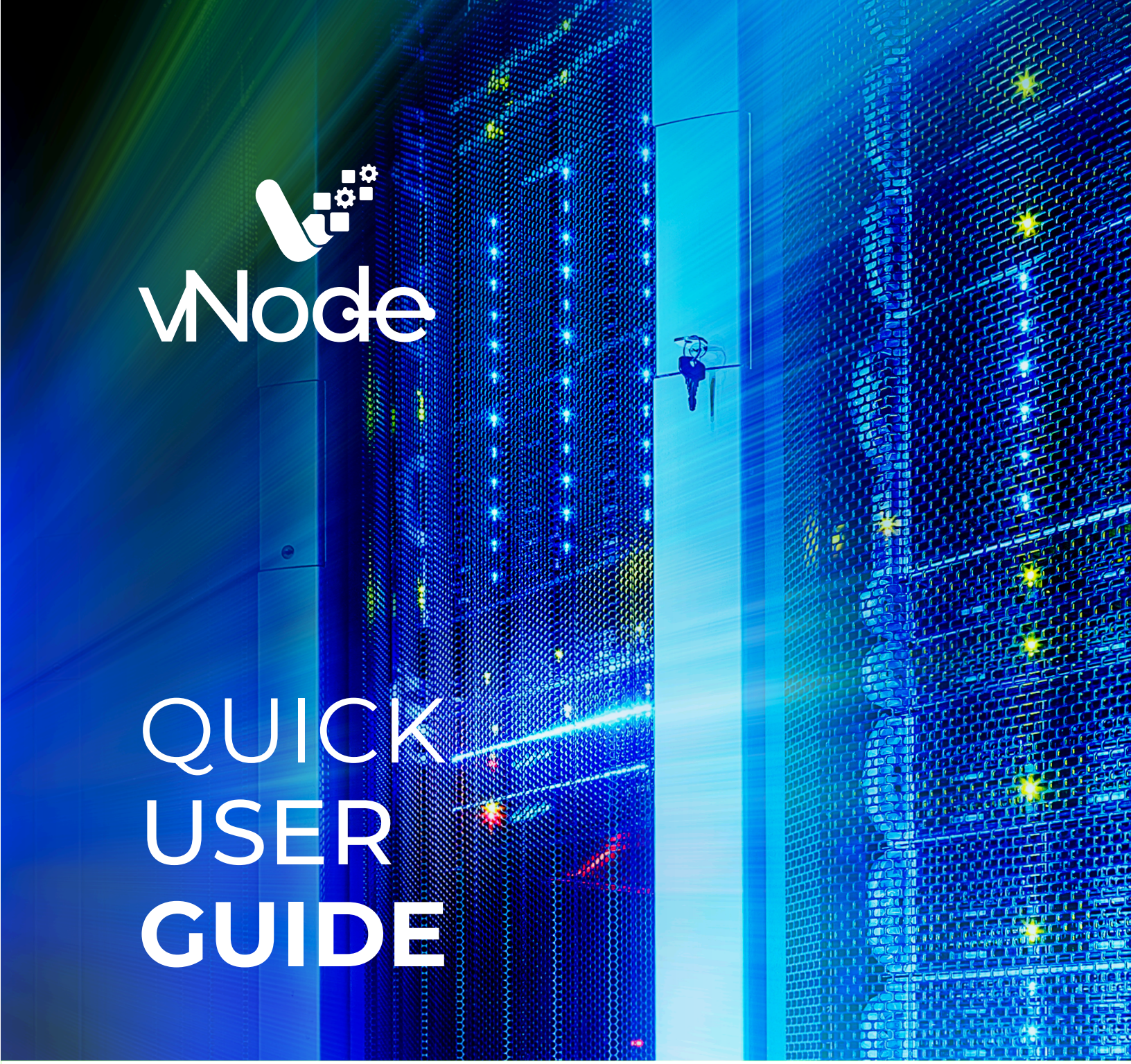


vNode

QUICK USER GUIDE

V.1.20

www.vnodeautomation.com

INDEX

INTRODUCTION

Welcome to vNode

Welcome to **vNode**, the next generation of accessible, scalable, and data-centric IIoT software. **vNode** was designed from the ground up to be approachable and easy to get started with, at the same time as being highly flexible and easy to scale up for larger projects. This guide aims to provide an introduction to **vNode** and its architecture, so you can get started as soon as possible.

vNode Modules

vNode is a modular platform, meaning that its functionality can be adapted to the specific needs of each application using different modules. The following connectors allow data to be exchanged with other systems:

Data Acquisition Modules

This set of **vNode** modules is responsible for collecting any signals originating from components placed within the industrial plant. All of our modules include the leading Automation System Vendor Protocols in order to meet the diverse needs of each and every company.

- **AcquisuiteXmlCollector:** HTTP collector for the Acquisuite-XML protocol. Receives and extracts data from the XML files received from Acquisuite dataloggers.
- **AuroraClient:** Client driver for PowerOne, ABB and Fimer solar PV inverters supporting Aurora protocol.
- **CustomClient:** User-configurable client driver for building any communication protocol.
- **DataImporter:** Enables data imports from CSV files.
- **DnpClient:** Client driver for DNP3 TCP/serial compatible slaves.
- **DominoClient:** TCP/serial driver for Domino industrial printers supporting Codenet protocol.
- **Iec102Client:** Client driver for electrical meters using IEC 60870-5-102 protocol.
- **LaetusWtClient:** Client driver for Laetus industrial supervision systems.
- **MarchesiniClient:** Client driver for Marchesini industrial packaging machines.
- **MettlerToledoClient:** Client driver for Mettler Toledo scales using SICS and Gareco protocols.
- **ModbusClient:** Modbus TCP/RTU client driver.
- **MqttClient:** Acts as a subscriber to enable data to be received from any MQTT broker.
- **OpcDaClient:** Driver for connecting to any OPC DA compliant server.
- **OpcUaClient:** Driver for connecting to any OPC UA compliant server.

- **OpcXmlClient:** Driver for connecting to any OPC XML DA compliant server.
- **RestApiClient:** Communicates to REST API Servers and extracts all data from the response. Supports JSON and XML formats.
- **SiemensClient:** Siemens S7 TCP client driver.
- **SmaClient:** Communicates to SMA solar inverters using the legacy SMA Sunny Net.
- **SqlClient:** SQL client driver compatible with SQL Server, MySQL, MariaDB and PostgreSQL.
- **XantrexClient:** Client driver for Xantrex GT solar inverters with a CCU2 board.

Data Delivery Modules

This set of **vNode** modules are in charge of delivering any signals collected by Data Acquisition Modules to major Clouds such as Azure, AWS, Google Drive Platform etc., as well as to any of the main SCADA systems currently available on the market, such as Wonderware, Ignition and others, for their subsequent data analysis.

- **DataExporter:** XML and CSV data aggregator and file exporter.
- **DbInjector:** Data injector for Microsoft® SQL Server.
- **DnpServer:** DNP3 server (slave) to provide data to any DNP3 compatible client.
- **ModbusServer:** Modbus server (slave) supporting Modbus TCP and RTU encapsulated.
- **MqttClient:** MQTT publisher/subscriber compatible with AWS, Azure, Google Cloud or any standard MQTT broker.
- **OpcUaServer:** OPC UA server.
- **RestApiServer:** REST server interface for real-time data, historical data and system status.
- **UflExporter:** Exports data to files ready to be consumed by Osisoft PI® UFL.

Edge Computing and Visualization Modules

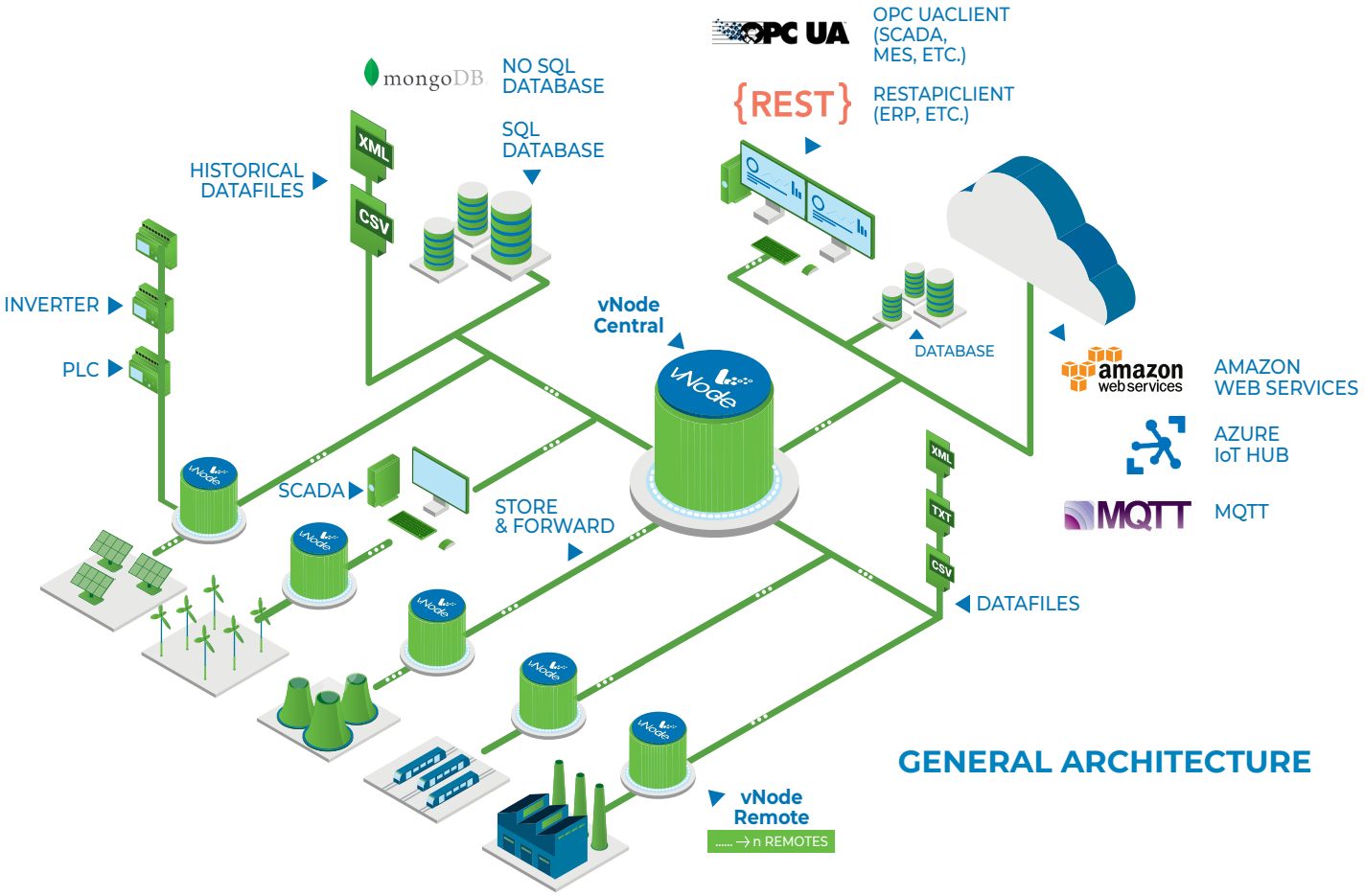
This set of **vNode** modules enable connected devices to process data closer to source, or even within the device itself. On the other hand, visualization system software modules are responsible for turning data into graphs for a better data analysis experience.

- **DerivedTags:** Configurable derived and aggregated data generator using expressions and data aggregation.
- **Historian:** High-performance time-series data storage and retrieval.
- **LinkedTags:** Configurable data linker.
- **ModbusGateway:** Modbus TCP/RTU gateway. Permits several Modbus concurrent connections to Modbus devices which only support one connection.
- **Scripting:** Advanced scripting based on NodeJS.
- **WebUI:** Web interface for configuration and commissioning of the **vNode** platform.
- **WebVision:** Pure web HMI/SCADA interface for industrial applications.

The Unlimited Industrial IoT Connectivity Software

vNode leverages the technologies and best practices from the Operations Technology (OT) and Information Technology (IT) worlds to providing an “Of-the-Shelf” solution for the Industrial Internet of Things (IIoT) and Industry 4.0.

The platform design allows user applications to connect, manage, monitor, and control diverse automation devices and software applications through one intuitive user interface.



GENERAL ARCHITECTURE

Licensing, Activation and Trial Mode

How the trial mode works

Each module in **vNode** can be used for one hour at a time, with no further restrictions. Upon expiration of the demo period, each module will automatically stop running. By logging into the **vNode** web interface, users can re-start the demo period and enable another hour of execution time for each module. The demo period may be restarted any number of times.

How licensing works

vNode is a modular platform, and licensing is therefore module-based. Licensed and un-licensed modules can operate side-by-side, allowing some modules to be tested in trial mode (demo) whilst others run in a licensed status (production).

After software installation is complete, a unique UID is generated and subsequently associated to the underlying hardware. After purchasing licenses for the required modules, the customer receives a file containing the corresponding license associated to that specific UID. Note that license files may contain the license for one or more modules.

In order to load a license file, open the WebUI, navigate to Licensing > Add license file and upload or copy/paste the license file. All modules included in the license file will automatically switch from demo to production mode. No restart is required.

vNode Installation

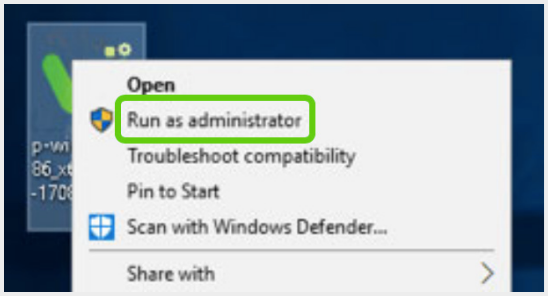
Getting **vNode** up and running is quick and easy. Installation takes less than a minute and the system will then be ready to immediately start collecting data. Simply download the installer from the **vNode** website, run the installer and the WebUI will automatically open as soon as the installer has finished.

Windows Setup

vNode is compatible with the following Windows versions:

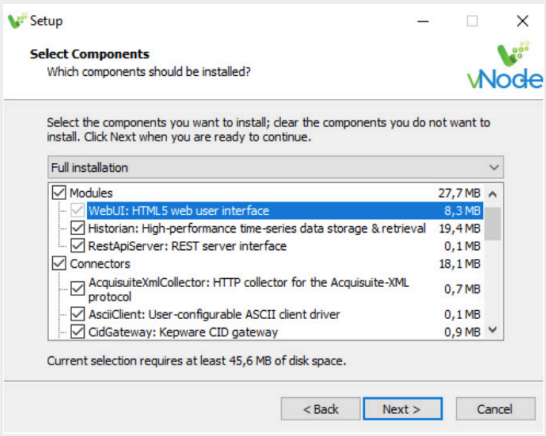
- **Windows XP, 7, 8, 8.1 and 10, including all editions (Home, Pro, IoT Enterprise).**
- **Windows Server 2003, 2008, 2008 R2, 2012, 2012 R2, 2016 and 2019.**

1 Setup: Right click on the setup file and select “Run as administrator”.



Step 1: vNode must be installed using “Run as administrator” option.

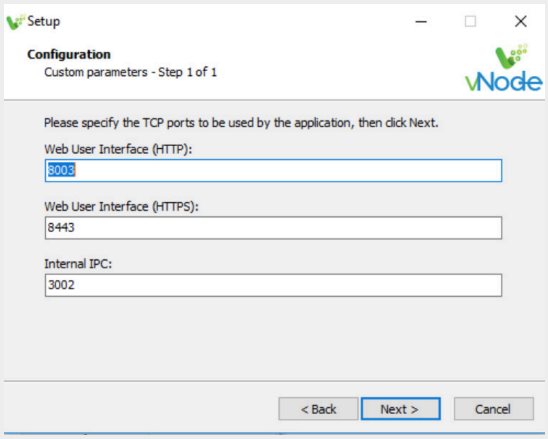
2 Setup: Choose the functionalities to be installed.



Step 2: Component selection.

In x64 bit systems, no external databases are required to run Historian, as a MongoDB instance will be automatically installed in the **vNode** folder to provide Historian storage. For other architectures (x32 and ARM), a MongoDB instance can be installed manually and used as a database for Historian.

3 Setup: Choose the required TCP ports and finish the installation.



Step 3: Default ports used by vNode.

TCP ports used by the vNode default installation:

- 8003: Web interface (HTTP)
- 8443: Secure web interface (HTTPS)
- 3002: Internal vNode communication

TCP ports assigned to vNode must not be in use by any other application.

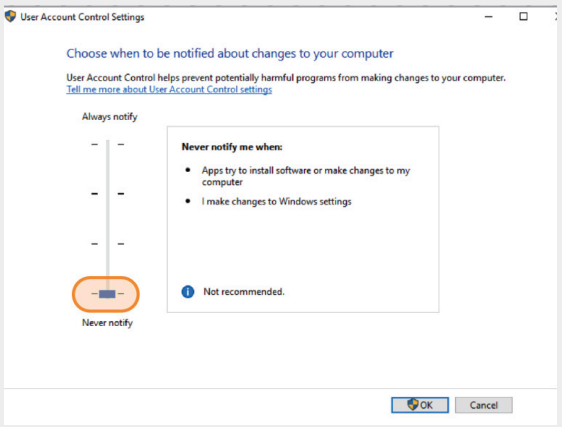
4 Setup: The WebUI will automatically launch in the default web browser. To access **vNode** WebUI from a different machine, use the machine's IP and the port that was configured for the WebUI during setup (by default 8003 for HTTP or 8443 for HTTPS)

	Full access	Read-only access
User:	admin	user
Password:	vnode	vnode

To access **vNode** WebUI from a different machine, make sure that the Windows Firewall on the host machine is not blocking the port that was been assigned to **vNode** WebUI during setup.

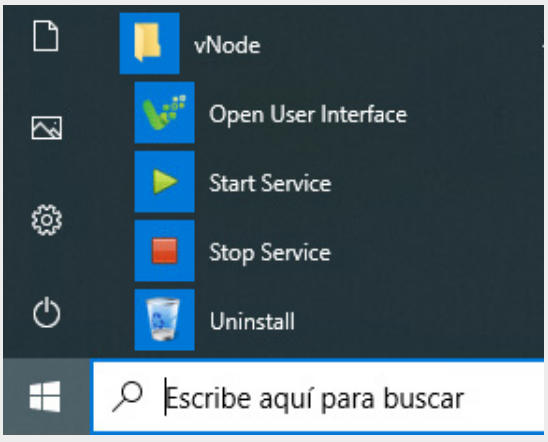
5 Setup: Set User Account Control to the lowest level possible.

The UAC may prevent loading of a backup configuration from the **vNode** Web interface. In order to avoid this problem, the UAC must be set to the lowest possible level.



Step 5: In order to load backup configurations from the vNode Web interface, User Account Control must be configured to the lowest possible level.

vNode runs as a service and is automatically initiated when Windows is started. The service can also be stopped/started manually from the Start menu.



Step 5: vNode Start Menu to Stop/Start the service

In demo mode, **vNode** will run with full functionality for one hour. To restart the demo period, simply restart each module or service from the Web interface. The service may also be restarted from the Windows Start Menu or Windows Services (**"vNode"** service). ("Stop Service" and them "Start Service").

Uninstalling vNode in Windows

To uninstall **vNode** from a Windows device, click the 'Uninstall' option in the **vNode** Start menu.

User Account Control Settings must be set to the lowest possible level in order to enable loading configuration backups from the **vNode** Web interface.

Updating vNode in Windows

To update **vNode** on a machine running on Windows, follow these steps:

- **Step 1:** Create a backup of **\vNode\bin and\vNode\config** folders. If something goes wrong during the update, restoring these folders will return the system to its original state.
- **Step 2:** Stop the service.
- **Step 3:** Run the installer for the new version using the "Run as Administrator" option in order to update the binary files to the new version. **vNode** will start automatically once the installation process is complete.
- **Step 4:** Login into the WebUI to check that everything is running properly.

It is strongly recommended to apply the update in a test environment before doing it in the production environment to ensure the configuration of the existing version is compatible with the new version.

Linux Setup

vNode has been tested on the following Linux distributions:

•Debian and derived from Debian like Ubuntu, Mint, Raspbian, DietPi and other specific distros.
•RHEL and derived like CentOS, Oracle Linux and Amazon Linux 2.
•Yocto Linux (depending on the specific compilation) and Gentoo.

For other distros please contact your **vNode** distributor or info@vnodeautomation.com

vNode does not require Linux GUI, which means that it can be installed on any headless device. It can either be installed using the console or an SSH connection. It can then be configured from the Web interface. This User Guide assumes a basic knowledge of Linux systems and their administration.

In order to run **vNode** in Linux Containers (LXC) the number of cores assigned to the LXC must be the same as the number of actual cores of the host system.

The following procedure will install **vNode** to the */opt* folder. It is also possible to install **vNode** to a different folder, depending on the end user's preferences.

1 Setup: Download **vNode** setup file.

A Linux setup file for all the different distributions can be downloaded from **www.vnodeautomation.com** and uploaded or copied to the target machine.

x64 Debian 9

cd /
sudo wget http://www.vnodeautomation.com/setup/linux/vNode-setup-linux-x64-debian9-last.tar.gz

x64 Debian 7,8

cd /
sudo wget http://www.vnodeautomation.com/setup/linux/vNode-setup-linux-x64-debian6_7_8-last.tar.gz

x64 CentOS 7

cd /
sudo wget http://www.vnodeautomation.com/setup/linux/vNode-setup-linux-x64-centos7-last.tar.gz

x86 Any OS

cd /
sudo wget http://www.vnodeautomation.com/setup/linux/vNode-setup-linux-x86-last.tar.gz

ARM Raspbian

cd /
sudo wget http://www.vnodeautomation.com/setup/linux/vNode-setup-linux-ARM-last.tar.gz

2 Setup: Decompress the file, where <distro> is the Linux distribution in the target machine and <version> corresponds to the **vNode** version downloaded.

sudo tar -xvzf vnode-setup-linux-<distro>-<version>.tar.gz -C /opt/

3 Setup: Install vNode

sudo /opt/vnode/bin/vnode install

4 Setup: To access the **vNode** WebUI from a different machine, use the machine's IP and the port that was configured for the WebUI (by default 8003 for HTTP or 8443 for HTTPS).

In x64 bit systems, external databases are not required to run Historian, as a MongoDB instance will automatically be installed in the **vNode** folder to provide Historian storage. For other architectures (x32 and ARM), a MongoDB instance can be manually installed and used as a database for Historian.

In order to use the embedded MongoDB database in Debian 9 x64 several dependencies must be installed using the following command:

sudo apt install libboost-chrono1.62.0 libboost-filesystem1.62.0 libboost-program-options1.62.0 libboost-regex1.62.0 libboost-system1.62.0 libboost-thread1.62.0 libgoogle-perftools4 libpcap0.8 libpcrecpp0v5 libsnappy1v5 libstemmer0d libtcmalloc-minimal4 libunwind8 libyaml-cpp0.5v5

Default vNode WebUI users.

	Full access	Read-only access
User:	admin	user
Password:	vnode	vnode

To access the **vNode** WebUI from a different machine, make sure that the **vNode** host machine is reachable and that there are no firewalls blocking the port assigned to **vNode** WebUI (8003 and/or 8443 by default).

In demo mode, each module runs with full functionality for one hour. In order to restart the demo mode, simply restart the module from the Web interface.

vNode service can be controlled from the console using the following commands:

```
sudo systemctl stop vnode
sudo systemctl start vnode
sudo systemctl restart vnode
systemctl status vnode
```

For older Linux versions, like Debian 7, a different command must be used to control the service

```
sudo service vnode stop
sudo service vnode start
sudo service vnode restart
service vnode status
```

Uninstalling vNode in Linux

To uninstall **vNode** from the host machine, run the following commands:

Step 1: Uninstall **vNode** service.

sudo <vnode folder>/bin/vnode uninstall

For example, if **vNode** is installed in the '/opt/vnode' folder, the command would be:

sudo /opt/vnode/bin/vnode uninstall

Step 2: Delete **vNode** folder (optional)

sudo rm -r <vnode folder>

For example, if the **vNode** folder is /opt/vNode the command would be:

sudo rm -r /opt/vnode

Updating vNode in Linux

To update **vNode** on a machine running Linux, follow these steps:

1 Setup: Create a backup of /vnode/bin and vnode/config folders. If something goes wrong during the update, restoring these folders will return the system to its original state.

2 Setup: Stop the **vNode** service:

```
sudo systemctl stop vnode
```

3 Setup: Extract the new version of the binary files from the installer, where <distro> is the Linux distribution on the target machine and <version> corresponds to the **vNode** version that has been downloaded. The following command assumes **vNode** is installed in /opt/vnode:

```
sudo tar -xvzf vnode-setup-linux-<distro>-<version>.tar.gz -C /opt/ ./vnode/bin
```

4 Setup: Re-start the service again and login into the WebUI to check that everything is running correctly:

```
sudo systemctl start vnode
```

It is strongly recommended to apply the update in a test environment before doing it in the production environment to ensure the configuration of the existing version is compatible with the new version.

vNode Web interface: WebUI

WebUI is the interface used to configure **vNode** and can also be used to quickly monitor any data collected by **vNode**. WebUI is a pure HTML5 web application, meaning that it can be opened in any modern web browser, which makes it very convenient for remote access.

By default, WebUI can be reached using plain HTTP (port 8003 by default) and secure HTTPS (port 8443 by default). In order to force secure connections only, HTTP mode can be disabled.

WebUI is automatically installed with **vNode** and does not require a license. More than one instance of the WebUI can be created in the same node (for instance when configuring a different logo, access from different networks, etc.). In this case, any new instances would run in demo mode unless a valid license is provided.

WebUI displays the following sections:

Data:

- **Real-Time:** Displays the values of all collected data, along with the quality, timestamp and description.
- **Historical:** Allows users to create charts displaying the historical values of all tags and export data to csv files. Historical data stored in other linked nodes can be retrieved and displayed also.

Alarms:

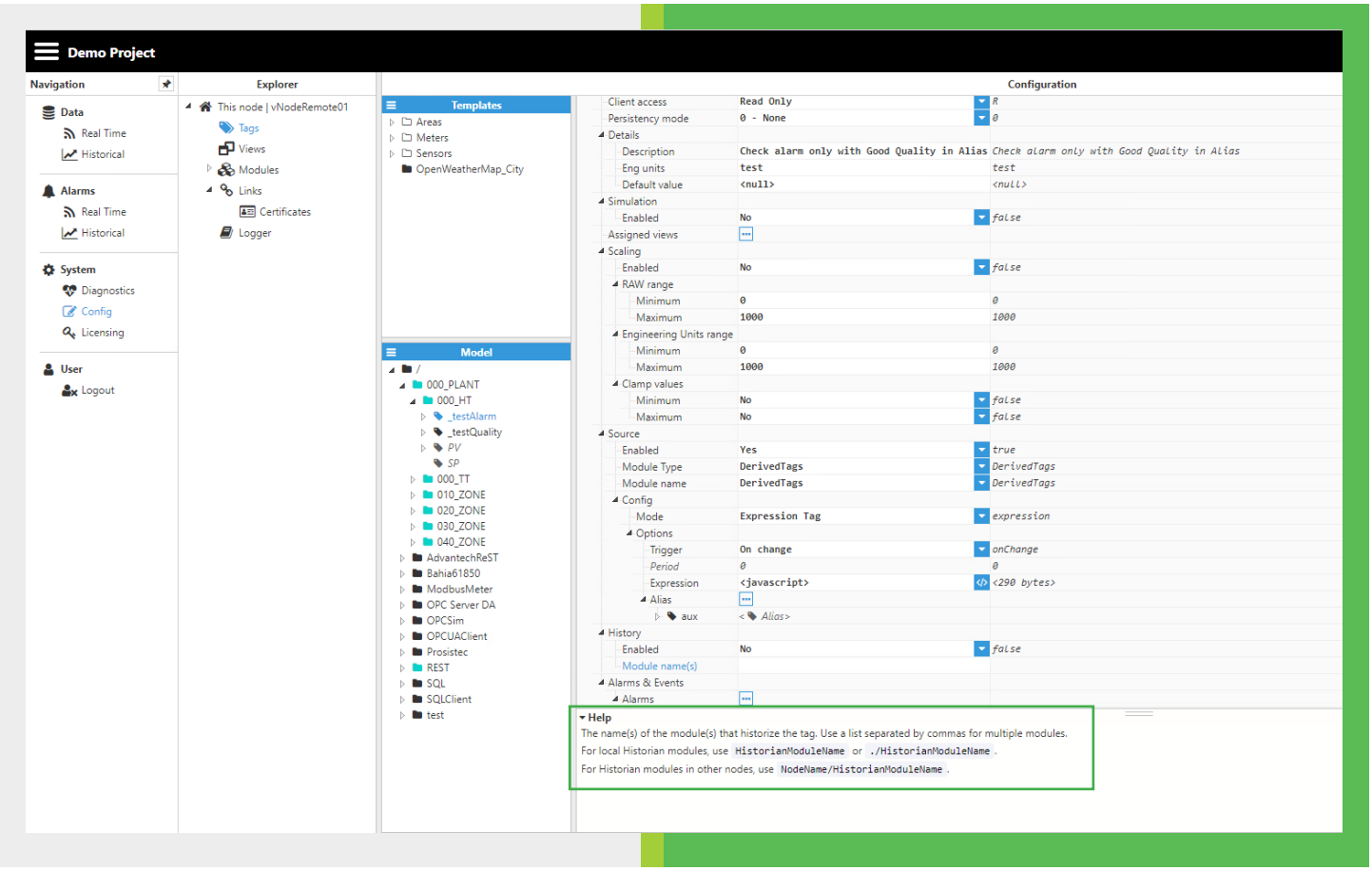
- **Real-Time:** Displays the current status of the alarms.
- **Historical:** Allows users to retrieve the events related to alarms from the historical data base.

System:

- **Diagnostics:** Displays the current status of each node and its components.
- **Config:** Allows users to configure the node and all the components.
- **Licensing:** Displays licensing information and allows to apply licenses to each node.

Inline Help

WebUI provides an inline help box at the bottom of the configuration area, displaying the description of each parameter and configuration examples.



Basic Steps for configuring a vNode node

The following steps allow users to configure a **vNode** node in order to collect data from field devices and share this data with other systems.

Step 1: Setup Modules to Activate Features

vNode functionalities are enabled using modules. In order to use a specific feature, the corresponding module must be installed during the setup process and activated in the configuration settings. For example, the WebUI is a module that is automatically activated, making it instantly accessible as soon as installation is complete. Since **vNode** is a microservice-oriented architecture, each module runs as an independent process. Bootstrap is the core service that manages the rest of the processes.

Active functionalities or modules also require licensing. Each module requires a valid license to run in production mode. If a module doesn't find a valid license, it will run in demo mode for one hour. In order to restart the demo period, the module must be restarted.

Step 2: Configuration of Data Source Modules (field connections)

This step involves configuring all connections with field devices and is only necessary for source modules that require field connections such as OPC UA client, OPC DA client, Modbus client, Siemens client, etc.

Step 3: Setup of Data Tags

In order to create tags, all main properties must be provided:

- Data format
- Scaling
- Data Source (pointing to a connection configured in the previous step)
- Alarms
- Historization

Once the tag has been created and the configuration has been saved, the real-time value of the tag will be available from the Real-Time menu.

Step 4: Configuration of Data Destination Modules

vNode can output the collected data in several different ways:

- OPC UA server, Modbus server and DNP3 server.
- MQTT to Azure, Amazon Web Services or standard MQTT broker.
- Send data to Historian (based on MongoDB) or to a Microsoft® SQL Server database.
- REST API providing real-time data, node status and historical data.
- UFL connector to OSISoft PI (csv files containing events).
- Data files in XML and CSV format (events and aggregated data).
- Exchange data with other **vNode** nodes securely and with Store&Forward mechanism using **vNode** Links.

Each **vNode** node can exchange data with other nodes. When receiving connections from other nodes, the inbound connection should be configured. When connecting to other nodes, the outbound connection must be configured. See the chapter on **vNode** Links for more information about **vNode** Links.

Working with Templates

Templates can be created and used at different levels of configuration:

- **Tag level:** Tag templates and device templates containing tags.
- **Communications level:** Templates for communication clients (Modbus, OPC, Siemens, etc.) including preliminary configuration of the connection.

The Model tree for each section can be used to instantiate templates and provide the values for Custom Properties, assigning different values for each instance.
This User Guide assumes a basic understanding of Object-Oriented paradigm.

Custom Properties

In order to use the templates, it is possible to create Custom Properties within each template to be used as parameters. Custom properties are what differentiate instances derived from the same template. These Custom Properties can then be referenced in the Expressions to calculate specific values for each instance.
Custom properties are referenced in the Expressions using the name of the Custom Property in curly braces { }.

Expressions

JavaScript expressions can be used on each data entry to calculate the value. All expressions start with "=" (like in spreadsheets). Expressions are only evaluated during the start-up of the module.

Examples of expressions assuming the following Custom Properties:

Custom Property Name	Type	Value
Boolean01	Boolean	True
Boolean02	Boolean	False
Number01	Number	5
Number02	Number	25
Text01	Text	Hello
Text02	Text	World!

Action	Expression	Result
Concatenation	={Text01}+" "+{Text02}	Hello World!
Sum	={Number01}+{Number02}	30
Conditional	={Boolean01}==1?{Number01}:{Number02}	5
Conditional	=['A','B','C','D','E'][{Number01}-2]	D
String methods	={Text02}.substr(0,1)+{Text01}.slice(1,4)	Well

vNode Links

Introduction to Links

Each **vNode** node can connect to other nodes and exchange data using **vNode** Links. These connections between **vNode** nodes provide the following advantages:

- **Real-time:** Data flows continuously between nodes, displaying the current value of tags in both the source and destination node. Data is time stamped at the origin, maintaining time consistency across the entire fleet.
- **Secure:** All data sent is encrypted using the TLS 1.2 cryptographic protocol to prevent data tampering. **vNode** nodes exchange Digital Certificates for instant authentication.
- **Reliable:** All connections between **vNode** nodes include automatic Store&Forward mechanism, meaning that any data which is not delivered due to a communication outage between nodes is saved locally and automatically sent once connection is restored.
- **Firewall friendly:** No open ports are required at remote facilities.
- **Bi-directional:** Once the connection is established, it is fully bi-directional, so each node can both send and receive data. This makes links very convenient for sending commands to remote nodes.
- **Easy configuration:** Tags are only configured in the source node. Destination nodes display the same information as source nodes, without requiring any extra configuration.
- **Low bandwidth requirement:** All data sent is highly compressed so that links will work correctly on slow and high latency TCP connections, such as 2G and Satellite.

Link configuration

Each link requires two different nodes; the node initiating the connection and the node receiving the connection. Once the connection is established, it is fully bi-directional and data is exchanged between both nodes, independently of which node initiated the connection.

Connections initiated by a node are configured as Outbound connections. A node can initiate connections to many other nodes.

Connections received by a node are configured as Inbound connections. A node can receive connections from many other nodes.

Configuring a Link between two nodes requires three steps:

Step 1: Provide a unique name among all **vNode** nodes to each of the nodes connected.

Step 2: Configure the Inbound connection for the node receiving the connections. Once Inbound connection is enabled, it will listen to the configured port for incoming connections from other **vNode** nodes. The default port for incoming connections is 3001.

Step 3: Configure the Outbound connection for the node initiating the connection. The name of the Outbound connection must be the same as the name of the node receiving the connection.

See the example of Link configuration chapter for more information on configuring links for node connections.

vNode Historian

Introduction to Historian

vNode Historian is a high-performance time-series storage system based on a non-SQL database (MongoDB). In x64 bit systems, the ModgoDB instance is automatically installed in the **vNode** installation folder, so Historian is ready to store data as soon as installation is complete. In x32 and ARM architectures, the user must first install MongoDB so that Historian can then be configured to store the data in this ModgoDB instance.
vNode Historian can store any tag values that have been collected locally in the same node or those collected remotely by other nodes and received in the Historian node through **vNode** Links.
Historian provides efficient data compression and partitioning mechanisms, allowing the storage of massive volumes of time-series data without reducing its performance over time.

Data retrieval

Historian stores events (changes in value, quality, or timestamp). Data can be retrieved from the storage in different modes:

- **Raw:** Data retrieved contains all the values for all events stored in the database.

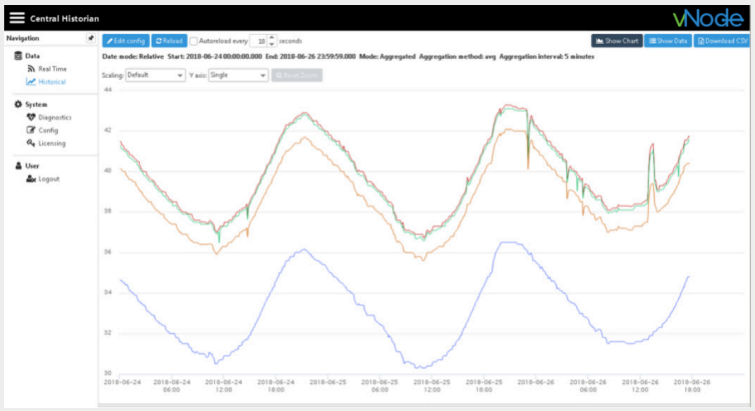
- **Aggregated:** Data is consolidated into aggregation periods. The following aggregation methods are available:
  - **vg:** Time-weighted average.
  - **min:** Minimum value during the aggregation period.
  - **max:** Maximum value during the aggregation period.
  - **first:** First value during the aggregation period.
  - **ast:** Last value received during the aggregation period.

- **Delta:** Data retrieved contains all the values of all events displaying incremental changes compared to any previous event that is larger than the configured deadband. It only contains the second event (the one taking place after the first event that surpassed the deadband).

- **Filter:** Data retrieved contains all the values for all events displaying changes compared to any previous event that is larger than the configured deadband. In this case it contains both events, the one before the change that surpassed the deadband and the one after the change.

Data stored in Historian can be retrieved using the following methods:

- **WebUI:** Includes a rich HTML5 interface to retrieve and visualize data in charts and data tables (raw and aggregated data). Data can be also exported to CSV files. The historical data retrieved can be located in the same **vNode** node as the WebUI or in a different **vNode** node, providing that the nodes are connected through a **vNode** Link.
- **REST API Server:** By using the optional RestApiServer module, historical data can be retrieved in JSON format using REST API calls (raw data and aggregated data). The historical data retrieved can be located in the same **vNode** node as the REST API server or in a different **vNode** node, providing the nodes are connected through a **vNode** Link.
- **Delta:** Data retrieved contains all the values of all events displaying incremental changes compared to any previous event that is larger than the configured deadband. It only contains the second event (the one taking place after the first event that surpassed the deadband).
- **MongoDB client:** Direct connection to the MongoDB database to retrieve data in raw mode.



SteHistorian charts from the WebUI

Historian configuration

Steps to configure Historian:

Step 1: Create the Historian module instance in the **vNode** node where data will be stored.

Step 2: Configure the historization for each tag pointing to the Historian module. If the Historian module is located in a different **vNode** node, then the module name will be "NodeName/ModuleName"

For more details about **vNode** Historian configuration, please refer to the Historian configuration example chapter.

vNode Logs

Bootstrap and instances from each module log all activity in their own log file. In this way, the log for one module is not affected by the behaviour of any other modules that are running in the same node. There are five different log levels:

- **Error:** Only logs errors
- **Warning:** Logs errors and warnings
- **Info (default):** Logs errors, warnings and general information messages
- **Debug:** Logs all activity pertaining to that module, with the exception of data from this activity.
- **Trace:** Logs all activity including all data associated with this activity. For example, in OPC clients, it will log any tag update together with its new value, quality and timestamp. In the ModbusClient module, this log includes all payloads exchanged with the Modbus devices.

Debug and Trace modes may log large amounts of data so they should only be used for troubleshooting. They should be avoided in production environments.

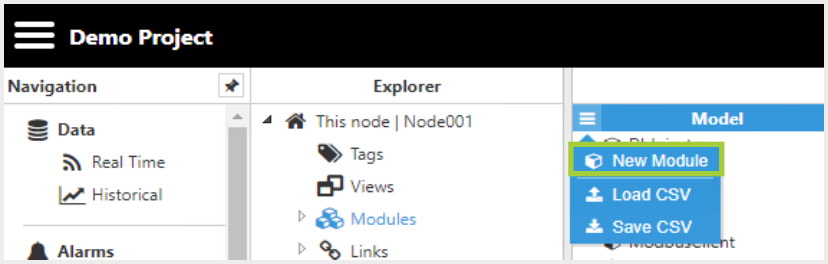
Log files can be retrieved from the WebUI in Diagnostics => This node => Export logs button. Log files can be opened using any text editor.

To avoid accumulating very large files, each module generates a new daily log file at 00:00 UTC. Older files are automatically deleted to avoid filling the hard drive with log files. The number of days that files should be stored for on the hard drive can be configured through the WebUI.

Modbus TCP client configuration example

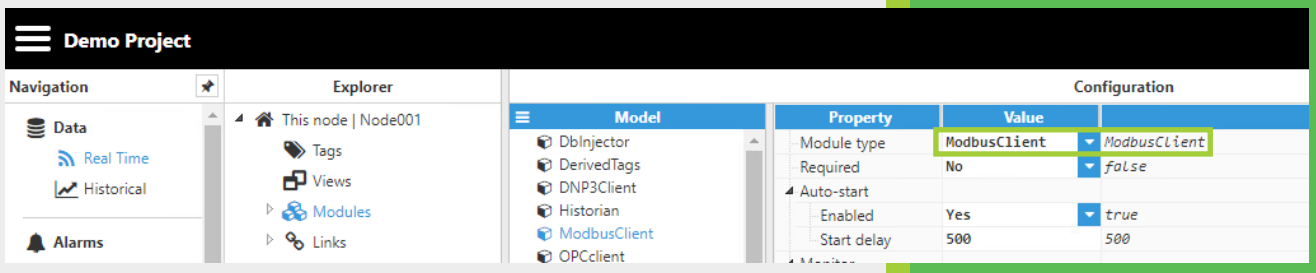
The following steps show how to read data from a Modbus TCP server. This User Guide assumes a basic understanding of Modbus communication protocol.

Step 1: Create the module: (Config => Modules => button to the left of Modules => New module)



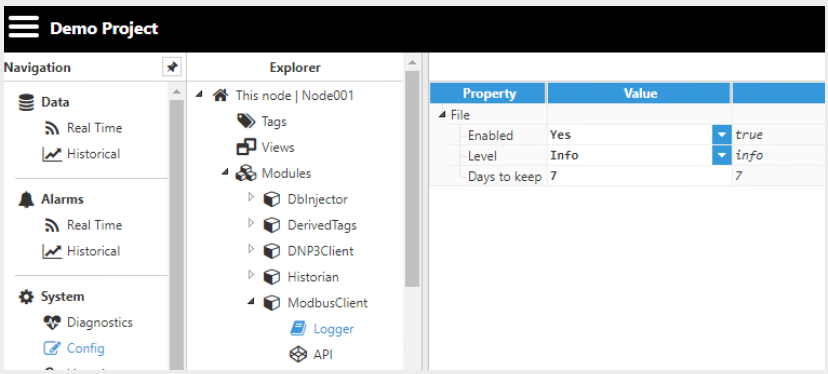
Step 1: New module creation

Step 2: Provide a name for the module (in this case MbClient), assign the type of module (in this case ModbusClient) and save the new configuration.



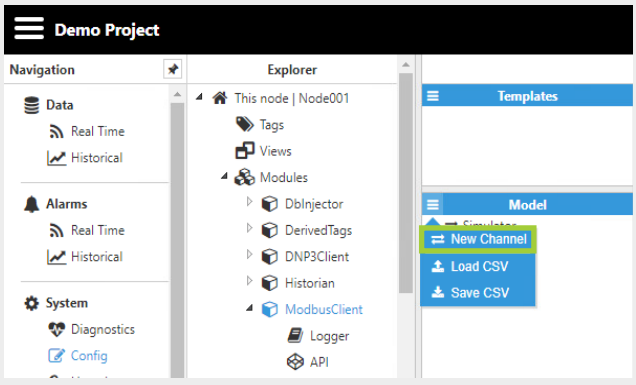
Step 2: Configuring new module as ModbusClient

Step 3: Configure the log (usually the default values are sufficient). Save the log configuration.



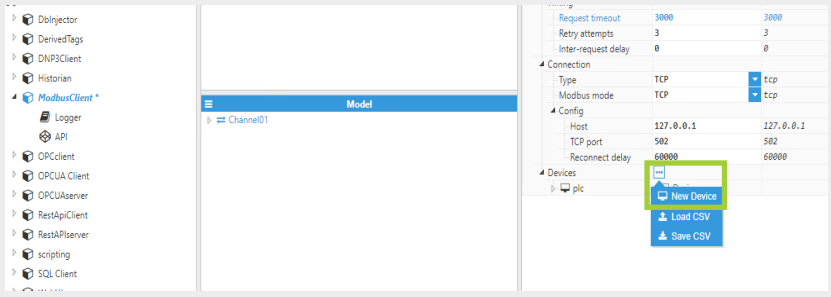
Step 3: Default log configuration

Step 4: Set up the Modbus channel. Each connection to a Modbus server is setup with a channel and a device. The channel represents the connection media (Ethernet or serial connection) and the device represents the Modbus server (or Modbus slave for serial connections). This means that in order to connect to a Modbus sever, a Modbus channel must be created and configured first, providing all the necessary communication settings.



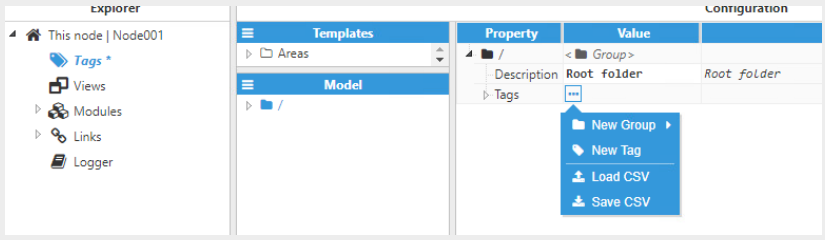
Step 4: New Modbus channel set up

Step 5: Set up the Modbus device. Save the Modbus driver configuration and restart the module once the device has been configured.



Step 5: New Modbus device setup

Step 6: Create a tag to connect through to the Modbus sever: Config => Tags => New Tag

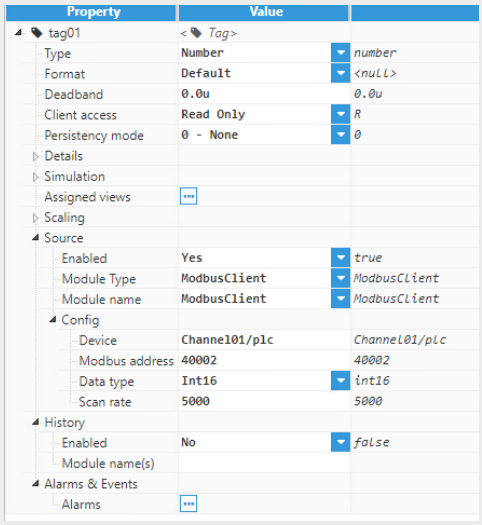


Step 6: New tag generation

Step 7: Configure the tag. All details regarding the communication should be configured in Source entry:

- **Source.Enabled:** True
- **Source.Module Type:** ModbusClient
- **Source.Module name:** MbClient (the module created in previous steps)
- **Source.Config.Device:** Channel01/1 (the channel/device created in previous steps)

- **Source.Config.Modbus Address:** The Modbus address of the tag
- **Source.Config.Data type:** The Modbus data type
- **Source.Config.Scan rate:** The signal's scan rate



Property	Value	
tag01	< Tag>	
Type	Number	number
Format	Default	<null>
Deadband	0.0u	0.0u
Client access	Read Only	R
Persistency mode	0 - None	0
Details		
Simulation		
Assigned views	...	
Scaling		
Source		
Enabled	Yes	true
Module Type	ModbusClient	ModbusClient
Module name	ModbusClient	ModbusClient
Config		
Device	Channel01/plc	Channel01/plc
Modbus address	40002	40002
Data type	Int16	int16
Scan rate	5000	5000
History		
Enabled	No	false
Module name(s)		
Alarms & Events		
Alarms	...	

Step 7: Configuration of tags belonging to Modbus devices

Step 8: The tag should now be available, displaying as good quality in the Real-Time display.

Step 8: Tag value in real-time display

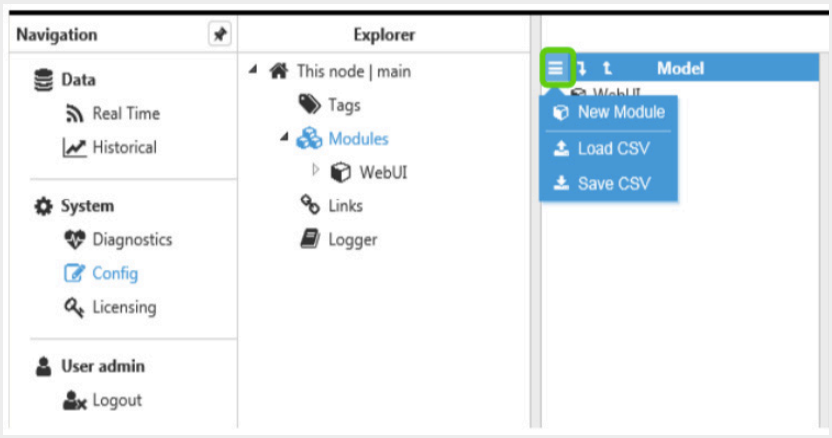
Log files for troubleshooting can be downloaded from the WebUI in Diagnostics => This node => Export logs button.

OPC DA client configuration

Example connection for KEPServerEX and TOPServer

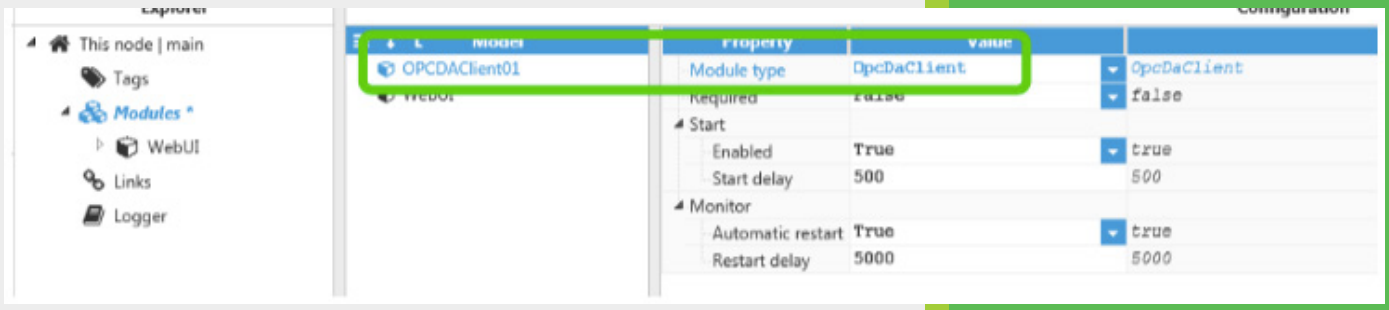
The following steps show how to connect to KEPServerEX and TOPServer using the OPC DA client.

Step 1: Create the module: (Config => Modules => button to the left of Modules => New module)



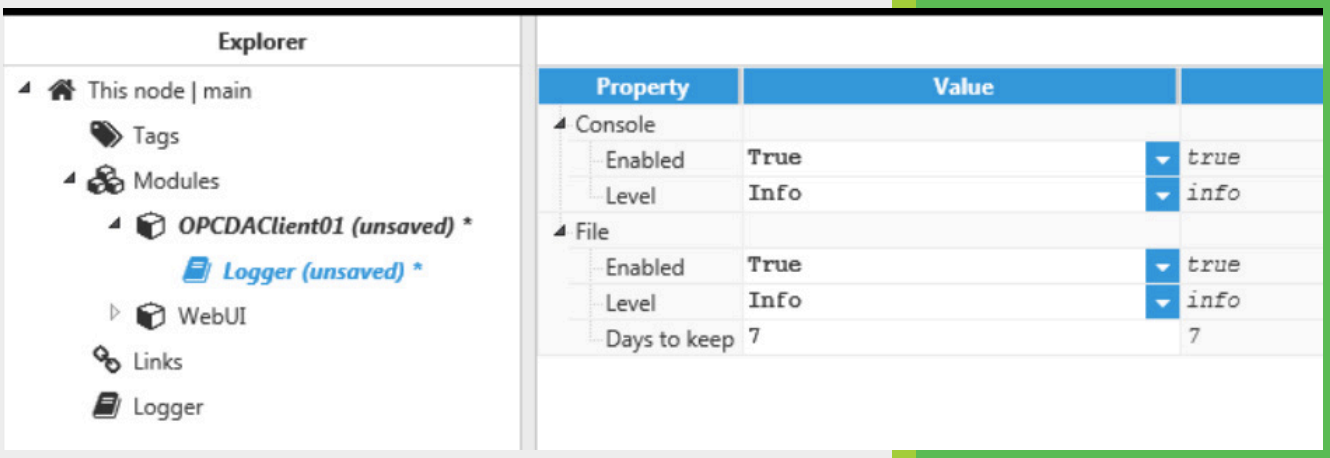
Step 1: New module creation

Step 2: Provide a name for the module (in this case OPCDAClient01), assign the module type (in this case OpcDaClient) and save the new configuration.



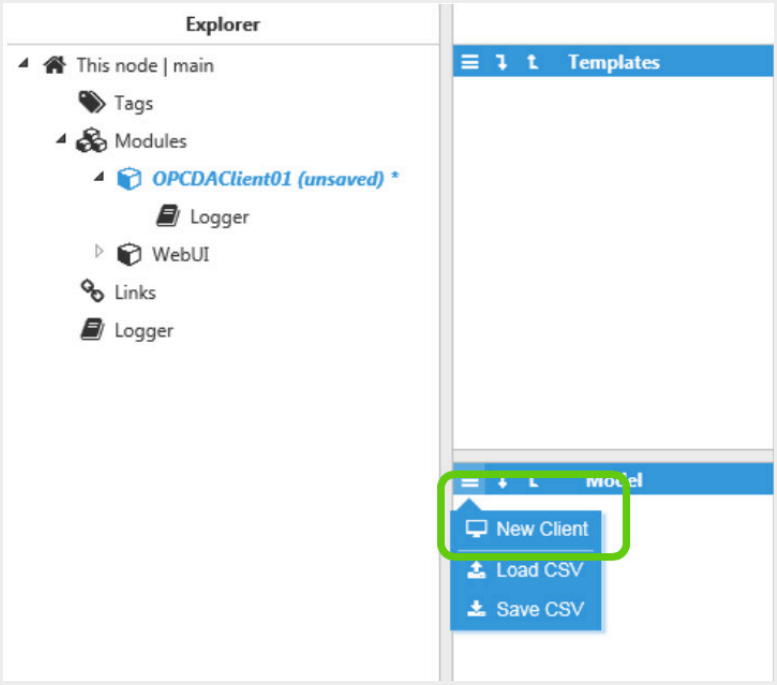
Step 2: Configuring a new module as OpcDaClient

Step 3: Configure the log (usually the default values are sufficient). Save the log configuration.



Step 3: Default log configuration

Step 4: Create the OPC DA connections. Each connection is an independent OPC DA client.

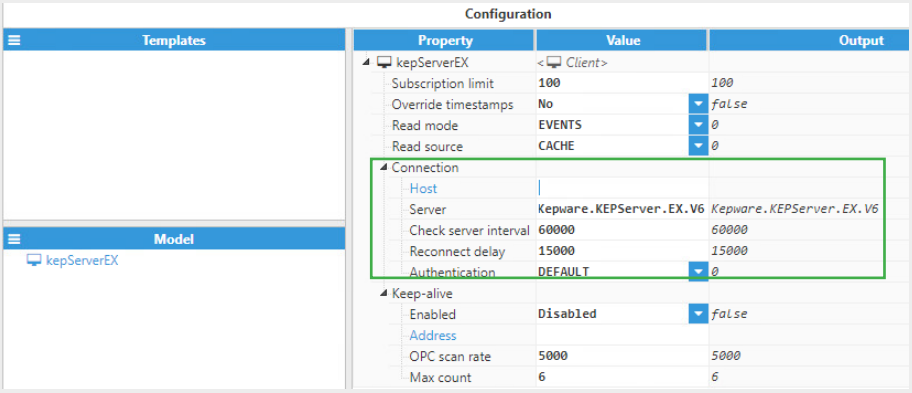


Step 4: Creating a new OPC DA connection to a server

Step 5: Configure the OPC DA client to connect to KEPServerEX or TOPServer.

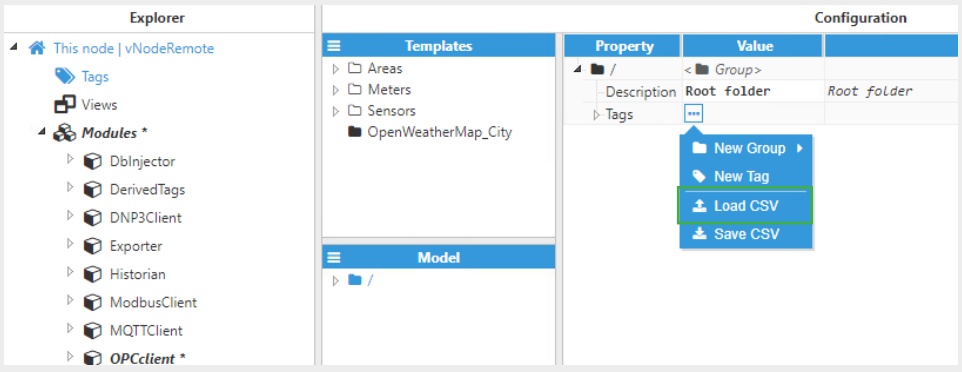
- **Host:** Enter the Hostname/IP address of the target OPC server. Empty means localhost.
- **Server:** Instance name of the target OPC server, also known as Prog ID. For KEPServer6 the instance name is Kepware.KEPServerEX.V6 and for KEPServerEX 5 the instance name is Kepware.KEPServerEX.V5. For TOPServer 6 the instance name is SWToolbox.TOPServer.V6 and for TOPServer 5 the instance name is SWToolbox.TOPServer.V5.

Save the configuration and select restart the module.



Step 5:
OPC DA client configuration for connecting to KEPServerEX 6 in the same host

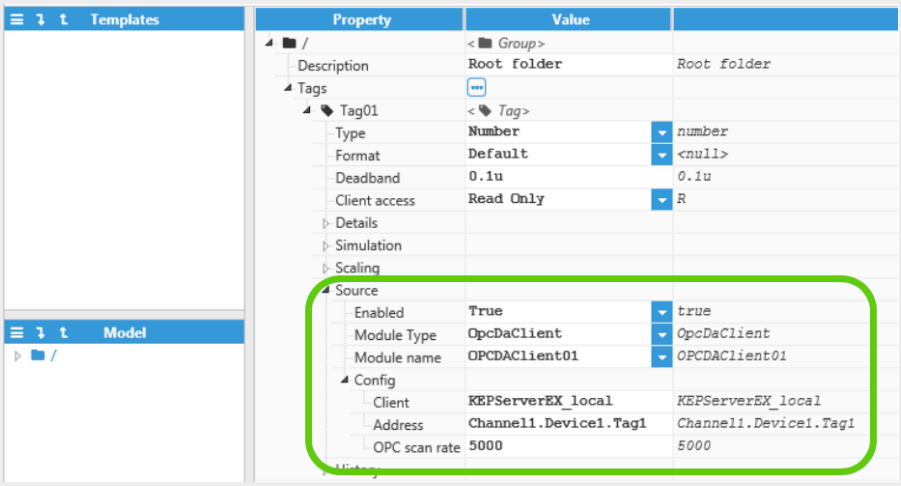
Step 6: Create a tag to connect to the OPC sever: Config => Tags => New Tag



Step 6:
New tag creation

Step 7: Configure the tag. All details regarding the communication should be configured in Source entry:

- **Source.Enabled:** True
- **Source.Module Type:** OpcDaClient
- **Source.Module name:** OPCDAClient01 (the module created in previous steps)
- **Source.Config.Client:** KEPServerEX_local (the connection created in previous steps)
- **Source.Config.Address:** The tag ID in the OPC server. In this example, a valid tag ID would be Channel1.Device1.Tag1 since KEPServerEX is running the default configuration after installation.



Step 7: New tag configuration

Step 8: The tag should now be available, displaying as good quality in the Real-Time display.

Step 8: Tag value in real-time display

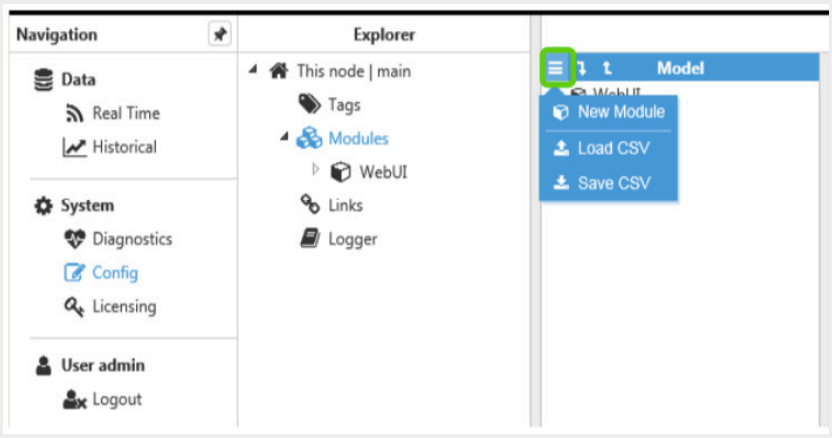
Usually no changes are required in the DCOM since **vNode** runs as a service and the default configuration for KEPServerEX permits the connection of local applications running as System.

Log files for troubleshooting can be downloaded from the WebUI in Diagnostics => This node => Export logs button.

Example connection to Matrikon OPC Simulation

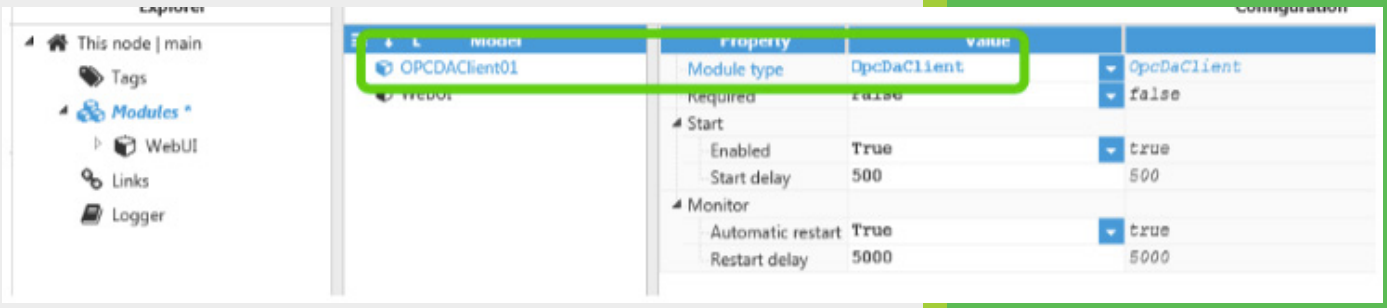
The following steps show how to connect to Matrikon OPC simulation using **vNode** OPC DA client.

Step 1: Create the module: (Config => Modules => button to the left of Modules => New module)



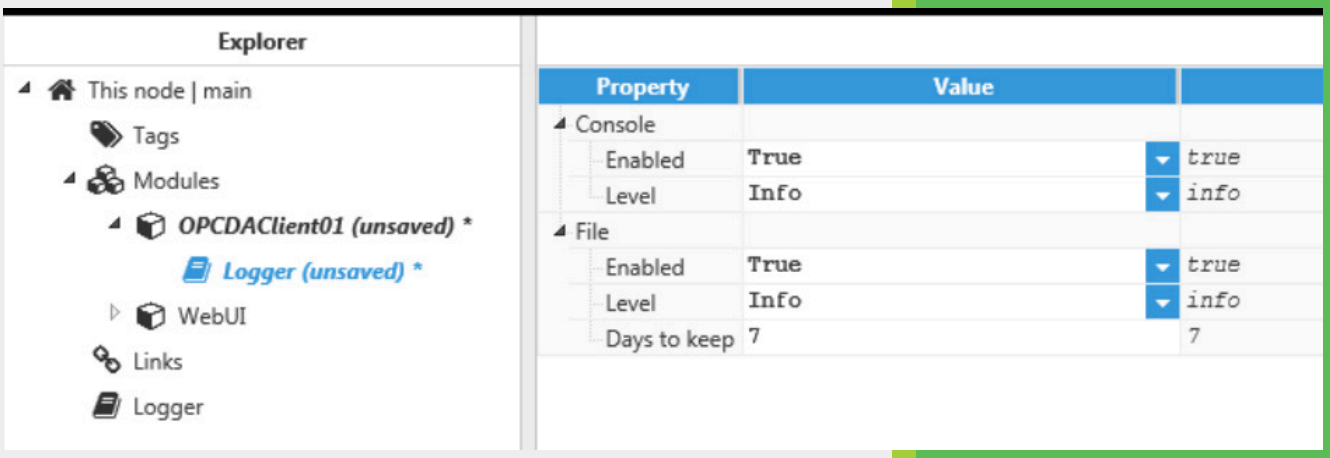
Step 1: New module creation

Step 2: Provide a name for the module (in this case OPCDAClient01), assign the module type (in this case OpcDaClient) and save the new configuration.



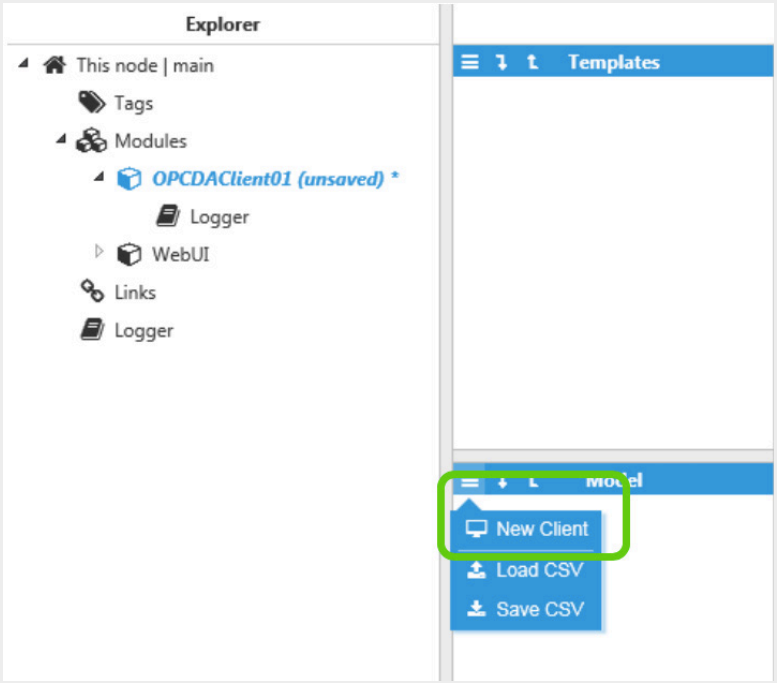
Step 2: Configuring a new module as OpcDaClient

Step 3: Configure the log (usually the default values are sufficient). Save the log configuration.



Step 3: Default log configuration

Step 4: Create the OPC DA connection to an OPC client and name it MatrikonOPCDA. Each connection is an independent OPC DA client.

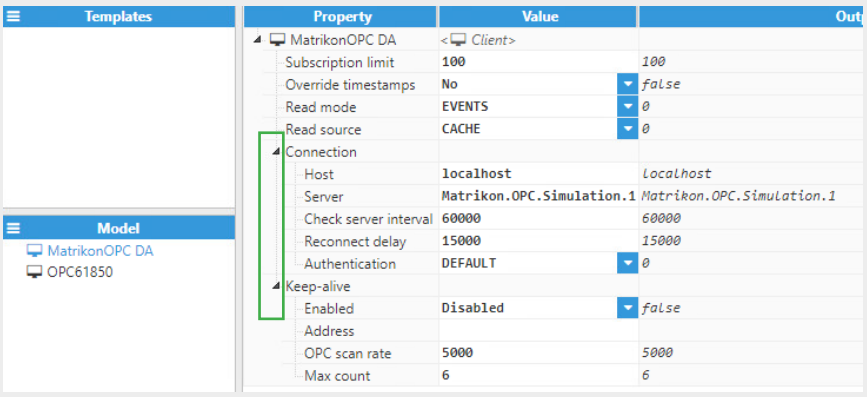


Step 4: Creating a new OPC DA server connection

Step 5: Configure the OPC DA client to connect to Matrikon.

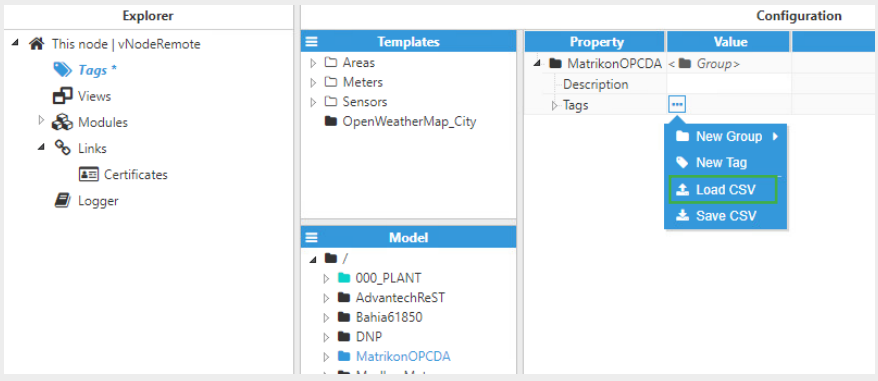
- **Host:** Enter the Hostname/IP address of the target OPC server. Empty means localhost.
- **Server:** Instance name of the target OPC server, also known as Prog ID. For Matrikon Simulator the instance name is “Matrikon.OPC.Simulation.1”.

Save the configuration and select restart the module.



Step 5:
OPC DA client configuration for connecting to Matrikon OPC in the same host.

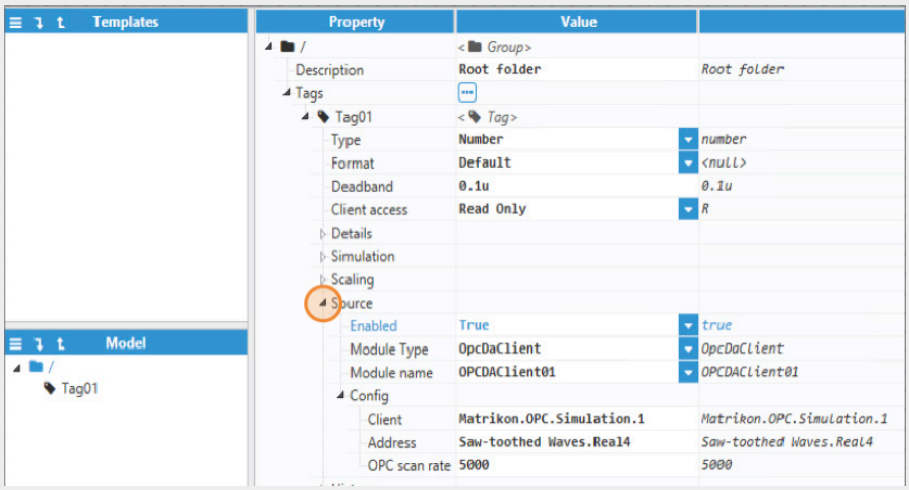
Step 6: Create a tag to connect to the OPC sever: Config => Tags => New Tag



Step 6:
New tag creation

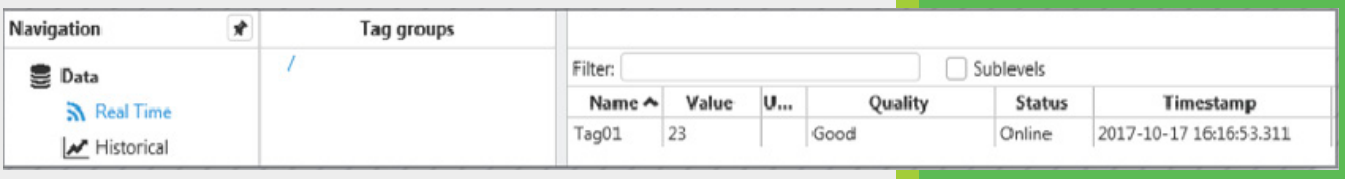
Step 7: Configure the tag. All details regarding the communication should be configured in Source entry:

- **Source.Enabled:** True
- **Source.Module Type:** OpcDaClient
- **Source.Module name:** OPCDAClient01 (the module created in previous steps)
- **Source.Config.Client:** MatrikonOPCDA (the connection created in previous steps)
- **Source.Config.Address:** The tag ID in the OPC server. In this example, a valid tag ID would be "Saw-toothed Waves.Real4" to retrieve simulated data from Matrikon Simulator.



Step 7: New tag configuration

Step 8: The tag should now be available, displaying as good quality in the Real-Time display.



Step 8: Tag value in real-time

Usually no changes are required in the DCOM since **vNode** runs as a service and the default configuration of Matrikon OPC permits the connection of local applications running as System.

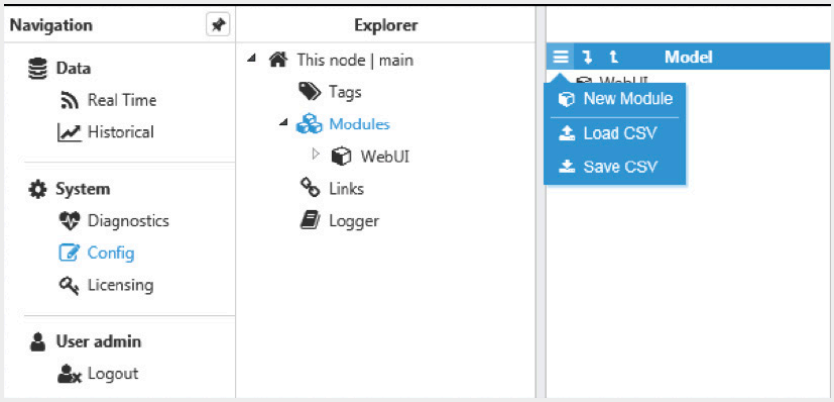
Log files for troubleshooting can be downloaded from the WebUI in Diagnostics => This node => Export logs button.

OPC UA client configuration

Example connection for KEPServerEX and TOPServer

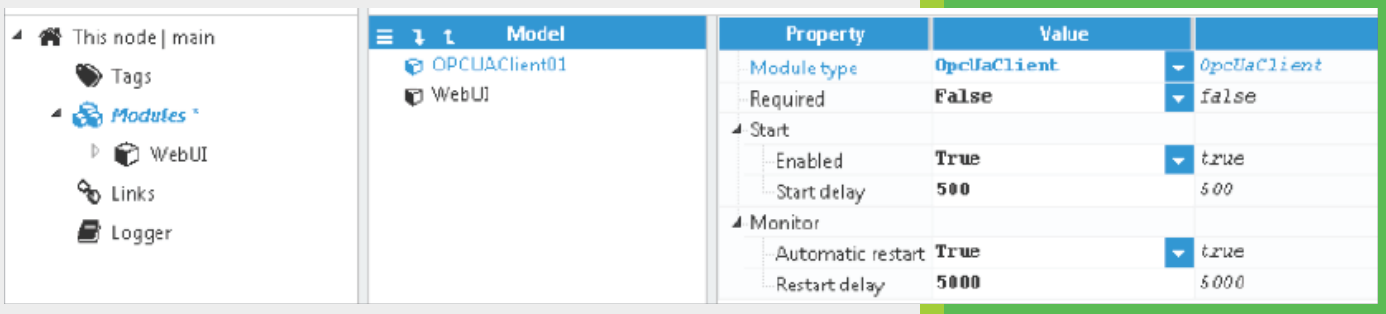
The following steps show how to connect to KEPServerEX and TOPServer using the OPC UA client.

Step 1: Create the module: (Config => Modules => button to the left of Modules => New module)



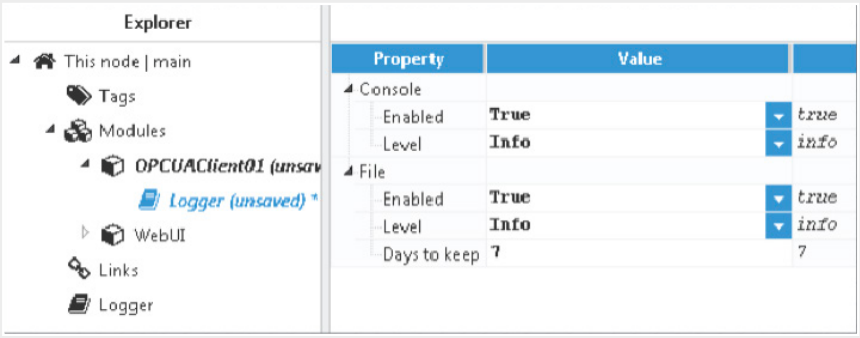
Step 1: New module creation

Step 2: Provide a name for the module (in this case OPCUAClient01), assign the module type (in this case OpcUaClient) and save the new configuration.



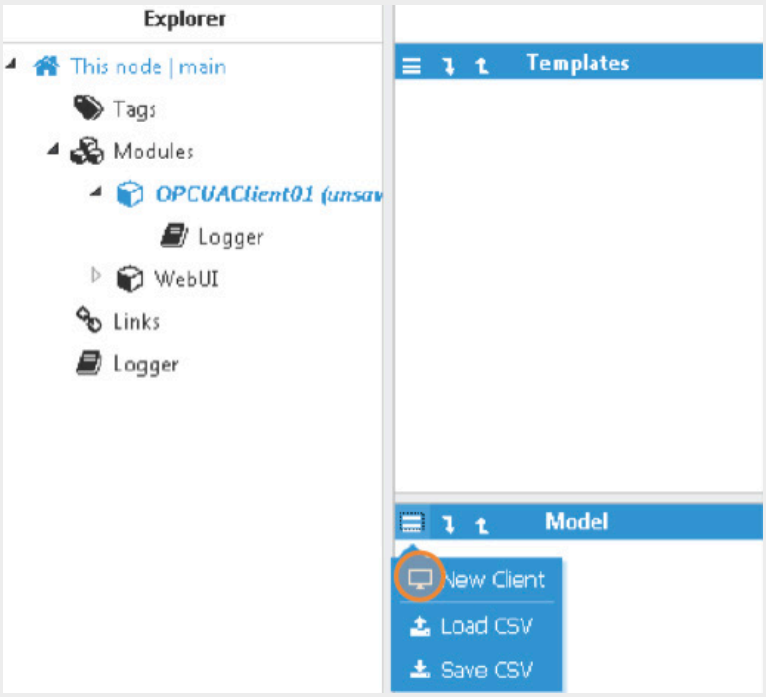
Step 2: Configuring the new module as OpcUaClient

Step 3: Configure the log (usually the default values are sufficient). Save the log configuration.



Step 3: Default log configuration

Step 4: Create the OPC UA connections. Each connection is an independent OPC UA client.

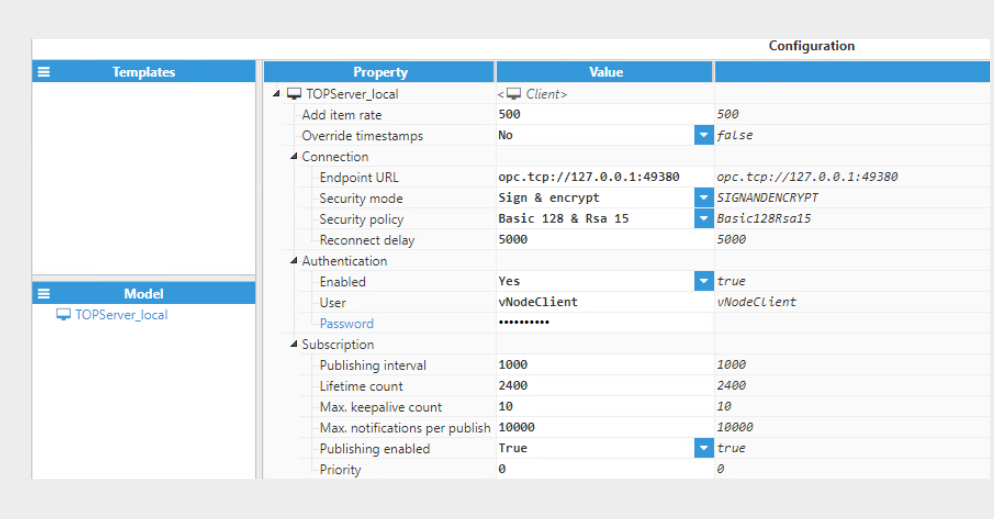


Step 4: Creation of a new OPC UA connection to a server

Step 5: Configure the OPC UA client to connect to KEPServerEX or TOPServer.

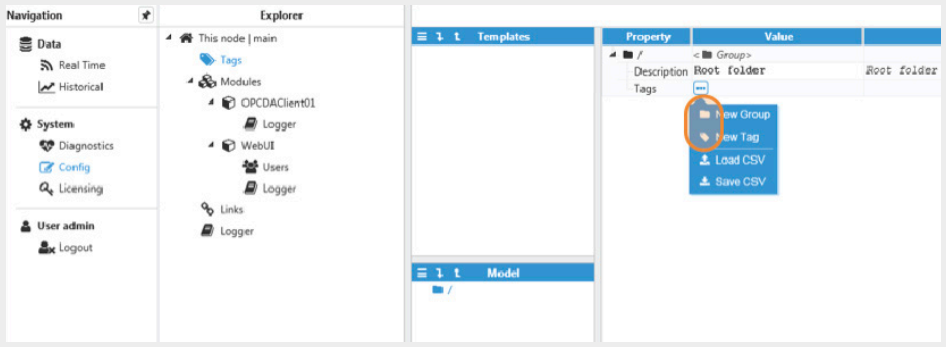
- **Connection:** The "Endpoint URL" is the Hostname/IP address of the target OPC server and the "port" with the format opc.tcp://endpointURL:port. Default OPC UA port for KEPServer is 49320 and for TOPServer is 49380
- **Authentication:** Permits user and password authentications to be enabled for server connections. If authentication is not enabled, the OPC UA server must permit anonymous login.
- **Subscription:** Permits configuration of the tag subscriptions in the OPC UA server.

Save the configuration and select restart the module.



Step 5:
OPC UA client configuration for connecting to TOPServer in the same host.

Step 6: Create a tag to connect to the OPC sever: Config => Tags => New Tag



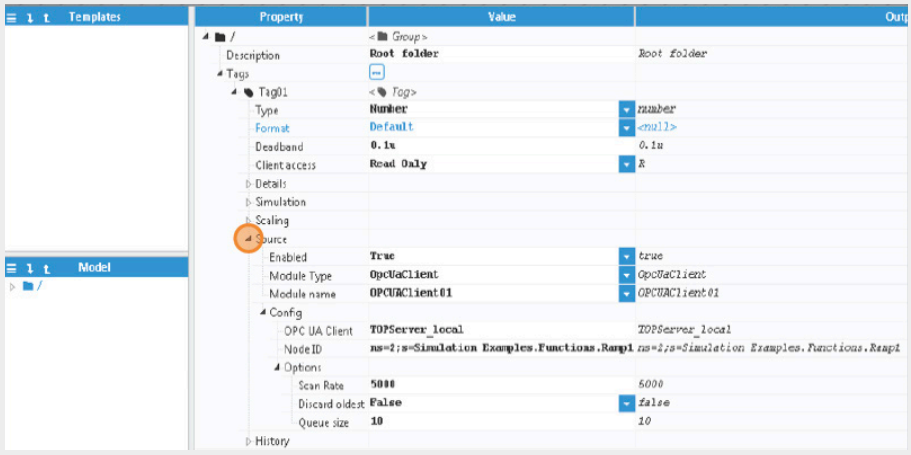
Step 6:
New tag creation

Step 7: Configure the tag. All details regarding the communication should be configured in Source entry:

- **Source.Enabled:** True
- **Source.Module Type:** OpcUaClient
- **Source.Module name:** OPCUAClient01 (the module created in previous steps)
- **Source.Config.OPC UA Client:** TOPServer_local (the connection created in previous steps)
- **source.Config.Node ID:** The Node ID in the OPC server including the Name Space Index (ns) and the Identifier separated by semi-colon. In this example, the ns would be 2 and a valid tag Identifier would be “Simulation Examples.Functions.Ramp1” so the Node ID is:

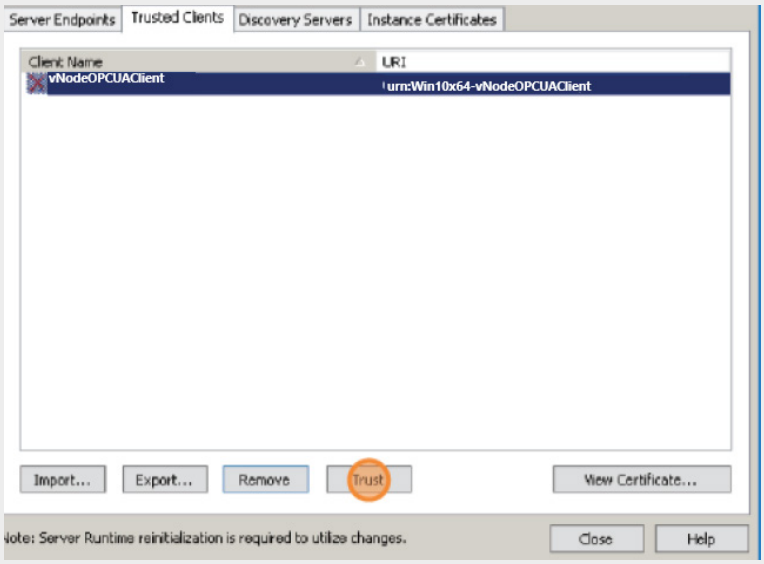
ns=2;s=Simulation Examples.Functions.Ramp1

Save the tag configuration



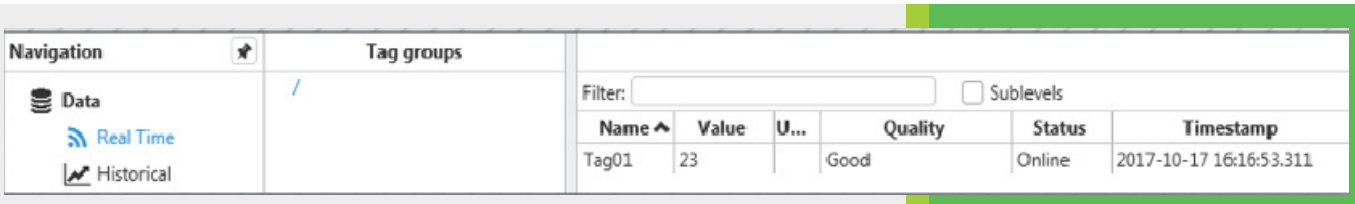
Step 7: New tag configuration

Step 8: Trust **vNode** OpcUaClient certificate in KEPServer/TOPServer OPC UA Configuration Manager.



Step 8: Digital Certificate trust process

Step 9: The tag should now be available, displaying as good quality in the Real-Time display.

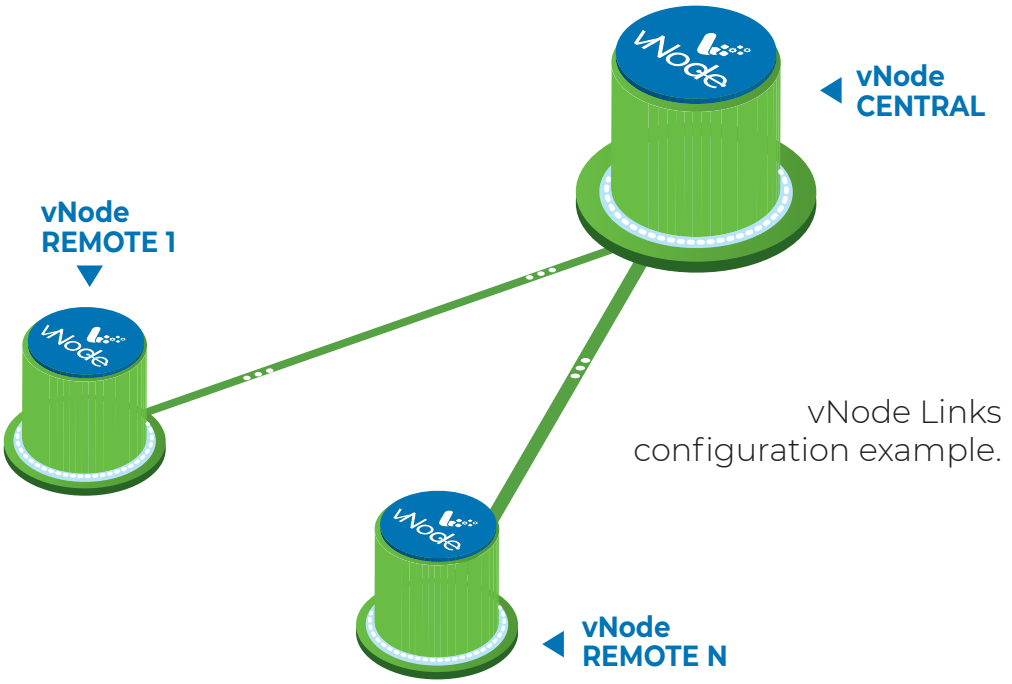


Step9: Tag value in real-time display

Log files for troubleshooting can be downloaded from the WebUI in Diagnostics => This node => Export logs button.

Example of Link Configuration

The following example involves 4 nodes: 2 nodes to initiate the connection (remote Nodes) to the other 2 nodes which are receiving the connection (Central nodes). In this way, both Central nodes will subscribe to the data collected by Remote nodes.

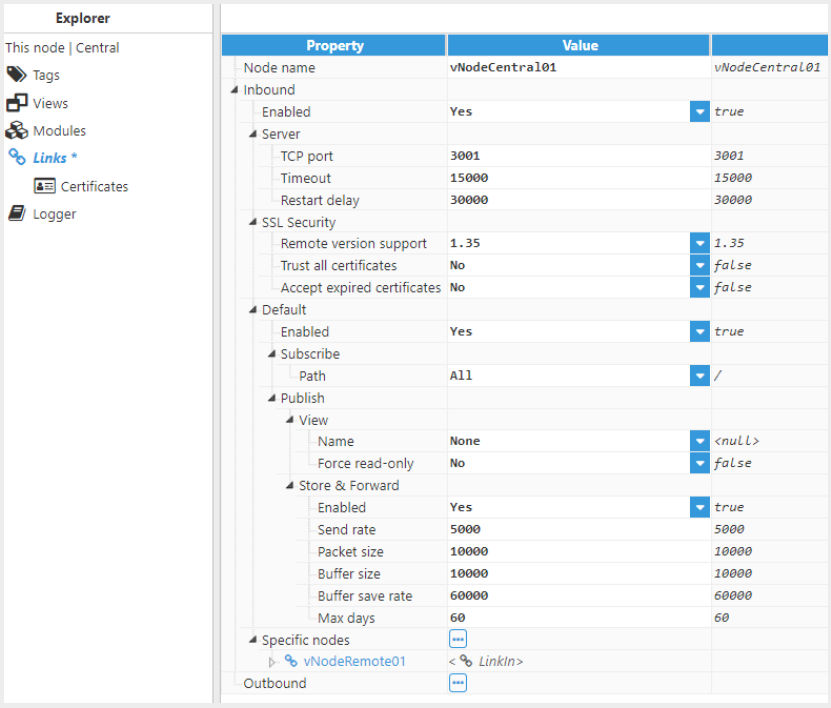


vNode Links configuration example.

This architecture is created by completing the following steps:

Step 1: Configure vNodeCentral01 and vNodeCentral02 to receive incoming connections (Config => Links).

- Provide the name (vNodeCentral01 and vNodeCentral02). This name must be unique among all connected nodes.
- Enable incoming connections and configure the listening port (3001 by default).
- Create a specific connection for remote nodes vNodeRemote01 and vNodeRemote02 by setting tag subscription as “All” and publish view as none. This means that these nodes are now subscribed to all available tags in the remote nodes but won’t be publishing any data to these remote nodes.

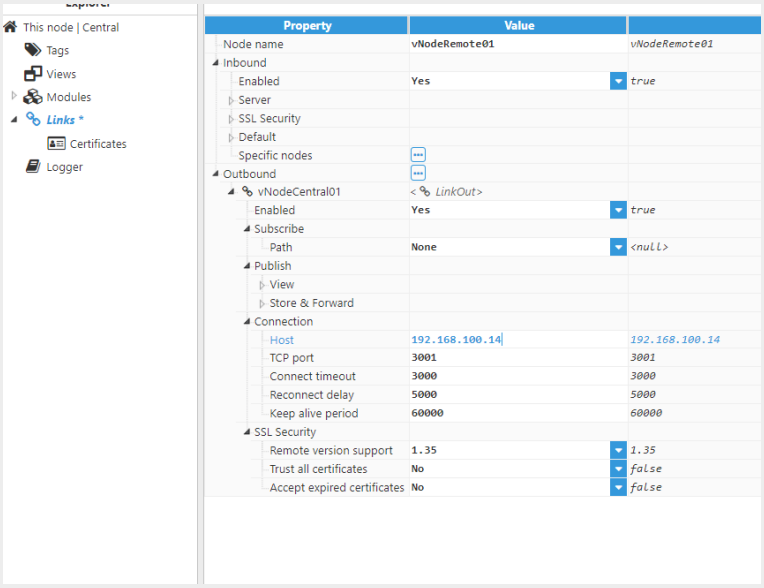


Step 1: Configuration of central nodes to receive connections from other nodes

Step 2: Configure vNodeRemote01 and vNodeRemote02 to provide data to the central servers (Config => Links).

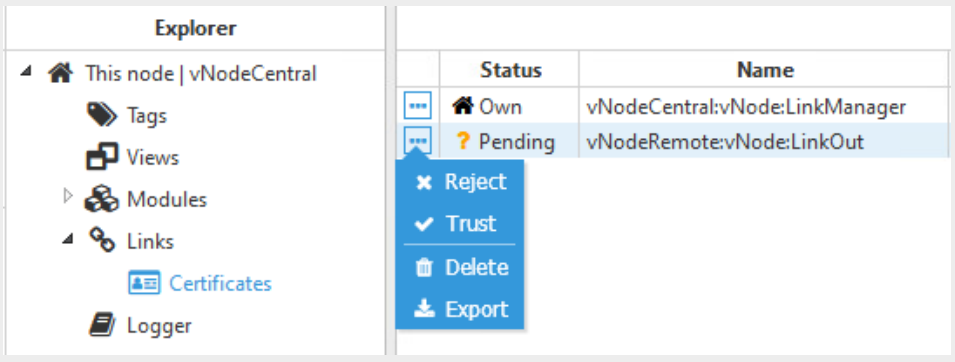
•Provide the name (vNodeRemote01 and vNodeRemote02). This name must be unique among all connected nodes.
•Add the outbound connections to the central servers. The name of each outbound connection must match exactly with the name of the destination central server.

- Configure tag subscription as “None” since the remote nodes won’t be subscribed to the tags in the central nodes.
- Configure the Publish view as “Full model” in order to push all local events to the remote nodes.
- Configure the connection details, IP, and port, for the central servers.



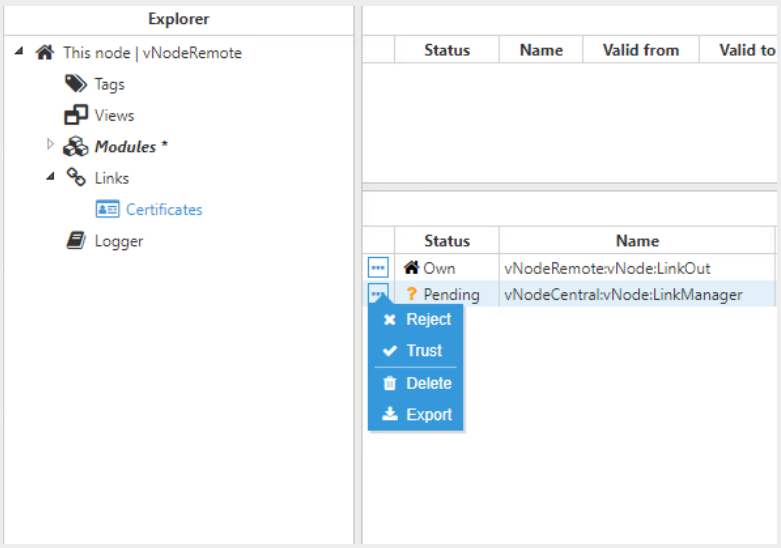
Step 2: Outbound connections configuration for vNodeRemote01

Once the configuration of all nodes is complete, the remote nodes will send a digital certificate to the central servers. All certificates must be set as "trusted" in the central servers.



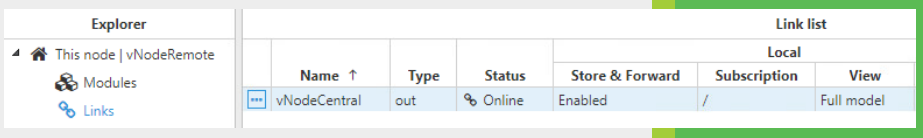
Step 2: Digital certificates of the remote nodes in the central node

Once the digital certificates for the remote nodes have been “trusted” by the central server, the digital certificate from the central servers must also be “trusted” by the remote nodes.



Step 2: Digital certificates of the central nodes in the remote node

Once all digital certificates have been trusted, the connection has been successfully established and the link status will be displayed in Diagnostics => Links to show that these remote tags are now available in the central nodes.

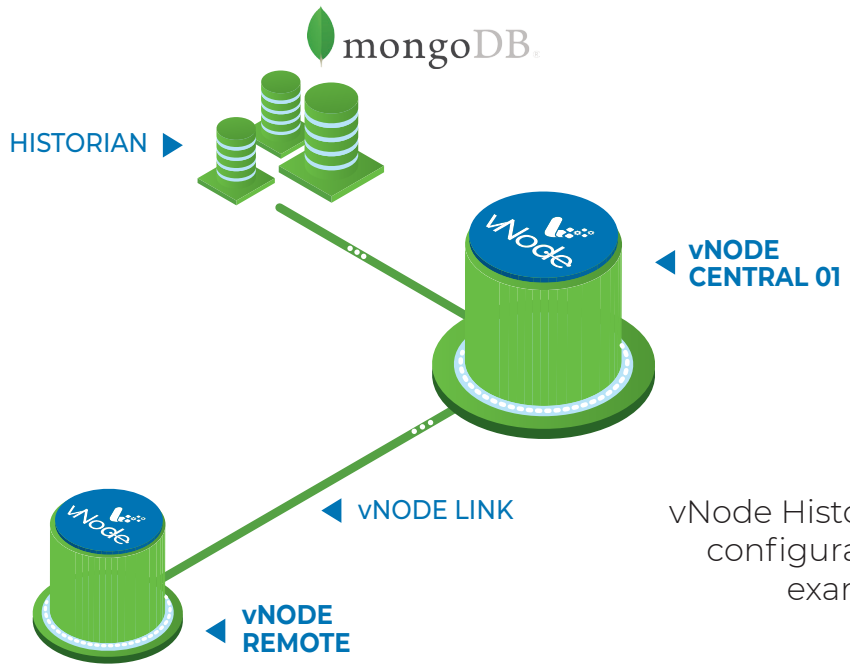


Step 2: Status of the links in the remote node

Step 2: Status of the links in the central node

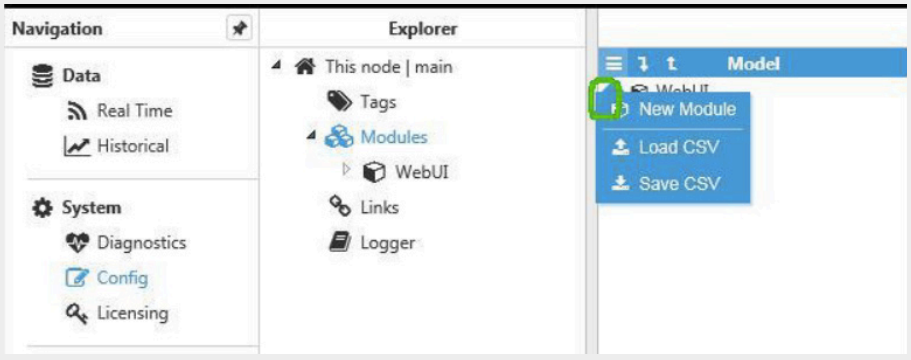
Historian Configuration Example

The following example involves two nodes: a remote node collecting and sending data through a **vNode** Link to a second node that is running Historian.



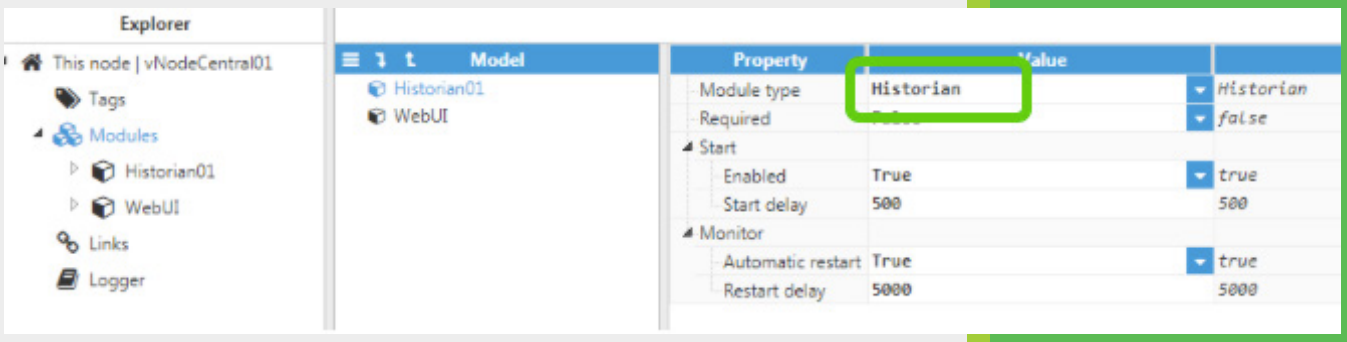
vNode Historian configuration example

Step 1: Create the module: (Config => Modules => button to the left of Model => New module)



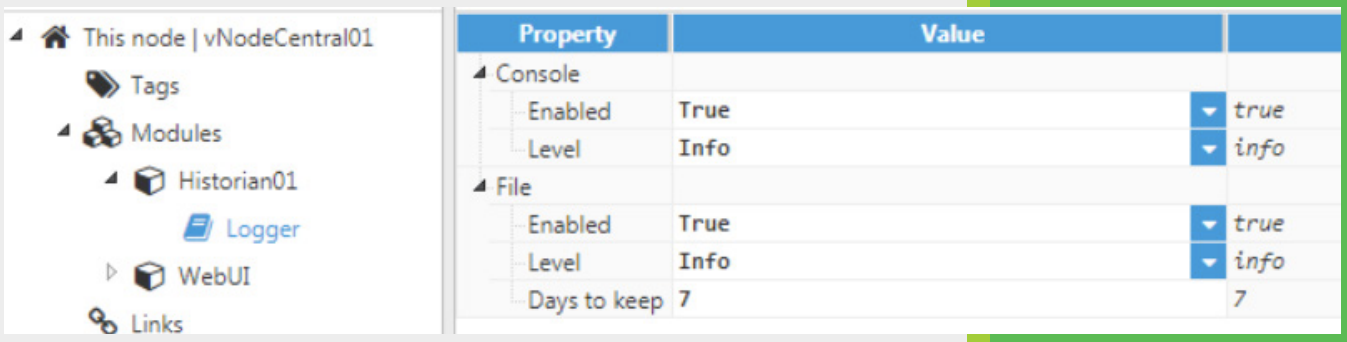
Step 1

Step 2: Provide a name for the module (in this case Historian01), assign the module type (in this case Historian) and save the new configuration.



Step 2: Configuring a new module as Historian

Step 3: Configure the log (usually the default values are sufficient). Save the log configuration.

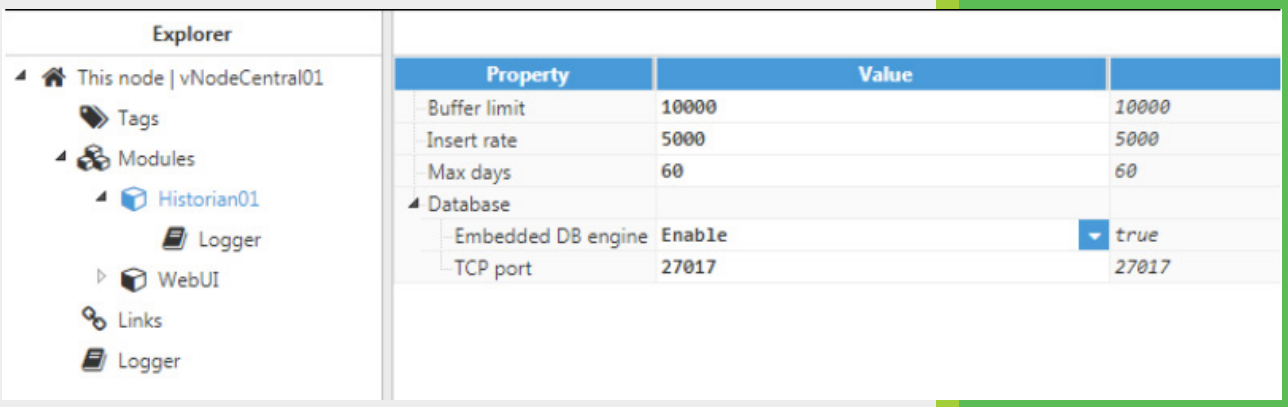


Step 3: Default log configuration

Step 4: Configure the historian instance.

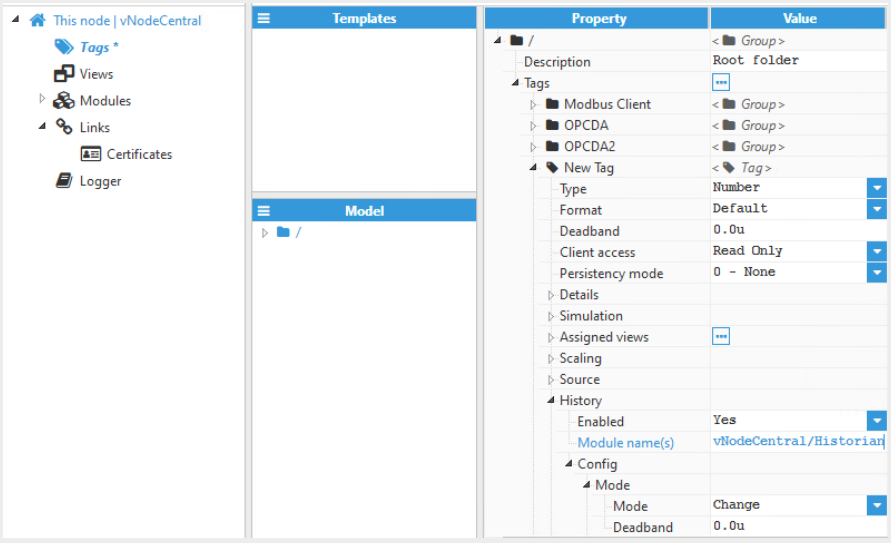
- **Buffer limit:** displays the maximum size of the event's buffer in kilobytes.
- **Insert rate:** displays the period for sending the event's buffer to the database.
- **Max days:** displays the maximum number of days data will be stored in the database. Data older than this will be automatically pruned.

  - **Embedded DB engine:** enables the use of the MongoDB instance, which is automatically created by **vNode**. In order to use a MongoDB that has been installed by the user, this parameter must be disabled.
  - **TCP port:** for connecting to MongoDB, both if for an instance created by **vNode** or a MongoDB installed by the user.



Step 4: Historian default configuration

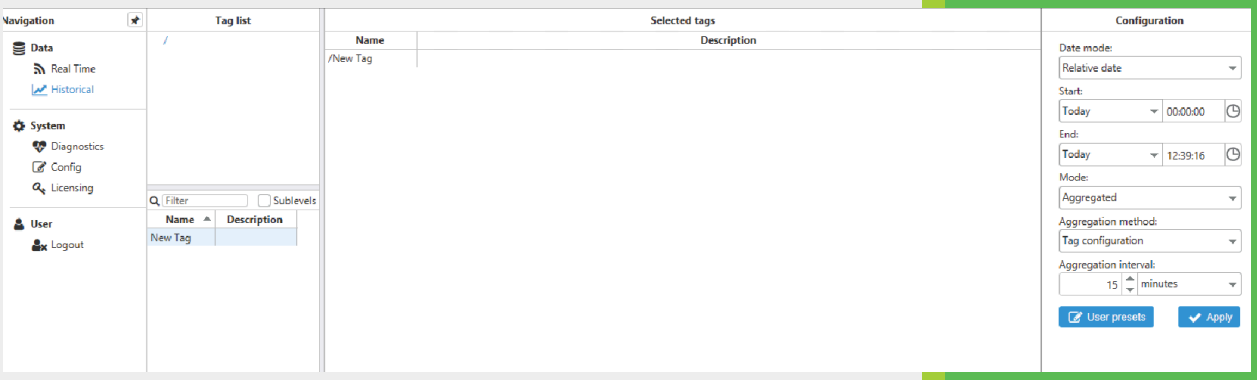
Step 5: In the remote node vNodeRemote01, create a new tag and navigate to the History section to enable the historization of the tag pointing to the Historian instance in node vNodeCentral01. Both nodes (vNodeRemote01 and vNodeCentral01) must already be linked.



Step 5: Tag History configuration

If the Historian instance was running locally in the vNodeRemote01 node, then the Module name in the History configuration of the tag should be vNodeRemote01/Historian01 or simply Historian01.

Step 6: Values stored in the database can be visualized in a chart within the WebUI of both the central and remote node, regardless of where the database is located.



Step 6: Example of tag picking

Select the tags, start and end date, retrieval mode and click apply.

Contact us:

Main office:
United States,
1549 NE 123St,
North Miami, FL
ZIP 33161.
+1 (754) 755-0009

Regional offices:
United Kingdom:
(+44) 161 660 3241
Spain:
(+34) 93 572 1007
Mexico:
(+52) 55 46282593
France:
+33 (0)4 13 68 01 06
Costa Rica:
(+506) 2225 2344

Email:
info@vnodeautomation.com.
support@vnodeautomation.com
sales@vnodeautomation.com,
saleseurope@vnodeautomation.com
supporteurope@vnodeautomation.com

Website:
vnodeautomation.com