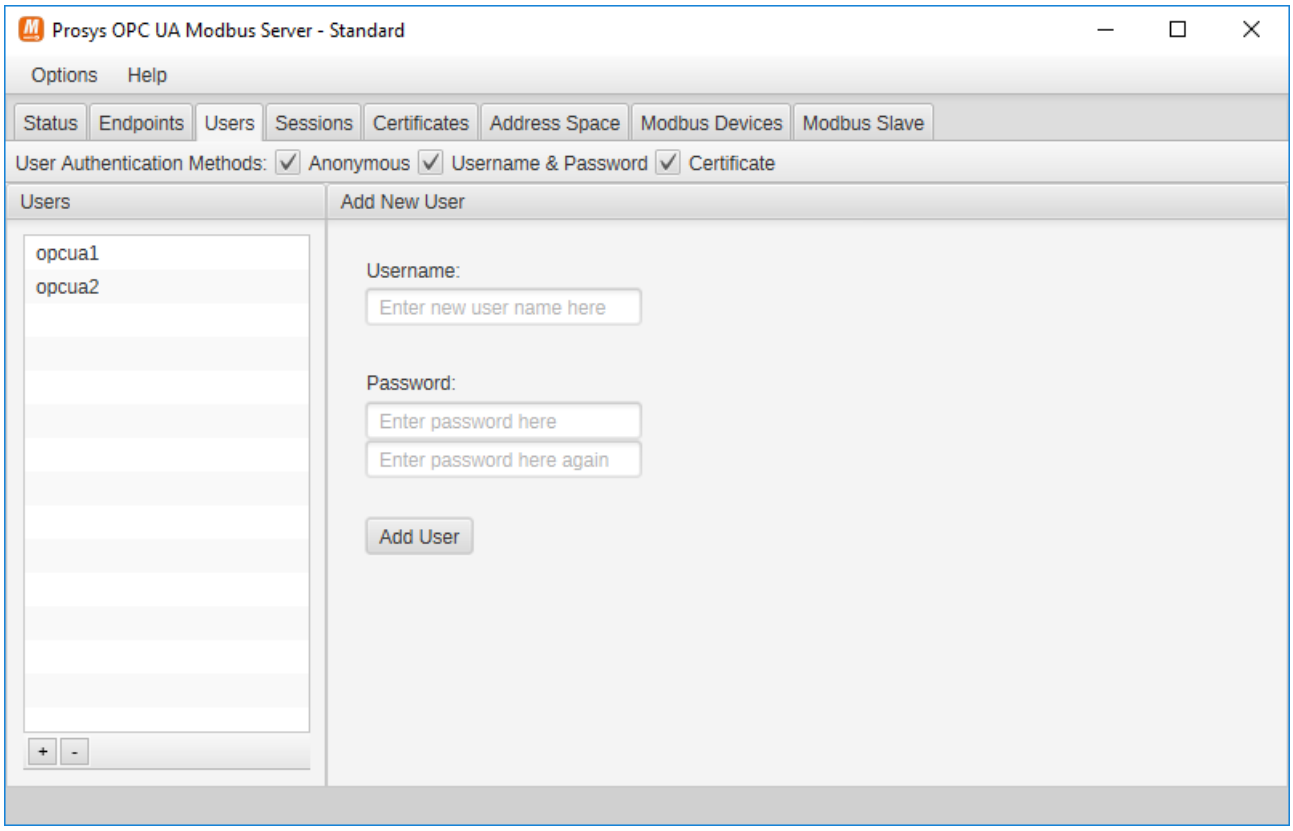


Figure 13. You can use the Users View to add or remove users that may access the OPC UA server.

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

- **Anonymous**, which enables connection without any specific user account.
- **Username & Password**, which enables traditional user name and password combinations to be defined.
- **Certificate**, which enables the server to accept users based on X.509 certificates.

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

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

If you have enabled the Username & Password authentication, you can define the users that may access the OPC UA server. The currently defined users are visible in the *Users* on the left. Use the + button at the bottom of the Users to add a new user and - button to remove the selected user.

If you have enabled the Certificate based authentication, you can define the trusted user certificates by adding them to the respective location in the USERS_PKI folder as described in Section 7.2, "Certificate Stores".

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

5.7. Sessions View

The *Sessions View* is only available in the Expert Mode (Section 5.4, “Expert Mode”).

The Sessions View (Figure 14, “The Sessions View displays all current OPC UA sessions in the OPC UA server.”) contains information about the currently open OPC UA client sessions.

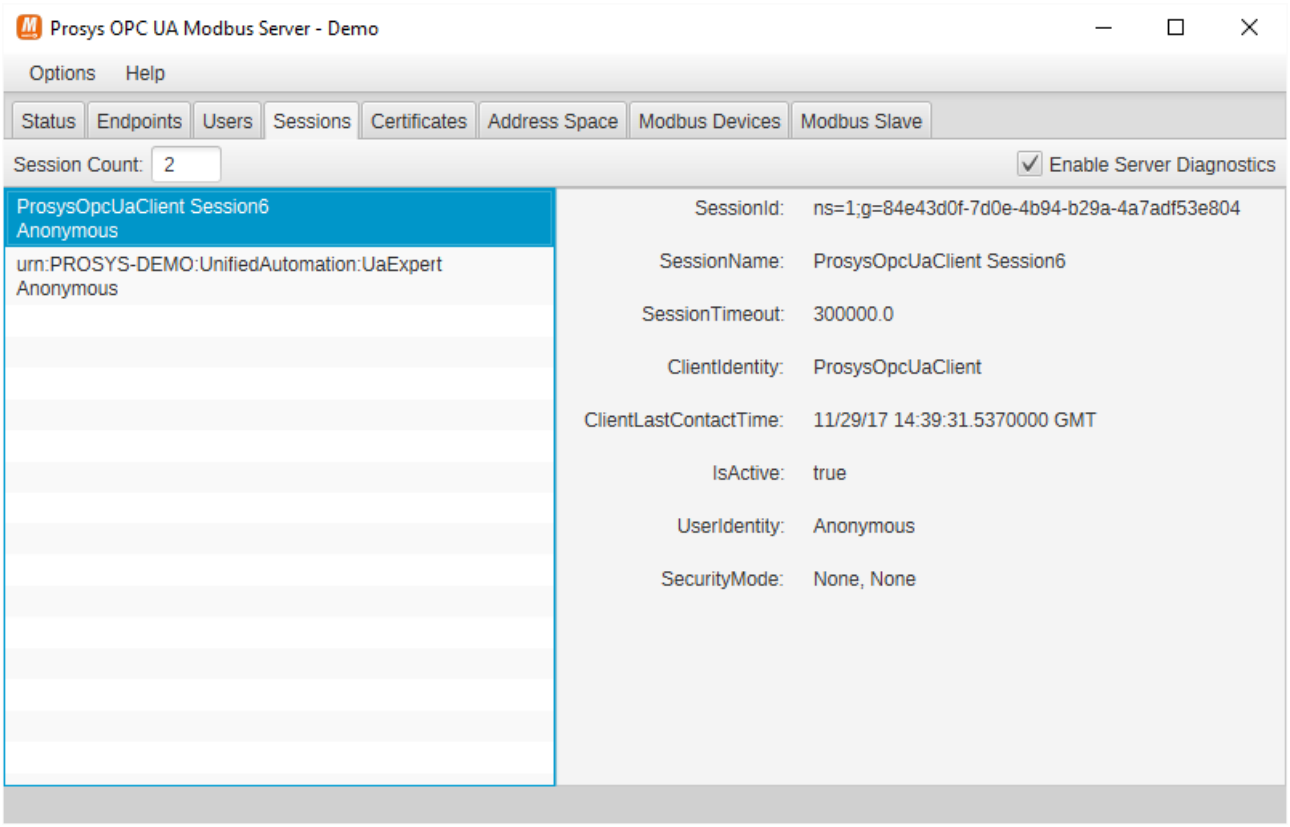


Figure 14. The Sessions View displays all current OPC UA sessions in the OPC UA server.

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

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

5.8. Certificates View

The *Certificates View* is only available in the Expert Mode (Section 5.4, "Expert Mode").

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

The Certificates View (Figure 15, "The Certificates View allows you to define which certificates you trust.") allows you to define which OPC UA client applications are allowed to connect to the OPC UA server. This is the first layer of validation that is available in OPC UA technology, prior to user authentication that is managed in the Users View.

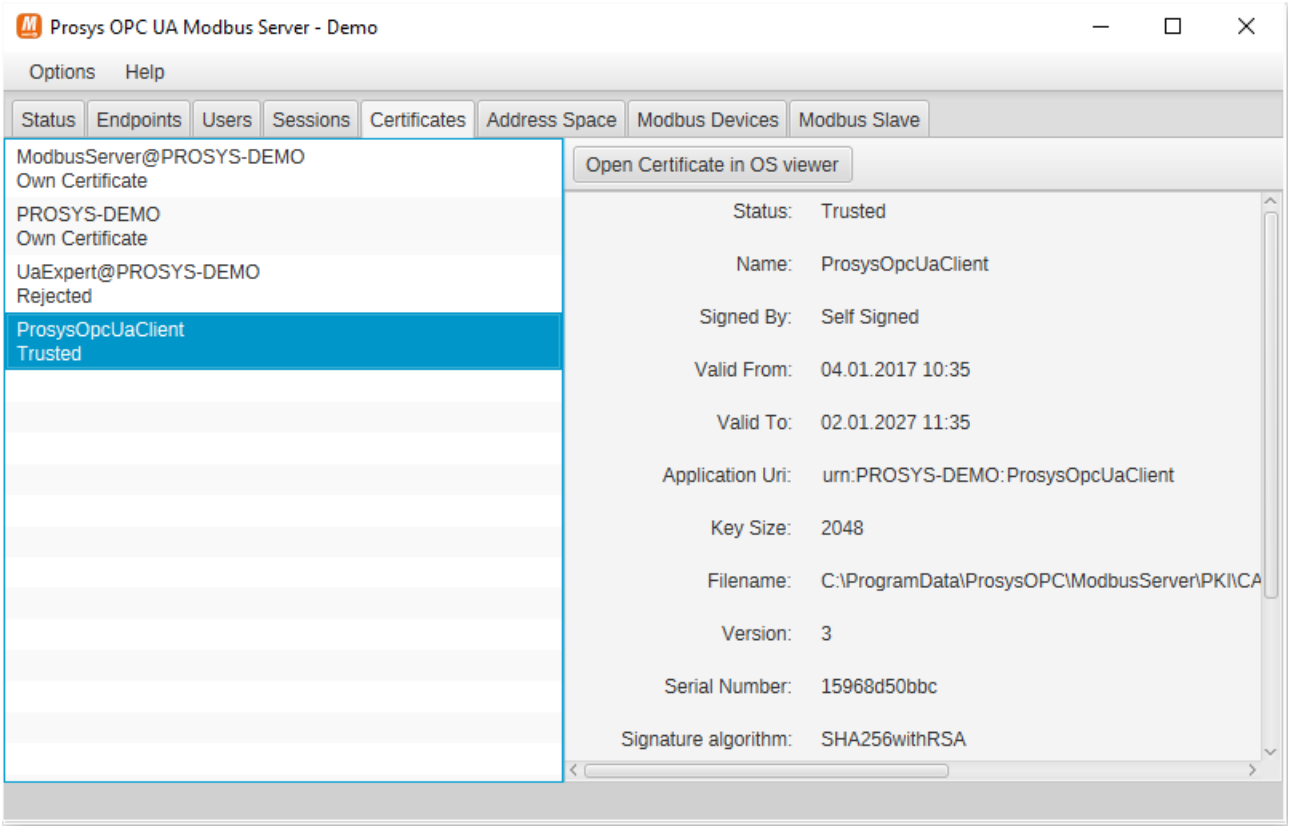


Figure 15. The Certificates View allows you to define which certificates you trust.

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

If your operating system is capable of displaying contents of certificate files, you can click the *Open Certificates in OS viewer* button to launch that for the active certificate. The certificates are stored in a Certificate Store (Section 7.2, "Certificate Stores").

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

OPC UA Application Instance Certificates are only used for authenticating applications, when secure OPC UA communications are used. If you wish to always require application authentication, you will need to disable the None level of security in the Endpoints View.

If you enable UA HTTPS from the Section 5.5, “Endpoints View”, the applications will not use the OPC UA Application Instance Certificates for authentication. Instead, they will use separate HTTPS certificates, via the TLS authentication. The HTTPS certificates are usually signed by a CA certificate and in order to trust each other, the applications may need to trust the CA certificates as well. Prosys OPC UA Modbus Server does not validate the HTTPS certificates of the client applications at the moment.

5.9. Address Space View

The *Address Space View* is only available in the Expert Mode (Section 5.4, “Expert Mode”).

The Address Space View (Figure 16, “The Address Space View. The Nodes are visible in the tree view on the left and the Attributes and References of the selected node are displayed on the right.”) shows the OPC UA server address space as it will be available for the client applications. The view is also similar to the one used by Prosys OPC UA Browser. The nodes are shown in the tree view on the left and the Attributes and References of the currently selected node are shown on the right.

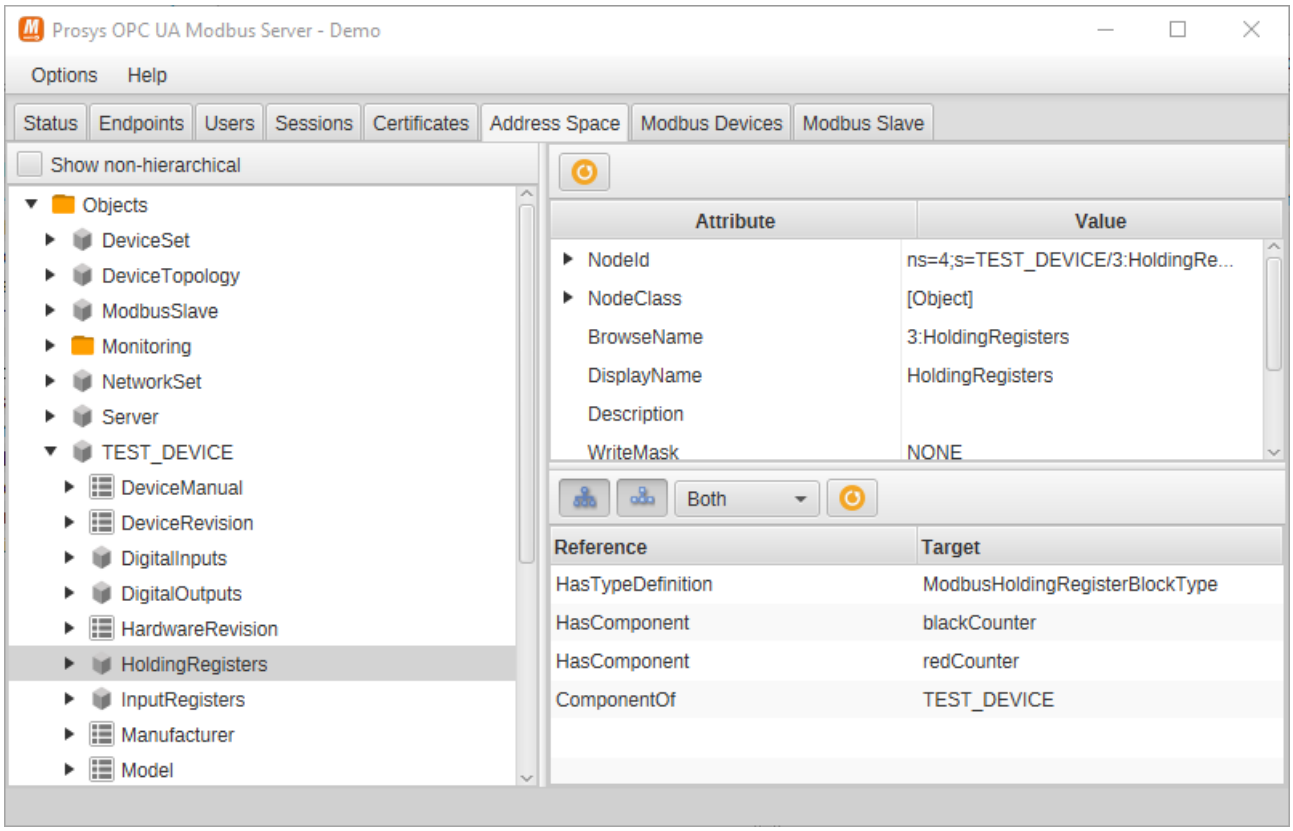


Figure 16. The Address Space View. The Nodes are visible in the tree view on the left and the Attributes and References of the selected node are displayed on the right.

Read more about the contents of the address space at OPC UA Server Address Space.

6. OPC UA Server

Prosys OPC UA Modbus Server will expose the configured Modbus devices and their blocks to OPC UA client applications in the network via an internal OPC UA server.

6.1. OPC UA Server Address Space

OPC UA applications provide the data that they have available for OPC UA client applications in the OPC UA address space. This helps the client applications to locate all relevant data from the server, when they don't have any prior knowledge about it.

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

You can use Prosys OPC UA Client to explore the address space visually and access the information of the OPC UA server. You can also use the Address Space View to verify how the address space will look for the client applications.

Prosys OPC UA Modbus Server follows the standardised outline of an OPC UA server address space. The address space contains three main parts, available as OPC UA Folders:

- *Objects*
- *Types*
- *Views*

The contents of each of these Folders that is specific to the Prosys OPC UA Modbus Server is described in the following sections.

6.2. Objects

Prosys OPC UA Modbus Server provides the following objects in the *Objects* folder:

- *Server* object is a standard object that provides the status and capability information about the server.
- *DeviceSet* object is a standard object that is defined in the OPC UA Device Integration (DI) information model. All Modbus devices that are available from the Prosys OPC UA Modbus Server are found under this object.
- *DeviceTopology* is a standard object that is defined in the DI information model. This is not used in Prosys OPC UA Modbus Server and therefore the object does not contain any additional information.
- *Monitoring* folder contains LifeBit variables for additional monitoring of the OPC UA connection.
- *ExternalModbusDevices* folder organises all the external Modbus slave devices that the Modbus Server is connected to.
- *InternalModbusDevices* folder organises all the internal Modbus slave devices hosted by the Modbus Server.
- *<Device>* objects, which appear under the configured device names, include the data of the connected Modbus devices. The same objects, including both the data and configuration can be

found under *DeviceSet* as well as divided into the *ExternalModbusDevices* and *InternalModbusDevices* folders.

Each <Device> object contains the following information:

- *DeviceRevision*, *Manufacturer*, *Model*, etc. These are defined in the Device Integration (DI) information model, but they are not currently used by the Prosys OPC UA Modbus Server.
- *ParameterSet* which includes all the defined parameters specific to the Modbus device such as the unit ID and protocol variant. For TCP/IP communication, the parameters also include the IP address and port number. For serial communication, the parameters include the baud rate, number of data bits, echo, input flow control, output flow control, parity, port and number of stop bits.
- *DigitalInputs* table contains the discrete inputs of the Modbus device.
- *DigitalOutputs* table contains the coils of the Modbus device.
- *InputRegisters* table contains the readable input registers of the Modbus device.
- *HoldingRegisters* table contains the readable and writable holding registers of the Modbus device.

Each of the above tables contain:

- The Variables that are configured to be available, including their configuration properties.

6.3. Types

In addition to the instance data (objects and variables), the OPC UA servers provide complete type information in the address space. The types supported by the OPC UA server can be found in the *Types* folder.

Prosys OPC UA Modbus Server includes the following types (in the respective type hierarchy).

6.3.1. ObjectTypes

ObjectTypes is a standard folder, which includes type definitions for all the Object types in the server address space. It is initialised with a few standard types that are defined in the Device Integration (DI) information model (as subtypes of *BaseObjectType*):

- *TopologyElementType* is a base type for different topology elements.
  - *ComponentType* is a base type for different devices components.
    - *DeviceType* is a base type for different devices.
  - *BlockType* is a base type for different data and function blocks in devices.

Prosys OPC UA Modbus Server defines the following types as subtypes of *DeviceType*:

- *ModbusDeviceType* is a common base type for the Modbus devices. It contains the Modbus tables as components and a set of Properties describing parameters common to all Modbus devices, including the used Modbus protocol variant, the Modbus Unit ID, a flag identifying if the device is an internal device hosted by the OPC UA Modbus Server or an external device as well as the device-wide default settings for the byte order of variables (swap bytes and swap words).
  - *ModbusSerialDeviceType* is a specific type for the Modbus devices that utilise serial port communication (Modbus RTU or Modbus ASCII). It has several Properties describing parameters specific to serial port communication, including the baud rate, number of data bits, echo, input flow control, output flow control, parity, port and number of stop bits.

- *ModbusTCPDeviceType* is a specific type for the Modbus devices that utilise TCP/IP communication (Modbus TCP or Modbus RTU over TCP). It has several Properties describing parameters specific to TCP/IP communication, including the IP address and port number.

Prosys OPC UA Modbus Server defines the following types as subtypes of *BlockType*:

- *ModbusTableType* is a common base type for all the Modbus tables.
  - *ModbusDigitalInputsTableType* is a specific type for the Modbus Digital Input table.
  - *ModbusDigitalOutputsTableType* is a specific type for the Modbus Digital Output table.
  - *ModbusHoldingRegistersTableType* is a specific type for the Modbus Holding Register table.
  - *ModbusInputRegistersTableType* is a specific type for the Modbus Input Register table.

The tables contain all the defined variables as components of each table.

Additionally, Prosys OPC UA Modbus Server defines the following type as a subtype of *BaseObjectType*:

- *ModbusVariableConfigurationType* is an object type that includes the configuration properties of a Modbus variable.

6.3.2. VariableTypes

VariableTypes is a standard Folder, which includes type definitions for all the Variable types in the server address space. Variable types are used to define structures for data in a case where the elements of the Variable should be available as single values.

Prosys OPC UA Modbus Server does not currently define any new Variable types, but it is using the standard *DiscreteItemType* and *AnalogItemType* for the digital and register Variables, respectively.

6.3.3. DataTypes

DataTypes is a standard Folder, which defines all the DataTypes that are used in the server address space. Each Variable configured in the Prosys OPC UA Modbus Server will define its DataType as a reference to the OPC UA DataTypes according to Table 4, “Supported Data Types”.

Prosys OPC UA Modbus Server also defines the following types as a subtype of *Enumeration*:

- *ModbusDataType* defines the alternative Modbus data types supported by the server.
- *ModbusProtocolVariant* defines the alternative Modbus communication protocol options: TCP, RTU over TCP, RTU and ASCII.

6.3.4. ReferenceTypes

ReferenceTypes is a standard Folder, which defines all the ReferenceTypes that are used to define relationships between the Nodes in the address space. Prosys OPC UA Modbus Server defines the following ReferenceType as a subtype of References/HierarchicalReferences/HasChild/Aggregates:

- *HasModbusVariableConfiguration* is the ReferenceType that is used to link the Variable Configuration to the Variables.

6.4. Modbus Information Model

All types that are specific to Prosys OPC UA Modbus Server are defined in the "http://prosysopc.com/UA/Modbus/" namespace. The respective ProsysOpc.Ua.Modbus.NodeSet2.xml file that includes the type definitions is found in the <installationFolder>/app/model.

6.5. Views

Prosys OPC UA Modbus Server does not define any Views in the OPC UA address space.

7. File Locations

7.1. Configuration Folder

The configuration files for the Prosys OPC UA Modbus Server are stored in different locations, depending on the way it was installed.

7.1.1. Windows Configuration Folder

If you installed the application from the Windows Installation Package, the configuration is located in %APPDATA%\ProsysOPC\ModbusServer where %APPDATA% is the location of the current computer's global application data folder. On Windows this is usually C:\ProgramData\ProsysOPC\ModbusServer

7.1.2. Linux Configuration Folder

If you installed the application to your Linux from the Debian or RPM Installation Package, the configuration files are located in /etc/ProsysOPC/ModbusServer

7.1.3. Portable Configuration Folder

If you installed the application using the Portable Installation Package, you will find the configuration files from <installation_folder>/conf

7.1.4. Configuration Files

The configuration of the application is stored in the following XML files in the Configuration Folder:

- uaServerConfig.xml - OPC UA server configuration
- userConfig.xml - user accounts for the OPC UA server
- devicesConfig.xml - configuration of the external Modbus device connections
- slaveConfig.xml - configuration of the internal Modbus slave functionality

You can modify the configuration settings directly in the files if the application is not running. But you should be **extremely careful** because invalid configuration will render the application unusable. It is therefore advisable to create a copy of the configuration file prior to modifying it, so that you can restore a working configuration in case any modifications cause program failure.

Before Modbus Server version 1.4.0, OPC UA server configuration was stored in serverConfig.xml XML file. When Modbus Server is started and no uaServerConfig.xml XML file is available, the application will attempt to migrate existing OPC UA server configuration from serverConfig.xml XML file. When migrating an existing configuration, the UA HTTPS transport protocol will be disabled regardless of whether or not it was enabled in the existing configuration. If no serverConfig.xml XML file is available, uaServerConfig.xml XML file with default settings is created.

The configuration format is subject to change in later versions of the application. Therefore, it is always advisable to use the application in the Configuration Mode (Section 5, “Configuration Mode”) to create and modify the configuration files.

The exact format of the configuration files is therefore out of the scope for this manual. But you can make small modifications or add more signal configurations, for example, by following the example of your existing configuration.

7.1.5. Resetting Configuration

If you want to reset to default settings, just remove the four Configuration Files mentioned above from the Configuration Folder.

There may be additional files in the same folder, which you should not remove.

7.1.6. Copying the Configuration

If you want to create a copy for backup purposes or copy the configuration from one server to another, you can just copy the four Configuration Files mentioned above from the Configuration Folder of the source server to the Configuration Folder of the target server.

7.1.7. Configuring Embedded Devices

As the configuration is based on XML files, you can always modify the configuration files directly on an embedded device, even when it does not have a graphical user interface to enable the Configuration Mode. However, the suggested way to initialize the configuration of a Portable Installation in an embedded device (a computer without graphical user interface), is to create the configuration in a Standard Installation and copy it from there to the Portable Installation (as mentioned above).

7.2. Certificate Stores

The Configuration Folder will also contain the Certificate Store Folders that are used to keep the certificates that are known by the application. The following directories are used:

- PKI for Application Instance Certificates
- USERS_PKI for User Certificates

The certificate stores contains a few sub-folders:

- certs for the trusted certificates
- crl for certificate revocation lists
- issuers for known issuer certificates and their revocation lists
  - issuers\certs for known issuer certificates
  - issuers\crl for certificate revocation lists of known issuer certificates
- rejected for the rejected certificates

Prosys OPC UA Modbus Server will reject all unknown certificates by default, unless they are signed by a

Trusted Issuer Certificate that is placed in the certs folder. Both certificate stores can also keep known Issuer Certificates in the issuers\certs folder to enable the application to validate trusted certificates that are signed by the respective Issuers or a chain of Issuers.

Although, you can modify the stores directly on the disk, it is usually better to use the Certificates View to define the trusted Application Instance Certificates. For User Certificates, you cannot yet use the user interface for this purpose so the only option is to define the trusted certificates in the folders.

7.3. License File

If you have purchased the license to the application, the respective information must be in the Configuration Folder. The best way to enter the license is to use the Enter License Dialog as explained in Section 4, "Application License".

If you are running the application in Portable Mode, you will have to copy the file into the Configuration Folder. Also make sure that the file name is changed to license.lic accordingly.

7.4. Application Logs

The application logs are placed in the log folder, which is in the Windows Configuration Folder (Section 7.1.1, "Windows Configuration Folder") in Standard Windows Installation, /var/log/ProsysOPC/ModbusServer in Standard Linux Installation and <installation_folder>/log in Portable Installation.

Logging is configured with the log4j2.xml file in the Configuration Folder (Section 7.1, "Configuration Folder"). See https://logging.apache.org/log4j/2.x/manual/configuration.html for more information on how to modify the logging configuration.

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

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

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