United Electronic Industries

The High-Performance Alternative

UEIOPC-UA User Manual

January 2018 Edition

United Electronic Industries, Inc.
27 Renmar Avenue
Walpole, MA 02081

www.ueidaq.com
508.921.4600

Contents

www.ueidaq.com
508.921.4600

www.ueidaq.com
508.921.4600

1 Introduction

This document provides documentation for the UEIOPC-UA device.

1.1 OPC-UA Specification

The OPC-UA specification is a multi-part specification that defines a number of functionalities such as security, services or information models. Those features are grouped in profiles and facets.

A facet is a type of profile that describes a group of functionality. However, it doesn't provide sufficient functionality to produce a working server or client.

OPC-UA profiles can be categorized as:

- Server profiles
- Client profiles
- Transport profiles
- Security profiles

For example an OPC-UA server must implement at least one fully featured server profile that must include at least one transport profile and one security profile. Facets define additional functionality (such as event subscription).

Fully featured profiles already contain facets, but additional facets can be added to extend the functionality supplied by the server.

Some facets are mandatory, and some facets are optional.

Profiles and facets enable client and server developers to choose which part of the specification they want to support.

The OPC foundation publishes a description of all profiles and facets defined on its website: http://www.opcfoundation.org/profilereporting/index.htm.

www.ueidaq.com
508.921.4600

1.2 UEIOPC-UA supported profiles

UEIOPC-UA supports the following profiles and facets.

Profile/Facet Type	Profile/Facet Supported
Server Profile	Embedded UA Server profile
Transport Profile	UA-TCP UA-SC UA Binary
Security Profiles	SecurityPolicy – Basic256Sha256 SecurityPolicy – Basic256 SecurityPolicy – None
Facets	DataAccess Server facet Historical Raw Data Server Facet Historical Data AtTime Server Facet Historical Access Modified Data Server Facet Historical Data Insert Server Facet Historical Data Update Server Facet Historical Data Replace Server Facet Historical Data Delete Server Facet

1.3 UEIOPC-UA architecture

UEIOPC-UA extends the capability of the PowerDNA distributed data acquisition platform by turning it into an OPC-UA server that can be accessed by any OPC-UA software client.

UEIOPC-UA is derived from the UEIPAC and runs a Linux embedded operating system.

The interface between the UEIOPC-UA analog, digital, or serial I/O cards and OPC-UA clients is handled by an OPC-UA server process that is automatically started when the unit is powered up.

www.ueidaq.com
508.921.4600

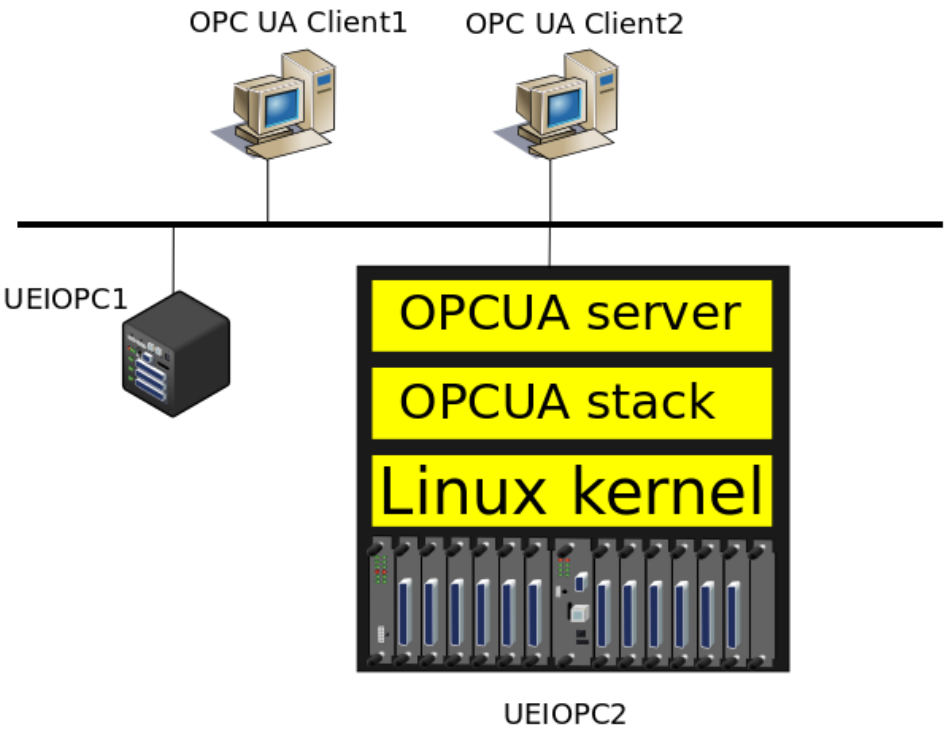


Figure 1 UEIOPC-UA Architecture

2 Configuring the UEIOPC-UA

UEIOPC-UA configuration consists of setting up the IP address and the I/O channels that you want to make available to OPC-UA clients.

- The IP address must be configured using the serial port.
- The I/O channels are configured in the “/etc/ueiopc.xml” file, located on the root file system (SD card). Channels can be configured using the UEIOPC-UA web UI or by editing the “/etc/ueiopc.xml” file manually.

You can also change OPC-UA security policies.

OPC-UA network and security parameters are configured in “/etc/opcua/settings.conf”.

2.1 Connecting through the serial port

To communicate with the UEIOPC-UA over the serial port, first connect the serial cable to the serial port on the UEIOPC-UA and the serial port on your PC.

You will also need a serial communication program:

- Windows: ucon, MTTTY or HyperTerminal.
- Linux: minicom or cu (part of the uucp package).

The UEIOPC-UA uses the serial port settings: 57600 bits/s, 8 data bits, 1 stop bit and no parity. Run your serial terminal program and configure the serial communication settings accordingly.

After configuring your serial terminal program, connect the DC output of the power supply (24VDC) to the “Power In” connector on the UEIOPC-UA and connect the AC input on the power supply to an AC power source.

You should see a message similar to the following on your screen:

```
U-Boot 1.1.3 (PowerDNA 8347 3.2.3) (Jan  3 2014 - 16:31:32)
MPC83XX

Clock configuration:
  Coherent System Bus:  264 MHz
  Core:                 396 MHz
  Local Bus Controller: 264 MHz
  Local Bus:             66 MHz
  DDR:                  264 MHz
  I2C:                  264 MHz
  TSEC1:                264 MHz
  TSEC2:                264 MHz
```


www.ueidaq.com
508.921.4600

```
  USB MPH:                88 MHz
  USB DR:                 88 MHz
CPU: MPC834x, Rev: 80830031
Rev: 31 at 396 MHz

<lots of uboot  messages>

Hit any key to stop autoboot:  5

## Booting image at fe000000 ...
   Image Name:   Linux-3.4.6-ueipac834x
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    1833524 Bytes =  1.7 MB
   Load Address: 00400000
   Entry Point:  00400568
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK...
< lots of kernel messages >
...
BusyBox v1.21.0 (2015-06-01 07:22:14 EDT) built-in shell (ash)
Enter 'help' for a list of built-in commands.
~ #
```

Once you see the Linux command line prompt, you can navigate the file system and enter standard Linux commands, such as `ls`, `ps`, and `cd`.

2.2 Configuring the IP address

Each UEIOPC-UA is equipped with dual Ethernet ports. The primary NIC port is configured at the factory with the IP address set to 192.168.100.2 to be part of a private network.

NIC1 is the primary Ethernet port for UEIOPC-UA models based on the 8347 CPU (UEIOPCUA 300/600-1G, UEIOPCUA 600/1200R, UEINET-OPC-UA, or -MIL models), and NIC Out is the primary for UEIOPC-UA models based on the 5200 CPU (UEIOPCUA 300/600).

You can change the IP address for the current session by typing the following command in the serial terminal:

```
setip <new IP address>
```

The `setip` command changes the current IP address and stores it in a file so that the new IP address will be used next time the system is powered-up.

To test the IP address is correctly set, use the `ping` command from your host PC.

You can now disconnect the serial port and configure the UEIOPC-UA by opening a web browser to the URL:
`http://<UEIOPC-UA IP address>/ueiopc.html`

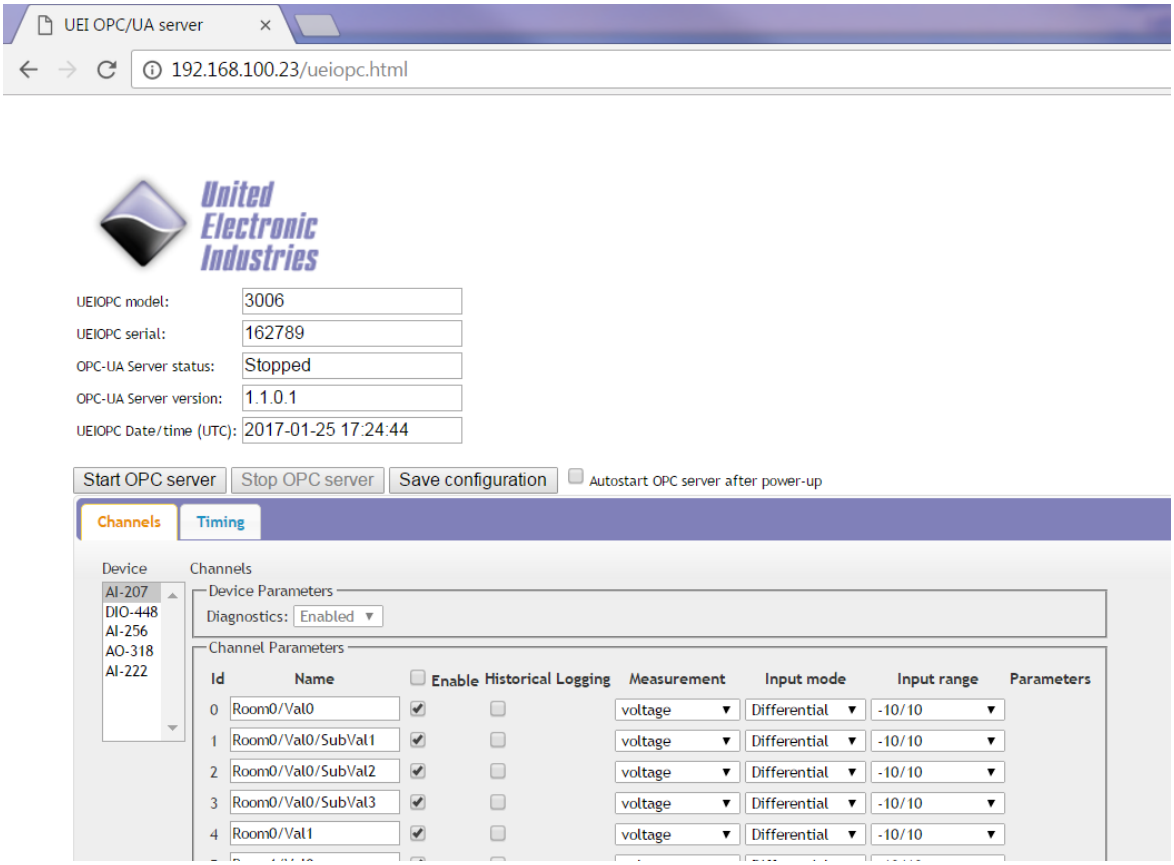


Figure 2 UEIOPC-UA Web UI

We recommend using Firefox, Google Chrome or Safari as the web browser when configuring the UEIOPC-UA. Microsoft Internet Explorer is not supported.

2.3 Configuring OPC-UA port and endpoint URL

In OPC-UA, servers are identified though endpoints which closely resemble a URL.

The default endpoint URL is **opc.tcp://<UEIOPC-UA IP address>:48020**

You can edit “/etc/opcua/settings.conf” to change the UEIOPC-UA endpoint.

www.ueidaq.com
508.921.4600

2.4 Starting/Stopping OPC-UA service

The UEIOPC-UA is pre-configured to automatically start the OPC-UA service at boot time. You must stop and restart the service after changing the server or channel configuration.

2.4.1 Starting/Stopping service using the web UI

To start the OPC-UA server, click the **Start OPC server** button. The status indicator should show that the server is in the “Running” state.

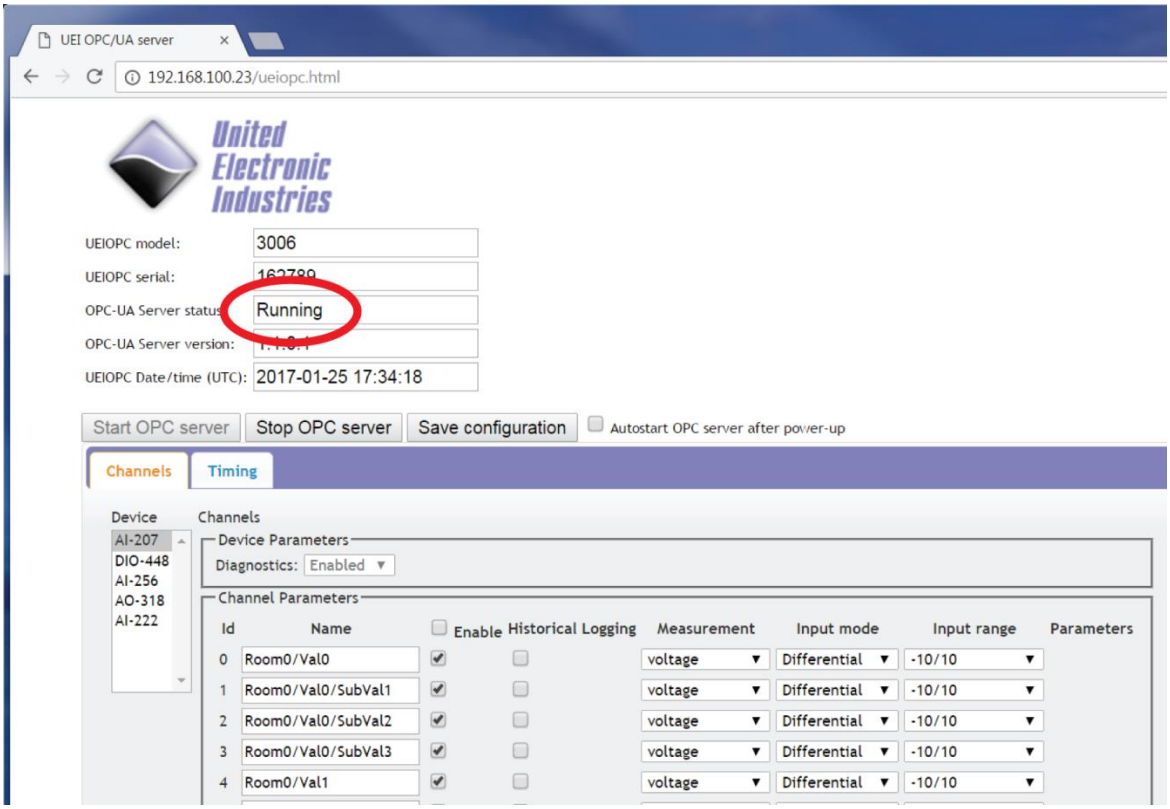


Figure 3 UEIOPC-UA Server Running

To stop the server, click the **Stop OPC server** button.

To automatically start the OPC-UA service at boot time, check **Autostart OPC server after power-up,** and click **Save Configuration**.

www.ueidaq.com
508.921.4600

2.4.2 Starting/Stopping service using command line

Use the following command to stop the OPC-UA service:

```
/etc/init.d/ueiopc stop
```

Use the following command to start the OPC-UA service:

```
/etc/init.d/ueiopc start
```

Use the following command to restart the OPC-UA service:

```
/etc/init.d/ueiopc restart
```

Use the following commands to disable automatic start of OPC-UA service:

```
rw
mv /etc/rc.d/S40ueiopc /etc/rc.d/K40ueiopc
ro
```

Use the following command to enable automatic start of OPC-UA service:

```
rw
mv /etc/rc.d/K40ueiopc /etc/rc.d/S40ueiopc
ro
```

2.5 Date and time configuration

OPC-UA specifies that each data point comes with a value, a quality status, a source timestamp and a server timestamp.

The source timestamp is the time at which the value was updated by the I/O hardware.

The server timestamp is the time at which the server was notified of the update.

Both timestamps are identical on UEIOPC-UA.

The UEIOPC-UA on-board RTC (real-time clock) is used to produce the timestamp for each data point. The RTC is backed by a battery.

You can either set the date and time manually, or you can configure the UEIOPC-UA to automatically synchronize its clock with an NTP server over the network.

2.5.1 NTP configuration

2.5.1.1 NTP configuration using the web UI

This feature is not implemented on the web UI.

2.5.1.2 NTP configuration using the command line

To synchronize the UEIOPC-UA with the NTP server, first verify that you can `ping` your NTP server.

NOTE: Editing files will require you to make the UEIOPC-UA system read-write. Before editing, type `rw`, and after editing type `ro` to make the system read-only again.

Edit the "/etc/init.d/ntp" file and set the **NTP_SERVER** variable to the IP address or host name of your NTP server.

Use the following command to stop the NTP service:

```
/etc/init.d/ntp stop
```

Use the following command to start the NTP service:

```
/etc/init.d/ntp start
```

Use the following command to restart the NTP service:

```
/etc/init.d/ntp restart
```

Use the following commands to disable automatic start of NTP service:

```
rw
mv /etc/rc.d/S50ntp /etc/rc.d/K50ntp
ro
```

Use the following commands to enable automatic start of NTP service:

```
rw
mv /etc/rc.d/K50ntp /etc/rc.d/S50ntp
ro
```

2.5.2 Setting local date and time

The UEIOPC-UA is equipped with a real-time clock (RTC) chip that preserves the date and time settings when the UEIOPC-UA is not powered.

By default, the date is set to the current data and time in the UTC (GMT) time zone.

You can use the web UI or the command line to set the date and time.

2.5.2.1 Setting date and time with the web UI

The UEIOPC date and time fields display the current UTC date and time programmed on the UEIOPC-UA on-board clock. If those displays do not refresh automatically, refresh your web browser to update the display.

Click **Set UTC Date/Time from host and save** to set the UEIOPC-UA RTC date and time to the same value as your host PC.

You can also manually enter a new date and time in the **New Date** and **New Time** controls, and click **Set UTC date/time and save** to save the new date and time to the UEIOPC-UA RTC.

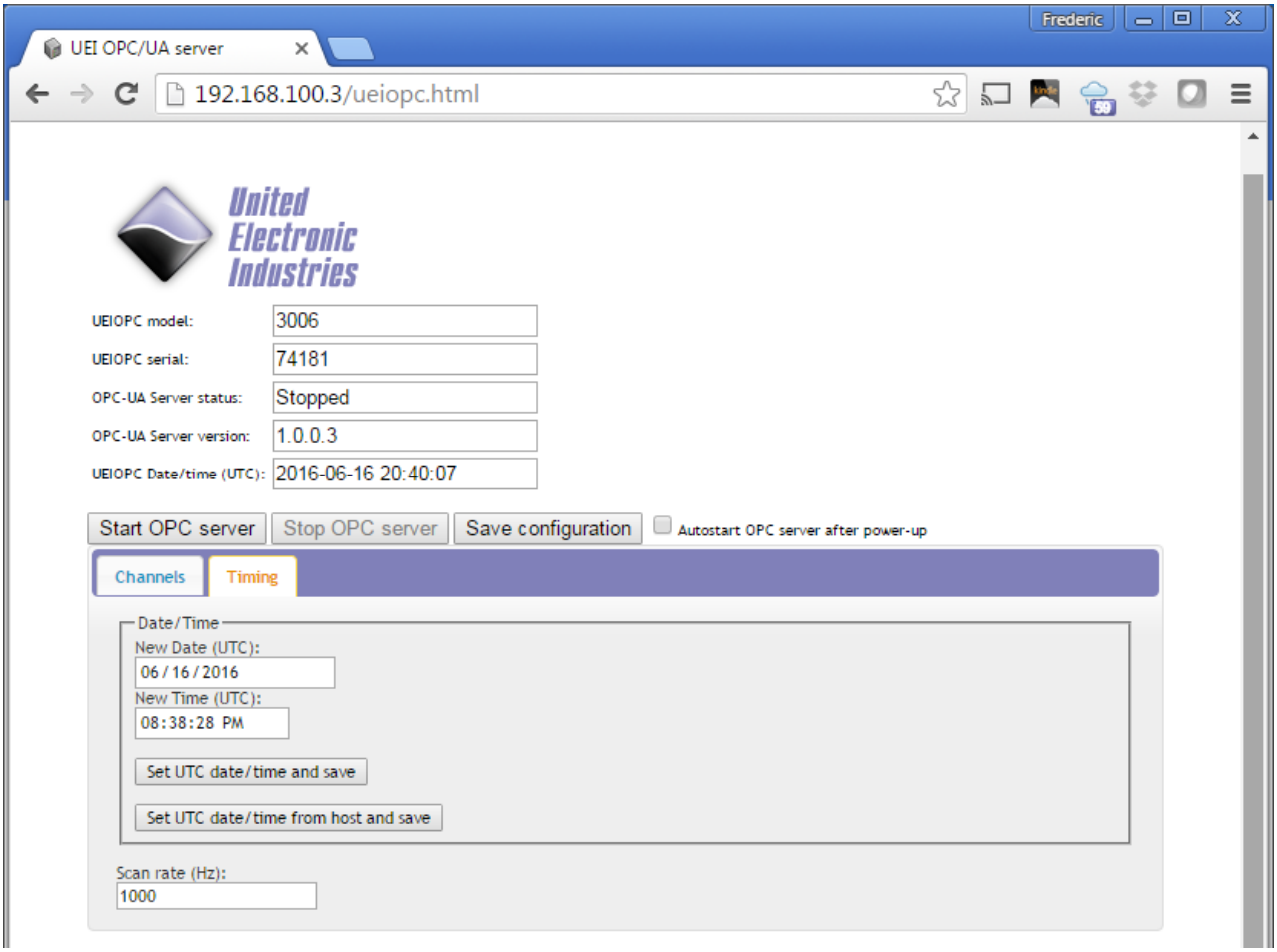


Figure 4 Setting Time and Date via the UEIOPC-UA Web UI

2.5.2.2 Setting date and time with the command line

To print the current date and time, use the following command:

```
date
```

To change the current date and time, use one of the following commands:

```
date –s MMDDhhmm
date –s YYYYMMDDhhmm.ss
```

For example “date –s 06021405” will set the new date to June second, 2:05 PM.

To make this change permanent upon reboot, save the date to the RTC chip with the following command:

```
hwclock –w –u
```

2.5.3 Changing the time zone

To set the time zone, you need to set the environment variable **TZ**.

For example the command below sets the time zone to eastern time with daylight saving time starting on the Sunday(0) of the second week(2) of March(3) and ending on Sunday(0) of the first week(1) of November(11):

```
export TZ=EST5EDT,M3.2.0,M11.1.0
```

NOTE: Editing files will require you to make the UEIOPC-UA system read-write. Before editing, type `rw`, and after editing type `ro` to make the system read-only again.

To make this change permanent upon reboot, add the command to the file “/etc/profile”.

You can find a detailed explanation of the syntax of **TZ** at:
http://www.gnu.org/software/libtool/manual/libc/TZ-Variable.html

www.ueidaq.com
508.921.4600

3 Configuring the address space

The Address Space is how an OPC-UA Server represents its functionality to clients. Inputs, outputs and data in a device are represented hierarchically in the address space.

Figure 5 shows an example of how the address space of UEIOPC-UA displays in a client system.

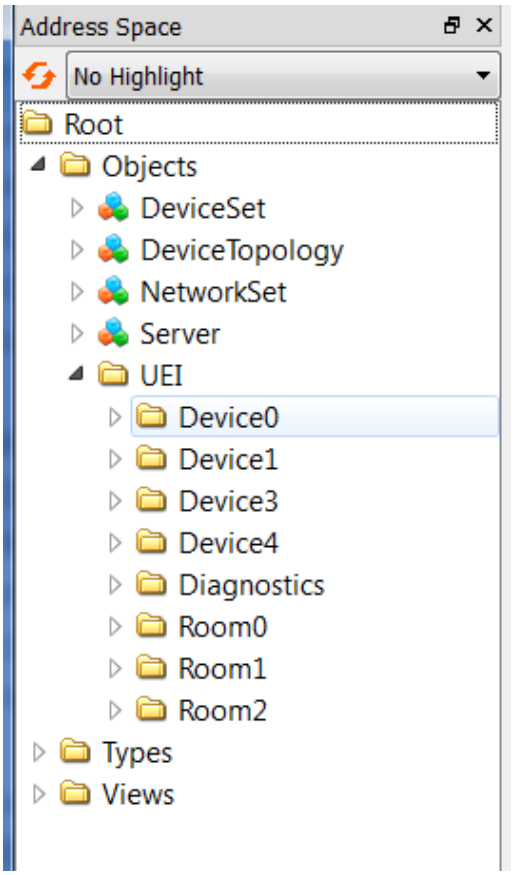


Figure 5 UEIOPC-UA I/O Channels Displayed in the Client Address Space

3.1 Server Object

The Server object is part of the OPC-UA standard information model.
It is implemented the same way on all OPC-UA servers.

The Server object provides a standard way to get status, diagnostic, capability and configuration information.

www.ueidaq.com
508.921.4600

- **Auditing**: A Boolean indicating if the Server is generating auditing events. This contains data such as who did what and when in terms of configuration, security or data updates.

- **NamespaceArray**: A table of uniform resource identifiers (URIs) for the various namespaces used by nodes in the Address Space. Index 0 in the array is the OPC Foundation namespace. Index 1 is the local OPC-UA Server, while index 2 and above reference other organizations that are responsible for the definition of Nodes used in the Address Space.

- **ServerArray:** A set of pointers to remote OPC-UA Servers that are referenced in the Address Space. UEIOPC-UA doesn't offer this capability.

- **ServerCapabilities:** This object describes the capabilities supported by the OPC-UA Server. The ServerCapabilities object contains the list of profiles supported by the Server, the list of signed software certificates from certification testing, the local IDs used for supporting multiple languages, and any number of other operational data variables.

- **ServerConfiguration:** This object provides security configuration capabilities such as updating the server certificate or configuration of the server trust list.

- **ServerDiagnostics:** This object contains items like session count, view count, session timeout, subscription counts and other counters that assist in troubleshooting Server operation.

- **ServerRedundancy:** This object describes the redundancy capabilities provided by the Server. This Object is required even if the Server does not provide any redundancy support. UEIOPC-UA doesn't provide redundancy.

- **ServerStatus:** The operational status of the Server, including build information such as manufacturer, product codes and software version.

- **ServiceLevel**: A quality of service level from 0 to 255 (best) that Clients can use to judge the relative reliability of Servers in a redundant Server network.

- **VendorServerInfo:** This object exists to allow vendors to extend Server information by adding additional proprietary information to the Server Object. Vendors can subtype the Object Type and create their own type where they can store additional useful information that is particular to their Server implementation and application.

3.2 UEI Folder

The UEI folder in the OPC-UA address space contains the mapping of the physical I/O channels in the address space. You can arbitrarily organize the channels in a way that best fits your application. Refer to Figure 6 on page 15.

I/O channel parameters are stored in an XML configuration file. The OPC-UA server opens and parses the configuration file at boot time and then configures and starts the I/O boards.

The UEI folder also contains a Diagnostics folder that provides chassis and select I/O board diagnostic information.

3.2.1 Diagnostic readings

The Diagnostics folder lists diagnostic readings, such as temperature and internal voltage references, for any I/O board designated as a Guardian series board and for the PowerDNA Power and NIC boards installed in UEIOPC-UA models based on the 8347 CPU (i.e., UEIOPCUA 600/1200R, UEIOPCUA 300/600-1G, UEINET-OPC-UA, and –MIL models).

Diagnostic data is read at approximately a 1 Hz scan rate. This scan rate is fixed and is different than the scan rate set for acquiring and storing samples (section 3.2.3.2).

Diagnostic information provided by the NIC and Power boards is listed in the tables below. Available diagnostic data includes voltage readings (in Volts), current readings (in Amps), and temperature readings (in Kelvins).

Power Diagnostics	NIC Diagnostics
2.5 V supply for slots 1-3	2.5 V supply for slots 4-6
2.5 V supply for slots 4-6	Ground reference
3.3 V supply for slots 1-3	3.3 V supply for slots 4-6
3.3 V supply for slots 4-6	24 V supply for slots 4-6
24 V supply for slots 1-3	1.5 V supply for slots 1-6
24 V supply for slots 4-6	1.2 V supply for slots 1-6
Input Voltage	3.3 V current for slots 4-6
1.5 V supply for slots 1-6	Temperature 1
1.2 V supply for slots 1-6	Temperature 2
Fan Voltage	
Input Current	
Temperature 1	
Temperature 2	

Please refer to the datasheets and/or user manuals for individual I/O boards to learn what diagnostic information that specific Guardian boards provide.

www.ueidaq.com
508.921.4600

3.2.2 Channel naming

Channels are organized as a hierarchical tree in the OPC-UA address space.

The channel name is the absolute path of the OPC-UA node associated with the channel in the OPC-UA address space. UEIOPC-UA uses the '/' character as a separator.

For example, the following list represents the channel names of an AI device as defined in the UEIOPC-UA server:

Room0/Val0
Room0/Val0/SubVal1
Room0/Val0/SubVal2
Room0/Val0/SubVal3
Room0/Val1
Room1/Val0
Room1/Val1

Naming channels as shown above will create an OPC-UA address space as shown below.

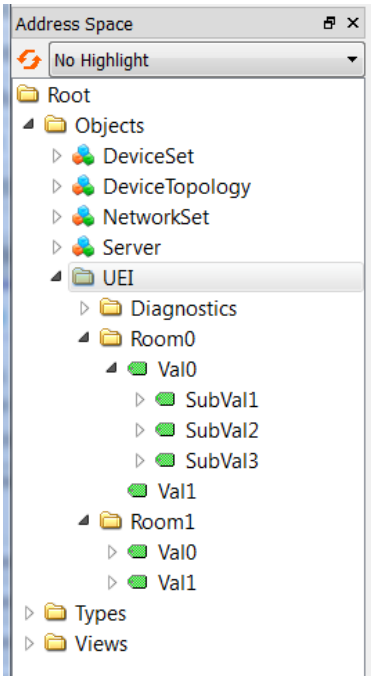


Figure 6 UEIOPC-UA AI Channels Displayed in OPC-UA Address Space

www.ueidaq.com
508.921.4600

3.2.3 Using the Web UI

The UEIOPC-UA can be configured, started and stopped through its web based UI.

The status string shows the state of the OPC-UA server:

- Stopped: OPC-UA service is not running.
- Running: OPC-UA service is running.

The **Start OPC server** and **Stop OPC server** buttons will start or stop the OPC-UA service.

The **Save configuration** button saves the current configuration to the UEIOPC-UA. You need to stop and start the service to make configuration changes take effect.

Note that a dialog box will pop-up after clicking **Save configuration** if the file system is write protected. You can click **Yes** to over-write the current UEIOPC-UA configuration or **No** to leave the configuration unchanged.

Once configuration is done, you can check the **Autostart** checkbox to automatically start the UEIOPC-UA application upon power-up.

3.2.3.1 Channels tab

UEIOPC-UA I/O devices are listed under the Channels tab in the web UI display.

To configure a channel, click the device where the channel you want to configure is located.

Enable **Diagnostics** if you want to access the diagnostic readings of an I/O board. Only I/O boards that are part of UEI's Guardian series support access to diagnostic features.

Enable and set the name of the channels you want to publish.

Enable **Historical Logging** to save a history of the channel values on the UEIOPC-UA on-board SD card. See chapter 5 for more information.

www.ueidaq.com
508.921.4600

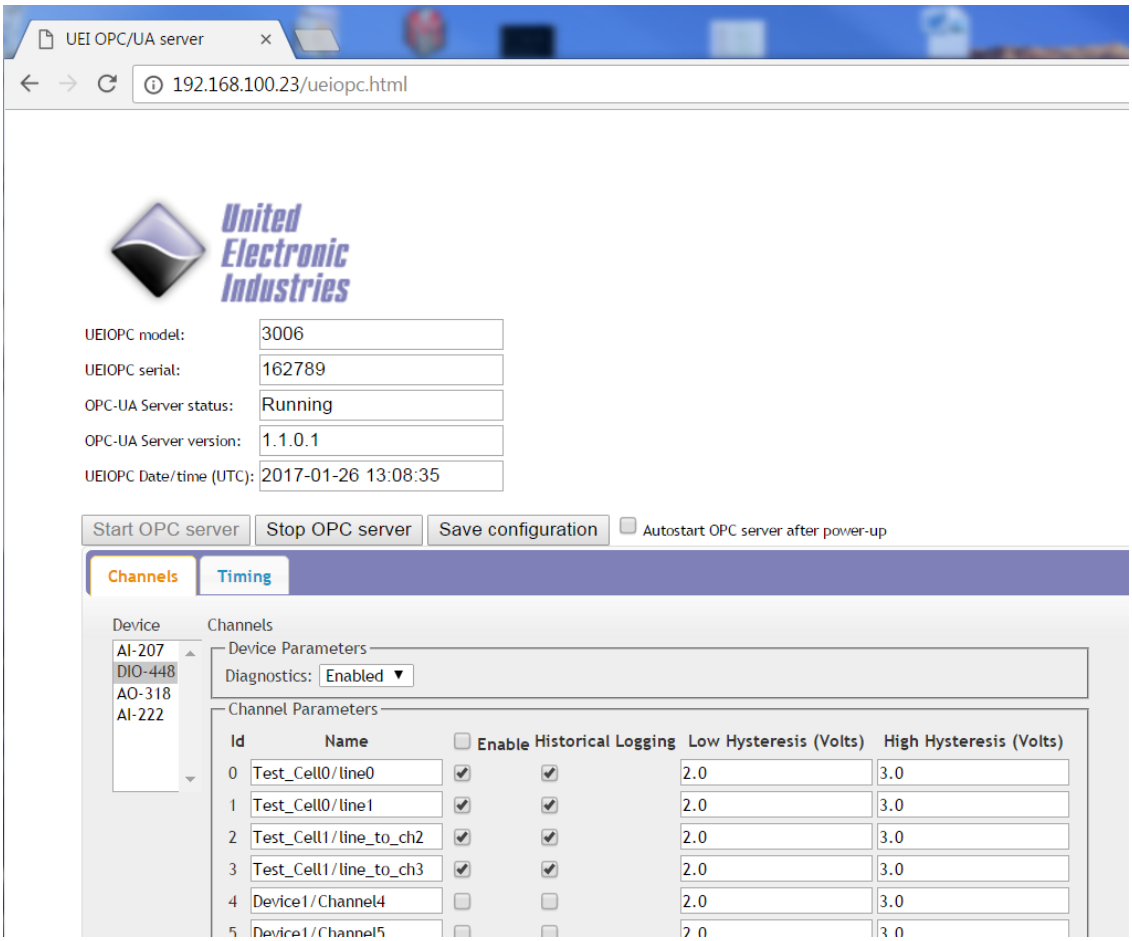


Figure 7 UEIOPC-UA Channels Display

The channel name specifies the OPC-UA path of the value node associated with the channel in the OPC-UA address space.

The '/' character is used to separate the hierarchical levels. For example the configuration shown in Figure 7 will produce the address space as shown below in Figure 8.

www.ueidaq.com
508.921.4600

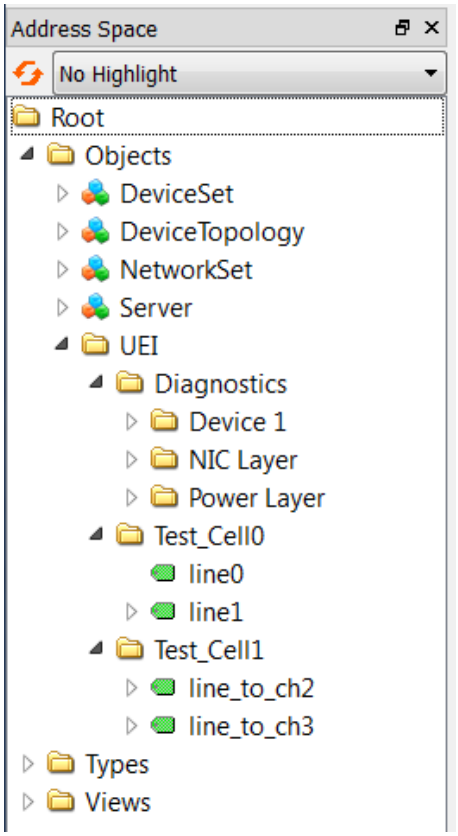


Figure 8 UEIOPC-UA Address Space

Configure other settings (such as gains and scaling) for each enabled channel. The channel settings are detailed in section 3.2.3.

3.2.3.2 Timing tab

The timing tabs contain the following configuration parameter:

- **Scan rate**: Scan rate is the rate at which samples are acquired and stored in the internal buffer. When an alarm is configured, the scan period is the maximum delay between the physical event and the alarm notification to the client.

NOTE: Diagnostic data is read at approximately a 1 Hz scan rate, which is fixed and is different than the scan rate set in the Timing Tab.

www.ueidaq.com
508.921.4600

3.2.4 Manually editing the configuration file

The configuration file is located at "/etc/ueiopc.xml". Configuring the UEIOPC-UA using the web UI will update the XML file automatically; however, users can alternatively edit the XML file directly to configure the UEIOPC-UA manually.

NOTE: Editing ueiopc.xml requires you to make the UEIOPC-UA system read-write. Before editing, type `rw`, and after editing type `ro` to make the system read-only again.

I/O channels are listed within their respective devices which are themselves listed within a parent element named **config**:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<config>
  <device devn="0" type="DI" enableDiag="1">
    <channel id='0' name='Room0/Val0'>
      <name>Room0/Val0</name>
      <enabled>on</enabled>
      <historicalEnabled>on</historicalEnabled>
      <lowHyst>1.8</lowHyst>
      <highHyst>3.4</highHyst>
    </channel>
    <channel id='1' name='Room0/Val0/SubVal1'>
      <name>Room0/Val0/SubVal1</name>
      <enabled>on</enabled>
      <historicalEnabled>on</historicalEnabled>
      <lowHyst>2.0</lowHyst>
      <highHyst>3.0</highHyst>
    </channel>
    <!-- rest of DI board's channels not shown -->
  </device>
  <device devn="1" type="AO" enableDiag="0">
    <channel id='0' name='Test_Cell1/line0'>
      <name>Test_Cell1/line0</name>
      <enabled>on</enabled>
      <initialValue>0.7</initialValue>
    </channel>
    <channel id='1' name='Test_Cell1/line1'>
      <name>Test_Cell1/line1</name>
      <enabled>on</enabled>
      <historicalEnabled>on</historicalEnabled>
      <initialValue>2.2</initialValue>
    </channel>
    <!-- rest of AO board's channels not shown -->
  </device>
</config>
```

Device elements contain the following attributes:

- **Devn**: The device number, which is the location of the device.
  On a Cube form-factor, the top I/O device is #0, the one below is #1, etc.
  On a RACK form-factor, the left-most I/O device is #0, etc.
- **Type**: The type of the device.
  AI: analog input
  AO: analog output
  DI: digital input
  DO: digital output
  CI: counter input and frequency measurement
  QUAD: quadrature encoder position measurement
  PWM: pulse width modulation output
  SERIAL: serial output
- **EnableDiag** ={1|0}: Enables (1) or disables (0) diagnostic functionality and diagnostic register mapping.
  Enabling diagnostics is only applicable for Guardian devices, (i.e., AO-318, AO-333, DIO-432, DIO-433, DIO-462, DIO-463, DIO-448, and DIO-449)

The **channel** elements inside each **device** constitute the channel list. All channels contain the attributes below:

- **Id**: The physical channel Id (as defined on the device's connector pinout)
- **Name**: The OPC-UA node path. Use '/' to separate levels of hierarchy
- **HistoricalEnabled**: Specifies whether this channel should be logged
- **Enabled**: Specifies whether the channel is enabled (useful to temporarily disable channels)

The node path specifies the position of the channel in the OPC-UA address space. The character '/' is used to separate the hierarchical levels.

For example, the XML configuration listed on the previous page will produce the address space below:

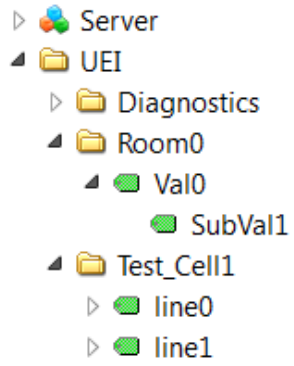


Figure 9 Example of Address Space

www.ueidaq.com
508.921.4600

3.2.5 Configuring an analog input device

Use one of the following channel element names to match your sensor type and analog input device:

- **channel**: Voltage measurement on AI-201, AI-205, AI-207, AI-208, AI-217, AI-218, AI-225, AI-248 and other V_{in} boards.
- **current**: Current measurement on AI-202 and AI-204
- **tcchannel**: Thermocouple temperature measurement on AI-201, AI-207, AI-225, AI-212, AI-248
- **sgchannel**: Strain gauge measurement on AI-208 and AI-224
- **iepechannel**: Accelerometer measurement on AI-211
- **rtdchannel**: RTD temperature measurement on AI-222
- **reschannel**: Resistance measurement on AI-222
- **lvdtchannel**: LVDT sensor position measurement on AI-254
- **synreschannel**: Synchro/Resolver angular position measurement on AI-255 and AI-256

3.2.5.1 Voltage

The voltage **channel** element can contain any of the following sub-elements. If a parameter is omitted, a default value will be used instead.
The parameter is not case sensitive.

- **InputMode**: The input mode possible values are DIFF (0) for differential input or RSE (1) for single ended input (default is ‘diff’).
- **Gain**: The gain 0-based index in the list of gains supported by the device (default is 0).
  For example the AI-207 supports the following gains:
  1,2,4,8,10,20,40,80,100,200,400,800.
  Setting Gain to 6 selects a gain of 40, using 7 to selects a gain of 80 and so on.
- **LinearScaleSlope**: A linear scale uses the equation y=mx+b, where y is the scaled value, m is the slope, x is the pre-scaled value, and b is the y intercept (default is 1.0).
- **LinearScaleYIntercept**: See **LinearScaleSlope** above (default is 0.0).

Note that the linear scale is applied on scaled values (voltage, temperature, etc.).

The following example configures device 1 to acquire voltages on channels 0 and 1 with different gains:

```
<device devn="1" type="ai">
    <channel id='0' name='voltage0'>
      <enabled>on</enabled>
      <inputMode>rse</inputMode>
      <gain>0</gain>
      <linearScaleSlope>3276.8</linearScaleSlope>
      <linearScaleYIntercept>32768</linearScaleYIntercept>
    </channel>
    <channel id='1' name='voltage1'>
      <enabled>on</enabled>
      <inputMode>rse</inputMode>
      <gain>3</gain>
      <linearScaleSlope>3276.8</linearScaleSlope>
      <linearScaleYIntercept>32768</linearScaleYIntercept>
    </channel>
  </device>
```

3.2.5.2 Current

Current measurements share the same **channel** element with voltage measurements.

3.2.5.3 Thermocouple

The thermocouple **tcchannel** element inherits the parameters of the voltage **channel** element.

In addition the sub-elements below configure the parameters specific to thermocouples.

- **ThermocoupleType**: The type of thermocouple connected to the channel. Possible values are E,J,K,S,R,T,B,N,C (default is E).
- **TemperatureScale**: The temperature scale used to convert TC voltage to temperature. Possible values are C,F or K (default is C).
- **CJCType**: The cold-junction compensation type. Possible values are “BUILTIN” to use the CJC sensor on the terminal panel or “CONSTANT” to use a constant value (default is BUILTIN).
- **CJCConst**: The cold-junction compensation temperature to use when CJCType is set to “CONSTANT” (default is 25.0).

The following example configures device 2 to acquire temperatures on channel 4:

```
<device devn="2" type="ai">
    <tcchannel id='4' name='oven_temperature'>
      <enabled>on</enabled>
      <inputMode>diff</inputMode>
      <gain>8</gain>
      <thermocoupleType>J</thermocoupleType>
      <temperatureScale>K</temperatureScale >
      <cjcType>builtin</cjcType>
    </tcchannel>
</device>
```

3.2.5.4 Strain gage

The strain gage **sgchannel** element inherits the parameters of the voltage **channel** element.

In addition the sub-elements below configure parameters specific to the strain gage.

- **ExcVoltage**: The excitation voltage used to power load cells or strain gauges (default is 0.0).
- **ScaleWithExcitation**: 0 to read S-/S+ voltage or 1 to read ratiometric measurements in mV/V (default is 1).
- **GAF**: The gain adjustment factors obtained during shunt calibration procedure (default is 1.0).

This is only available on devices designed to work with load cells or strain gauges (AI-208, AI-224, etc.).

The example below configures device 1 to acquire strains on channel 0:

```
<device devn="1" type="ai">
    <sgchannel id='0' name='stress'>
      <enabled>on</enabled>
      <gain>8</gain>
      <excVoltage>10.0</excVoltage>
      <ScaleWithExcitation>1</ScaleWithExcitation>
    </sgchannel>
</device>
```

3.2.5.5 IEPE/Accelerometer

The IEPE **iepechannel** element inherits the parameters of the voltage **channel** element.

In addition the parameters below configure the parameters specific to the IEPE/Accelerometer.

- **Coupling**: The input coupling. Possible values are DC, AC, AC_1 (1Hz HPF) or AC_0.1 (0.1Hz HPF) (default is AC).
- **Lowpass**: 0 to disable low pass filter or 1 to enable low pass filter (default is 0).
- **Sensitivity**: The sensor sensitivity. The voltage read will be converted to mV and divided by the sensitivity specified in mVolts/[EU] (default is 1000.0).
- **ExcCurrent**: Excitation current used to power the sensor (in mA) (default is 0.0).

This is only available on devices designed to work with IEPE sensors (AI-211).

The example below configures device 1 to acquire acceleration on channel 0:

```
<device devn="1" type="ai">
    <iepechannel id='0' name='vibration0'>
      <enabled>on</enabled>
      <gain>1</gain>
      <coupling>dc</coupling>
      <lowpass>0</lowpass>
      <excCurrent>1.0</excCurrent>
      <sensitivity>1234</sensitivity>
    </iepechannel>
</device>
```

3.2.5.6 RTD

The RTD **rtdchannel** element inherits the parameters of the voltage **channel** element.

In addition the parameters below configure the parameters specific to RTDs.

- **TemperatureScale**: The temperature scale used to convert TC voltage to temperature. Possible values are C,F or K (default is C).
- **Wiring**: 2 for two wires RTD, 3 for three wires RTD and 4 for four wires RTD.
- **RtdType**: Temperature coefficients of resistance of the RTD. Possible values are 3750, 3850, 3902, 3911, 3916, 3920, 3926, 3928.
- **RtdRes**: Nominal resistances of the RTDs at 0° C.
- **LeadsResistance**: Resistance of the RTD leads (used in 2 wires mode only).

This is only available on devices designed to work with RTDs (AI-222).

The example below configures device 7 to acquire RTD temperature on channel 0:

```
<device devn="7" type="ai">
    <rtdchannel id='0' name='nozzle_temperature'>
      <enabled>on</enabled>
      <inputMode>diff</inputMode>
      <gain>8</gain>
      <rtdType>3850</rtdType>
      <temperatureScale>K</temperatureScale >
      <rtdRes>100</rtdRes>
      <wiring>3</wiring>
    </rtdchannel>
</device>
```

3.2.5.7 LVDT/RVDT

The LVDT **lvdtchannel** element inherits the parameters of the voltage **channel** element.

In addition the parameters below configure the parameters specific to LVDTs.

- **Wiring**: 4 for four wires connection, 5 for five wires connection
- **ExcVoltage**: Excitation voltage used to power the LVDT
- **ExcFreq**: Excitation frequency used to power the LVDT
- **Sensitivity**: the voltage read will be converted to mV and and divided by the excitation voltage and the sensitivity specified in mVolts/V/[EU]
- **ExtExc**: 0 to use internal excitation, 1 to use external excitation

This is only available on devices designed to work with LVDT sensors (AI-254).

3.2.5.8 Synchro/Resolver

The Synchro/Resolver **syncreschannel** element inherits the parameters of the voltage **channel** element.

In addition the parameters below configure the parameters specific to Synchros or Resolvers.

- **Mode**: "synchro", "resolver" or "synchrozground"
- **ExcVoltage**: Excitation voltage used to power the synchro
- **ExcFreq**: Excitation frequency used to power the synchro
- **ExtExc**: 0 to use internal excitation, 1 to use external excitation

This is only available on devices designed to work with Synchro or Resolver sensors (AI-255 or AI-256).

3.2.6 Configuring an analog output device

Compatible analog output devices are AO-308, AO-318, AO-332, AO-333.

The **channel** element for an analog output device can contain any of the following sub-elements. If a parameter is omitted, a default value will be used instead.

Parameters are not case sensitive.

- **InitialValue**: The initial value to write to the output channel when the server is started (default is 0.0).
- **LinearScaleSlope**: A linear scale uses the equation y=mx+b, where y is the scaled value, m is the slope, x is the pre-scaled value, and b is the y intercept (default is 1.0).
- **LinearScaleYIntercept**: See **LinearScaleSlope** above (default is 0.0).

The linear scale is applied before sending the values to the D/A converters.

The following example configures device 1 to output voltages on channels 0 and 1:

```
<device devn="1" type="ao">
    <channel id='0' name=''>
      <enabled>on</enabled>
      <initialValue>2.5</initialValue>
      <linearScaleSlope>2.0</linearScaleSlope>
      <linearScaleYIntercept>12</linearScaleYIntercept>
     </channel>
     <channel id='1' name=''>
      <enabled>on</enabled>
      <initialValue>2.5</initialValue>
```

```
      <linearScaleSlope>2.0</linearScaleSlope>
      <linearScaleYIntercept>12</linearScaleYIntercept>
     </channel>
</device>
```

3.2.7 Configuring a digital input device

Compatible digital input devices are DIO-401, DIO-403, DIO-404, DIO-405, DIO-406, DIO-448, DIO-449.

The **channel** element for a digital input device represents an individual input line (1 bit).

It can contain any of the following sub-elements. If a parameter is omitted, a default value will be used instead.

Parameters are not case sensitive.

- **LowHyst**: Program the low hysteresis threshold. The low threshold is used in combination with the high threshold to define a hysteresis window. The input line state will change only when the input signal crosses both thresholds (default is 3.0V).
- **HighHyst**: Program the high hysteresis threshold (default is 9.0V).

Note that some digital input devices do not have a built-in hysteresis comparator.

The following example configures device 5 to acquire digital signals on the first 8 input lines:

```
<device devn="5" type="di">
    <channel id='0' name='input line 0'>
      <enabled>on</enabled>
    </channel>
    <channel id='1' name='input line 1'>
      <enabled>on</enabled>
    </channel>
    <channel id='2' name='input line 2'>
      <enabled>on</enabled>
    </channel>
    <channel id='3' name='input line 3'>
      <enabled>on</enabled>
    </channel>
    <channel id='4' name='input line 4'>
      <enabled>on</enabled>
    </channel>
    <channel id='5' name='input line 5'>
```

```
      <enabled>on</enabled>
    </channel>
    <channel id='6' name='input line 6'>
      <enabled>on</enabled>
    </channel>
    <channel id='7' name='input line 7'>
      <enabled>on</enabled>
    </channel>
</device>
```

3.2.8 Configuring a digital output device

Compatible digital output devices are DO-402, DIO-403, DIO-404, DIO-405, DIO-406, DIO-432, DIO-433, DIO-452, DIO-462, and DIO-470.

The **channel** element for a digital output device represents an individual output line (1 bit). The configuration can contain any of the following sub-elements.

If a parameter is omitted, a default value will be used instead. Parameters are not case sensitive.

- **InitialValue**: The initial value to write to the output channel when the server is started (default is 0).
- **OutputMask**: selects the ports that are configured for output (for bi-directional devices such as the DIO-403) (default is 0, all lines are input).
- **LowCurrentLimit**: specifies the minimum low current for over-range circuit breaker. This parameter is only available for Guardian DIO devices, (i.e., DIO-432, DIO-433, and DIO-462).
- **HighCurrentLimit**: specifies the maximum high current for over-range circuit breaker. This parameter is only available for Guardian DIO devices, (i.e., DIO-432, DIO-433, and DIO-462).

The following example configures line 0 of device 0 to trip its circuit breaker if the current measures over 500 mA or under -500 mA (Note: `enableDiag` is only for DIO-432/433/462. It must be enabled to set diagnostic current limits):

```
<device devn="0" type="DO" enableDiag="1">
    <channel id='0' name='Device0/Channel0'>
      <name>Device0/Channel0</name>
      <enabled>on</enabled>
      <initialValue>0.0</initialValue>
      <lowCurrentLimit>-0.50</lowCurrentLimit>
      <highCurrentLimit>0.50</highCurrentLimit>
    </channel>
</device>
```

As another example, the following configures device 1 to generate digital patterns on output lines 12 and 16 for the 403:

```
<device devn="1" type="do" outputMask=”0x06”>
    <channel id='12' name='output line 0'>
      <enabled>on</enabled>
    </channel>
    <channel id='16' name='output line 1'>
      <enabled>on</enabled>
    </channel>
</device>
```

Special notes for DIO-403: The DIO-403 is the only device where the direction of its six 8-bit digital ports is configurable. Each bit contained in the **outputMask** parameter sets the direction of its respective port. For example “outputMask=0x11” will set ports 0 and 4 as outputs and ports 1, 2, 3 and 5 as inputs.

3.2.9 Configuring a counter input device

Compatible counter input devices are CT-601 and CT-602.

The **channel** element for a counter input device can contain any of the following sub-elements. If a parameter is omitted, a default value will be used instead. Parameters are not case sensitive.

- **Mode**: The mode used to configure the counter. Possible values are **count** to count events, **quad** for basic quadrature encoder position, **period** for period measurement and **pulsewidth** for pulse width (half-period) measurement (default mode is count).
- **Source**: The source signal to count or measure, **internal** uses the on-board 66MHz clock, **external** uses the signal connected to the counter’s input pin (default source is internal clock).
- **Gate**: The gate signal to enable/disable the counter, **internal** sets the gate automatically when counter starts, **external** uses a signal connected to the counter’s gate pin (default is internal gate).
- **InputInverted**: Set to 1 to invert the input signal (default is 0).
- **SourceDebouncer**: The minimum pulse width in micro-seconds detected on the counter input (default is 0).
- **GateDebouncer**: The minimum pulse width in micro-seconds detected on the counter gate (default is 0).

3.2.10 Configuring a quadrature encoder input device

Compatible quadrature input device is QUAD-604.

The **channel** element for a quadrature input device can contain any of the following sub-elements. If a parameter is omitted, a default value will be used instead. Parameters are not case sensitive.

- **Initialpos**: Initial count loaded in counter (default is 0)
- **Decodingtype**: 1x, 2x or 4x. 2x and 4x decoding are more sensitive to smaller changes in position (default is 1x)
- **Zeroindex**: 1 to enable zero indexing. Resets the measurement to the initial value when the Z, A and B inputs are in a given state (default is 0)
- **Zeroindexphase**:
  - zhigh: resets measurement when Z goes high
  - alowblow: resets measurement when Z goes high, A is low and B is low
  - alowbhigh: resets measurement when Z goes high, A is low and B is high
  - ahighblow: resets measurement when Z goes high, A is high and B is low
  - ahighbhigh: resets measurement when Z goes high, A is high and B is high
- **ADebouncer**: the minimum pulse width in micro-seconds on A input. Any pulse whose width is smaller will be ignored (default is 0)
- **BDebouncer**: the minimum pulse width in micro-seconds on B input. Any pulse whose width is smaller will be ignored (default is 0)
- **ZDebouncer**: the minimum pulse width in micro-seconds on Z input. Any pulse whose width is smaller will be ignored (default is 0)

3.2.11 Configuring a frequency/PWM output device

Compatible PWM output devices are CT-601 and CT-602.

The **channel** element for a frequency output device can contain any of the following sub-elements. If a parameter is omitted, a default value will be used instead. Parameters are not case sensitive.

- **Mode**: The mode used to configure the counter to output pulse(s), **pulse** will output a single pulse each time a new value is written to the device. **train** will continuously output pulses (default is pulse).
- **lowticks**: The initial number of clock ticks used to specify the low state duration (default is 10000).
- **highticks**: The initial number of clock ticks used to specify the high state duration (default is 10000).

4 Configuring the security policy

The UEIOPC-UA supports the following security policies by default:

- **None**: This policy does not use security.
- **Basic256**: This policy uses AES-256 for symmetric encryption and SHA-1 for hashing.
- **Basic256sha256**: This policy uses AES-256 and SHA-256 for additional confidentiality.

The following message security modes are supported:

- **Sign**: All messages between client and server are signed, but not encrypted.
- **Sign & Encrypt**: All messages between client and server are encrypted.

4.1 Certificates and trust

Asymmetric encryption algorithms are used during secure communications. This is accomplished by using both private and public keys.

Client and Server that want to establish a secure connection must have the appropriate public and private keys.

The private key must remain secret and is used to sign and/or encrypt messages while the public key is placed into a certificate and distributed to the communication partner.

Certificates are called self-signed certificates, and they are typically generated during installation of the software or at first start.

On UEIOPC-UA, the certificate is generated upon first start at the factory. You can force a re-generation of the keys and certificate by deleting the directory "`/etc/pki`".

To establish a relation of trust between client and server, the self-signed certificate of the client is installed in the trust list of the server, and the server certificate is installed in the trust list of the client.

The exchange of the certificate happens automatically during the first connection attempt.

Your OPC-UA client will most likely prompt you to trust the UEIOPC-UA server certificate. The screenshot below was taken from UaExpert, a free OPC-UA client made by Unified Automation.

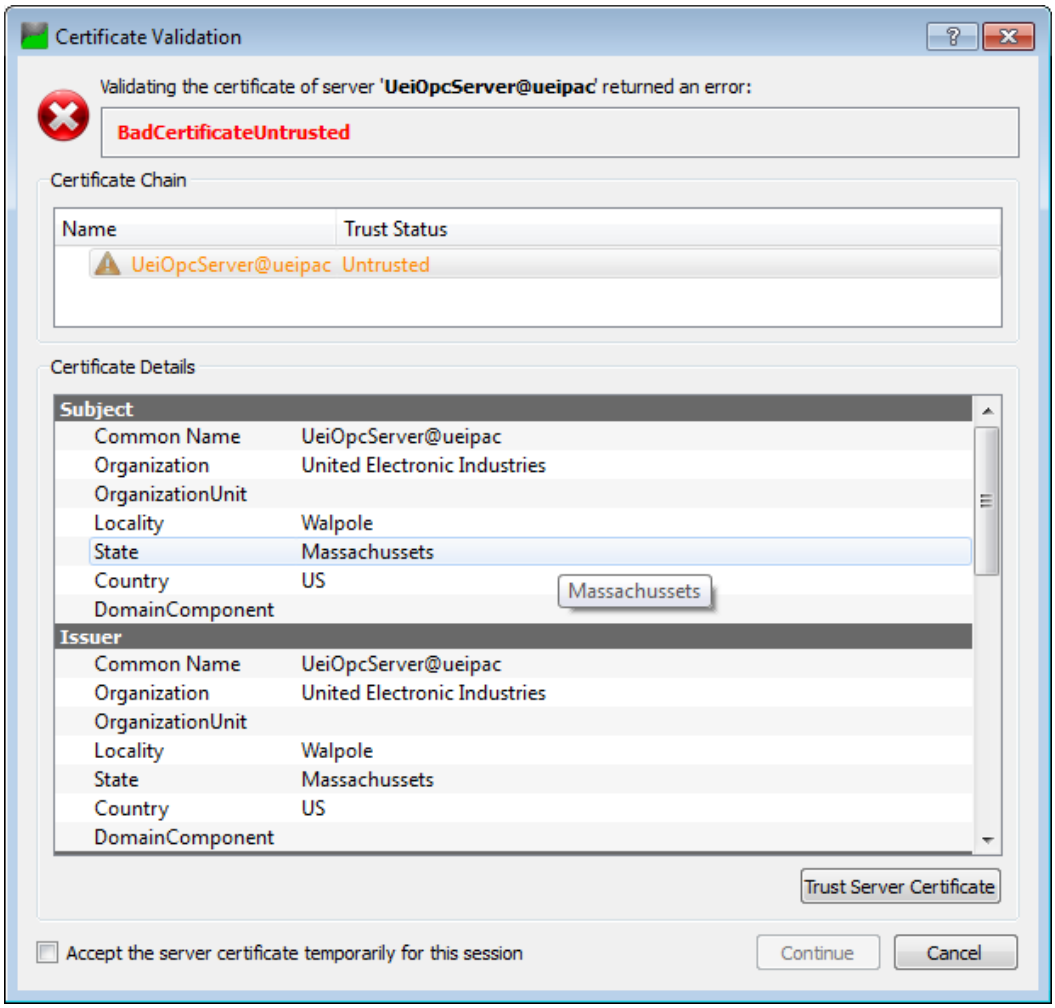


Figure 10 OPC-UA Certificate Validation Window

Important note: The UEIOPC-UA will not prompt you to trust the client certificate. It rejects all certificates by default for security reason.

You must log into the UEIOPC-UA with a remote secure shell or telnet and move the client certificate from `/etc/pki/rejected` to `/etc/pki/trusted/certs`.

Copying or moving files will require you to make the UEIOPC-UA system read-write. Before editing, type `rw`, and after editing type `ro` to make the system read-only again.

To trust a client certificate, run the command below on the UEIOPC-UA command prompt:

```
cp /etc/pki/rejected/<certificate name>.der /etc/pki/trusted/certs
```


www.ueidaq.com
508.921.4600

4.2 Disabling None policy

You can remove **None** from the supported security policies to make sure that only trusted clients connect to UEIOPC-UA.

NOTE: Editing files will require you to make the UEIOPC-UA system read-write. Before editing, type `rw`, and after editing type `ro` to make the system read-only again

Edit "`/etc/opcua/settings.conf`" and look for the line below in the **[Endpoints]** section.

```
Endpoints/0/SecurityPolicies = SecurityPolicy_None,
SecurityPolicy_Basic256, SecurityPolicy_Basic256Sha256
```

Delete **SecurityPolicy_None**, save the file and restart the OPC-UA service.

www.ueidaq.com
508.921.4600

5 Authentication

Authentication is used to control a user's access to nodes in the address space.

5.1 Anonymous

No control is done. All OPCUA clients have access to the entire address space.

5.2 Username/Password

User names and passwords are used to control access. They are configured in a custom group and passwd file.

Edit the file "/etc/opcua/settings.conf" and configure the path to the group and passwd files using the keys **PasswdFilePath** and **GroupFilePath** in the section **[Authentication]**.

5.2.1 Creating a passwd file

Create an `/etc/opcua/passwd` file that includes the users' names and passwords. This file contains four columns:

- `<UserId>`: A unique positive integer used to identify a user
- `<GroupId>`: A unique positive integer used to identify the user's group
- `<UserName>`: The log on name of the user
- `<SHA1 hash of password>`: Hexadecimal representation of the SHA1 hash of the user's password

The `UserId` and `GroupId` are expected to be positive integers, the username can be any string, the SHA1 hash is expected in hexadecimal representation. If the username contains spaces, it must be enclosed in quotation marks.

The user names are private to the OPCUA server. You don't need to create user accounts for them in the operating system.

The following example shows the layout of the file:

```
1       1       root        e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4
2       4       joe         21298df8a3277357ee55b01df9530b535cf08ec1
3       5       john        4f26aeafdb2367620a393c973eddbe8f8b846ebd
4       6       "user one"  e122d28c768ab44ceafe00d71adedc80a535cdf1
```

You can use the command below to generate the SHA1 hash:

```
~# echo -n "password" | sha1sum
```

5.2.2 Creating a custom group file

Create an "/etc/opcua/group**"** file that includes the users' IDs and group IDs. This file contains three columns:

- `<GroupId>`: A unique positive integer used to identify the group
- `<GroupName>`: The name of the group
- `<Users>`: A comma separated list of users that belong to the group, must not contain whitespaces.

Following example shows the layout of the file:

```
0          anonymous    anonymous
1          root         root
2          operators    joe,john
3          users        joe,john,sue
```

5.3 Certificate and private key

The OPCUA client sends a certificate along with a signature to the server. The server can verify the correctness of the signature using the public key embedded in the certificate. If the signature is valid, the server knows that the client has the private key belonging to the certificate.

You can either obtain a certificate from a certificate authority (CA) or issue yourself a self-signed certificate.

Self-signed certificates can be used to encrypt data just as well as CA-signed certificates but a warning will display that says that the certificate is not trusted.

5.3.1 Generating certificates and private keys

You can use OpenSSL to generate the certificate and private key (section 5.3.1.1). Some OPCUA clients are also capable of generating self-signed certificates (see UaExpert example, section 5.3.1.2).

5.3.1.1 Generating certificates with OpenSSL

The command below uses `openssl` to create a self-signed certificate (`mycert.pem`) and a private key (`myprivkey.key`). Both files will be ASCII PEM encoded.
The `-x509` option tells req to create a self-signed certificate.
The `-days` 365 option specifies that the certificate will be valid for 365 days.
The `-nodes` option disables encryption of the new key.

```
openssl req -newkey rsa:2048 -nodes -keyout myprivkey.key -x509 -days
365 -out mycert.pem -sha256
```

Answer the `openssl` information prompts:

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Massachusetts
Locality Name (eg, city) []:Walpole
Organization Name (eg, company) [Internet Widgits Pty Ltd]:United Electronic
Industries
Organizational Unit Name (eg, section) []:R&D
Common Name (e.g. server FQDN or YOUR name) []:JohnDoe
Email Address []:
```

You must convert the certificate to a DER-encoded certificate with the command:

```
openssl x509 -in mycert.pem -outform der -out mycert.der
```

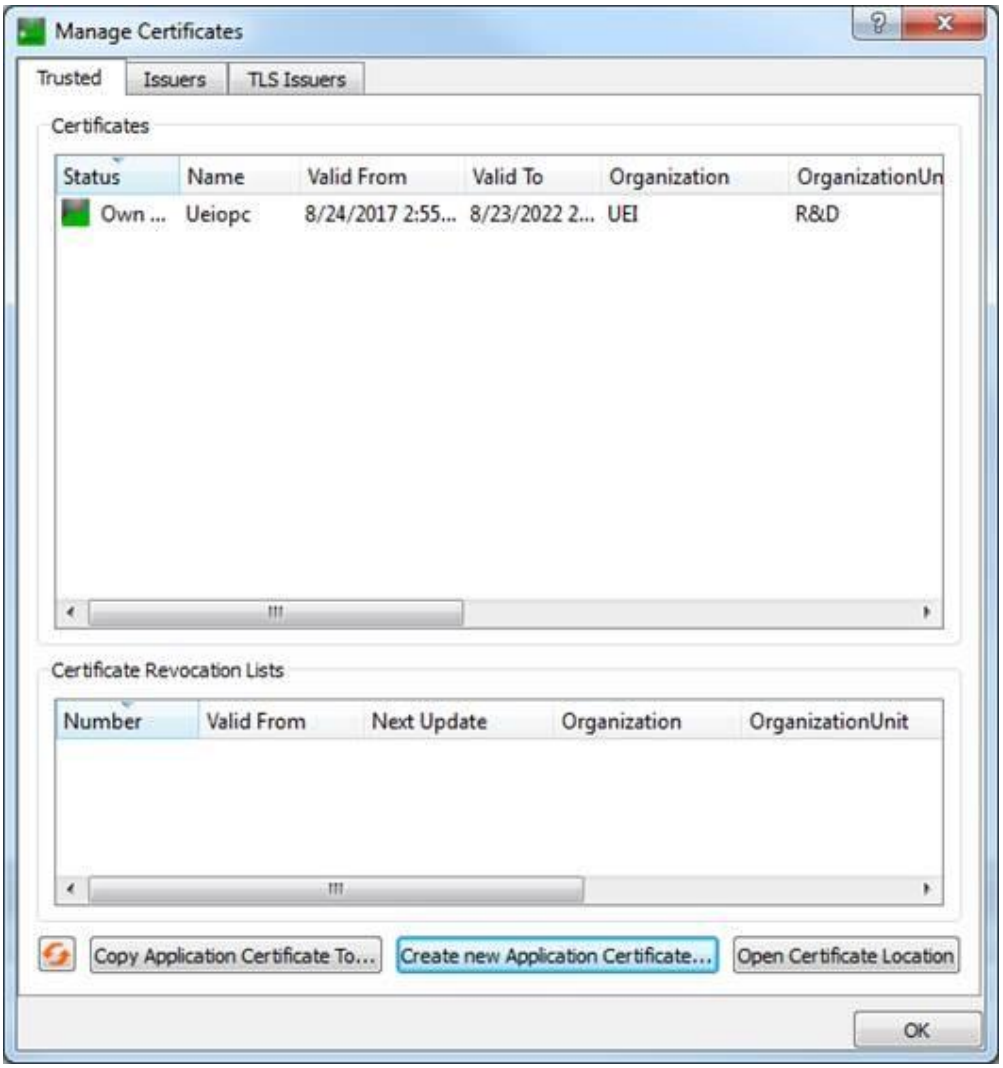
Once you have a DER-encoded certificate, you can store it for use on the UEIOPC-UA. Refer to section 5.3.2 for instructions.

5.3.1.2 Generating certificates with an OPCUA client

The following example uses UaExpert as the OPCUA client generating a certificate and private key.

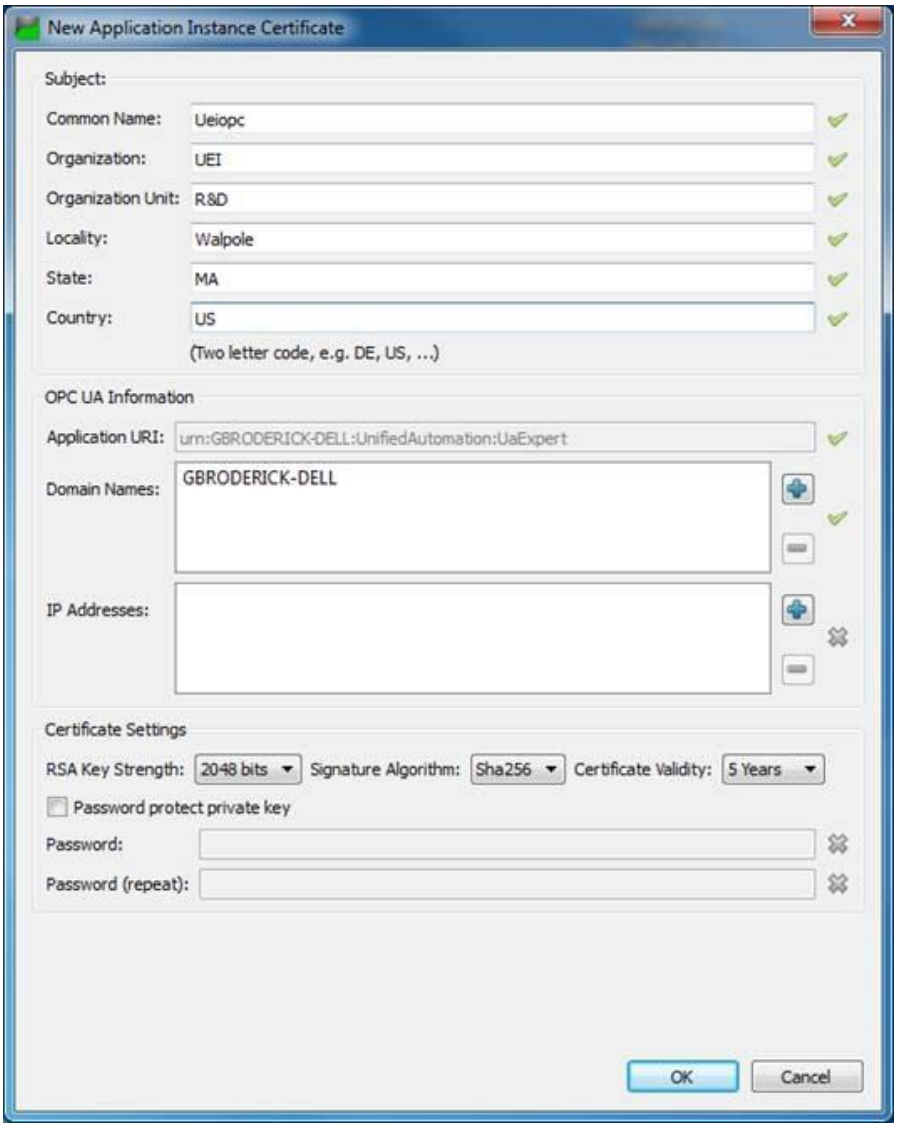
On initial startup, UaExpert will ask you to generate an application instance certificate. This will create both the certificate and private key.

If you have already created a certificate, you can still create a new one by selecting *Settings >> Manage Certificates*. A Manage Certificates dialog will open (see figure on following page).



In the Manage Certificates window, click **Create new Application Certificate.**

Fill in fields in the New Application Instance Certificate dialog.



Click **OK**.

On a Windows machine, the certificate is stored in the following directory:
`C:\Users\<UserName>\AppData\Roaming\unifiedautomation\uaexpert\PKI\own\certs\uaexpert.der`
The private key is at
`C:\Users\<UserName>\AppData\Roaming\unifiedautomation\uaexpert\PKI\own\private\uaexpert_key.pem`

Once created, only the certificate needs to be moved over to the `/etc/auth_pki/certs` directory on the UEIOPC-UA.

Refer to section 5.3.2 for setting up the UEIOPC-UA to store and use authentication certificates.

www.ueidaq.com
508.921.4600

5.3.2 Storing authentication certificate on UEIOPC-UA

You will need to make the UEIOPC-UA read-write before editing the system. Type `rw` at the Linux prompt to make the UEIOPC-UA read-write and after editing type `ro` to make the system read-only again.

Create directories to store authentication certificates on the UEIOPC:

```
mkdir /etc/auth_pki/certs
mkdir /etc/auth_pki/crl
mkdir /etc/auth_pki/cacerts
mkdir /etc/auth_pki/cacrl
```

Copy your certificate file `(<cert>.der)` to "`/etc/auth_pki/certs`" on the UEIOPC.

Edit the file `/etc/opcua/settings.conf` and configure the path to the authentication certificates in the **[Authentication]** section:

```
# The folder containing accepted user certificates for X509 authentication
tokens.
UserCertsDir = /etc/auth_pki/certs
# The folder containing certificate revocation lists for X509 authentication
tokens.
UserCrlDir = /etc/auth_pki/crl
# The folder containing issuer certificates for X509 authentication tokens.
UserIssuerCertsDir = /etc/auth_pki/cacerts
# The folder containing issuer revocation lists for X509 authentication tokens.
UserIssuerCrlDir = /etc/auth_pki/cacrl
```

Finally, add a user in `/etc/opcua/passwd` and `/etc/opcua/group` that matches the CommonName (CN) of the certificate.

```
~ # cat /etc/opcua/passwd
1     1     root  dc76e9f0c0006e8f919e0c515c66dbba3982f785
1000  1000  JohnDoe     40123e9c6273385ea69892c48c80aa6cb25b9113

~ # cat /etc/opcua/group
1     root  root
1000  users JohnDoe
```

6 Historical data logging

The UEIOPC-UA is compliant with the historical data server facets in the OPC-UA specifications.

Historical data logging is disabled by default. You can enable it on a per channel basis. Check the **Historical Logging** checkbox in the web UI to enable it.

You can then access the data history from your OPA-UA client.

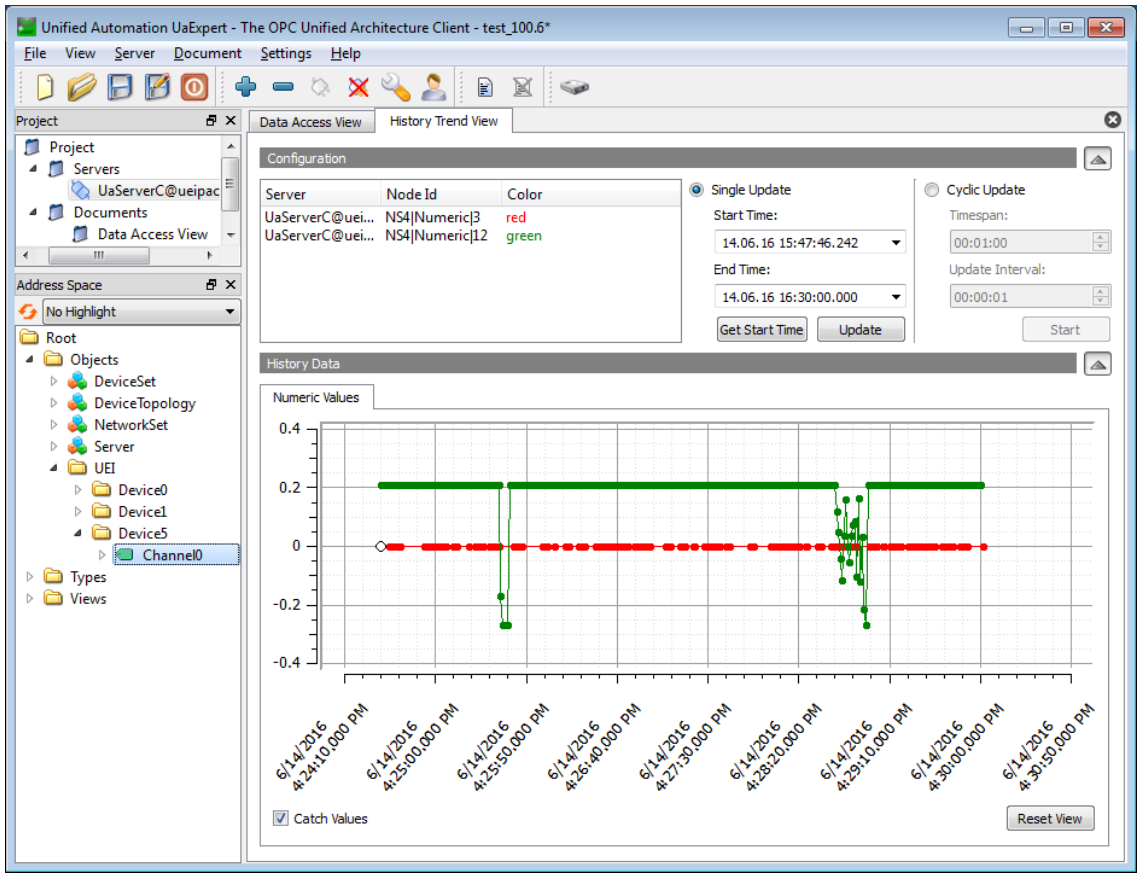


Figure 11 Historical Logging Data History Display

www.ueidaq.com
508.921.4600

7 Booting strategies

The UEIPAC file system contains the libraries, executables and configuration files needed to make the system functional.

By default, the file system is stored on the SD card inserted on the front panel of the UEIOPC-UA.

The file system can alternatively be located in a RAM drive loaded from the FLASH memory or loaded from a remote server using the NFS protocol.

The standard UEIOPC-UA file system is read/write to ease the configuration and allow uploading of files during the development phase.

Once an application/configuration is stable, it is recommended to convert the file system to read-only mode to render the UEIOPC-UA file system resilient against un-scheduled shutdowns.

7.1 Booting an SD card with system partition read-only

The procedure below converts the standard UEIOPC-UA file system to a read-only one.

1. Edit /etc/fstab as below to mount a RAM disk at /var (ram disk maximum size is set to 2MBytes):

```
/dev/sdcard1    /           ext3    defaults,noatime    1   1
none            /proc       proc    defaults            0   0
none            /sys        sysfs   defaults            0   0
none            /dev/pts    devpts  defaults            0   0
tmpfs           /var        tmpfs   defaults,size=2M    0   0
```

2. Create a new script /etc/varsetup.sh with the content below. It sets up the folders needed in /var and maps a few writable folders at /tmp, /mnt and /home.

```
mkdir /var/tmp
mkdir /var/log
mkdir /var/lib
mkdir /var/lib/misc
mkdir /var/spool
mkdir /var/spool/cron
mkdir /var/spool/cron/crontabs
mkdir /var/run
mkdir /var/lock
mkdir /var/mnt
```

```
mkdir /var/home

mount --bind /var/tmp /tmp
mount --bind /var/mnt /mnt
mount --bind /var/home /home
```

3. Edit /etc/inittab as shown below to execute varsetup.sh.

```
# Mount all filesystem listed in /etc/fstab
::sysinit:/bin/mount –a

# Create and mount non-persistent folders
::sysinit:/etc/varsetup.sh

# Configure local network interface
::sysinit:/sbin/ifconfig lo 127.0.0.1 up
::sysinit:/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo

# run rc scripts
::sysinit:/etc/rcS

# Start a shell on the console
ttyS0::respawn:-/bin/sh

# unmount root file system when shutting-down
::shutdown:/bin/umount -a -r
```

4. Create symbolic links to files stored in /etc that need to be kept writeable.

```
ln –s /var/resolv.conf /etc/resolv.conf
ln –s /var/layers.xml /etc/layers.xml
```

5. Connect the console serial port, power-up the UEIOPC-UA, and press any key to enter U-Boot. Type the following commands to load the root file system read-only:

```
setenv bootargs console=ttyS0,57600 root=62:1 ro
saveenv
reset
```

7.2 Restoring or creating a new SD card

Restoring or initializing a new SD card can only be done on a Linux PC (real or virtual).

1. Locate the SD card image file rfs-x.y.z.tgz on your UEIOPC-UA CDROM as well as the script containing the sequence of commands to partition, format and initialize a new SD card.
2. Connect the SD card via a USB adapter (or directly if your computer has a built-in reader).
3. Type the command ***dmesg*** to find out what device node is associated with the SD card. (Linux kernel outputs messages when it detects a new removable drive)
4. Assuming that /dev/sdb is the SD card device node, type ***./createsdcard_2part.sh /dev/sdb rfs-x.y.z.tgz*** to partition, format and copy files to the card.

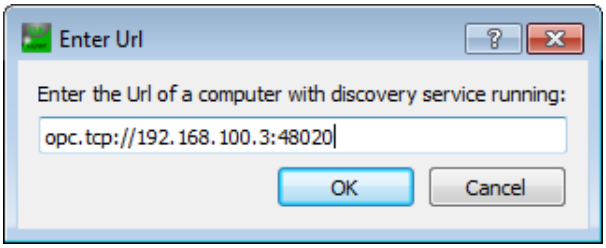
8 Connecting with an OPC-UA client

The instructions below assume that you already started the UEIOPC-UA server using the web interface or the command prompt. This example uses UaExpert, a full-featured OPC UA Client, though any OPC-UA client can be used with UEIOPC-UA.

8.1 UaExpert

Download the UaExpert installer at:
https://www.unified-automation.com/products/development-tools/uaexpert.html

1. Start UaExpert.
2. Click *Server >> Add…* , and then double-click **Double click to Add Server…**
3. Enter URL with the UEIOPC-UA IP address and port configured in section 2.3 (the port is 48020 by default).



4. Select **Anonymous** for Authentication Settings, and then click **OK.**

   NOTE: UA Expert will pop up a certificate validation dialog window (see Figure 10 on page 32) even if you connect using anonymous authentication.

5. Right-click the new entry under Servers in the “Project” panel, and select **Connect**.

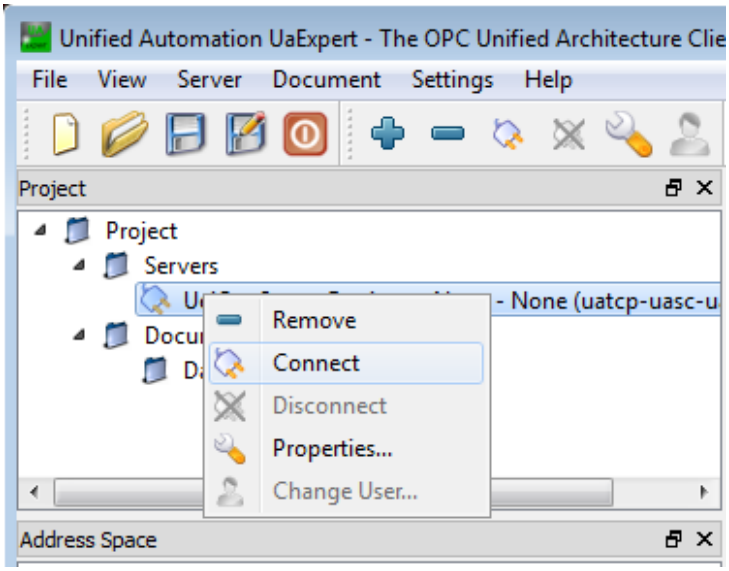


Figure 12 Connect OPC-UA Client to Server

6. Verify that the UEIOPC-UA channels are visible in the “Address Space” under the UEI folder (refer to Figure 13).

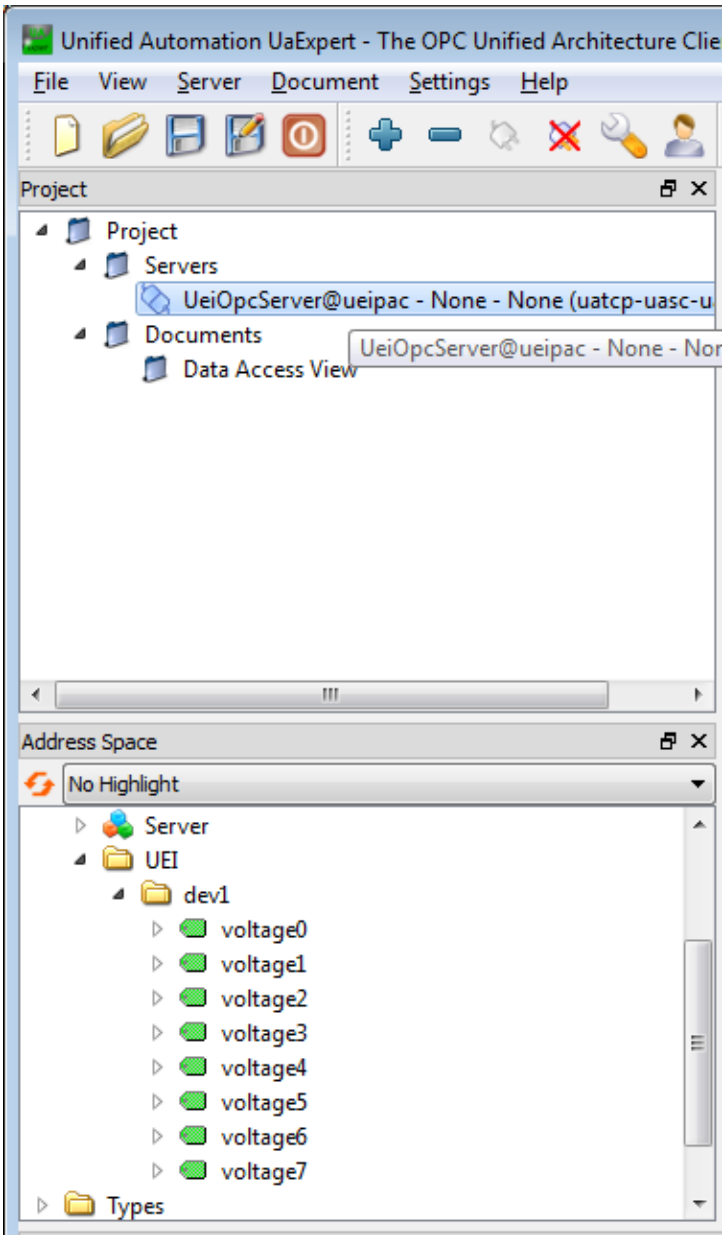


Figure 13 Channels Display in Client Address Space

7. Drag a few channels to the “Data Access View”.

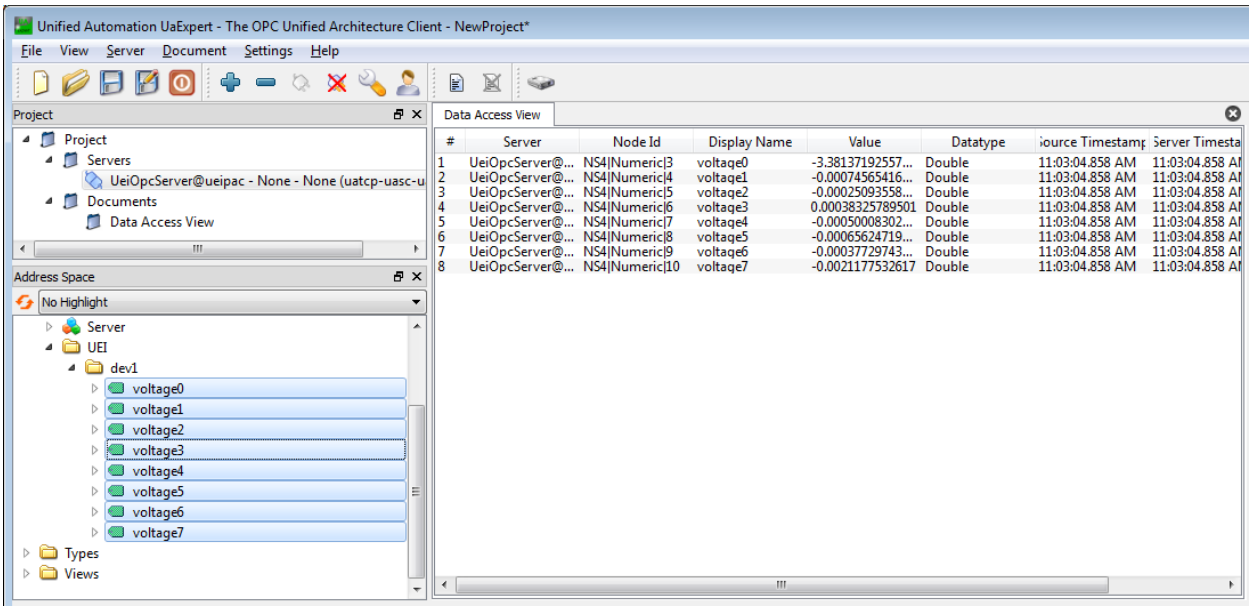


Figure 14 Channels in the Data Access View

8. If you enabled Historical logging on some of your channels, click *Document* >> *Add…* and select “History Trend View”.

9. Drag and drop the channel node to the “Configuration” box, and click **Update** to view the data (refer to Figure 15).

www.ueidaq.com
508.921.4600

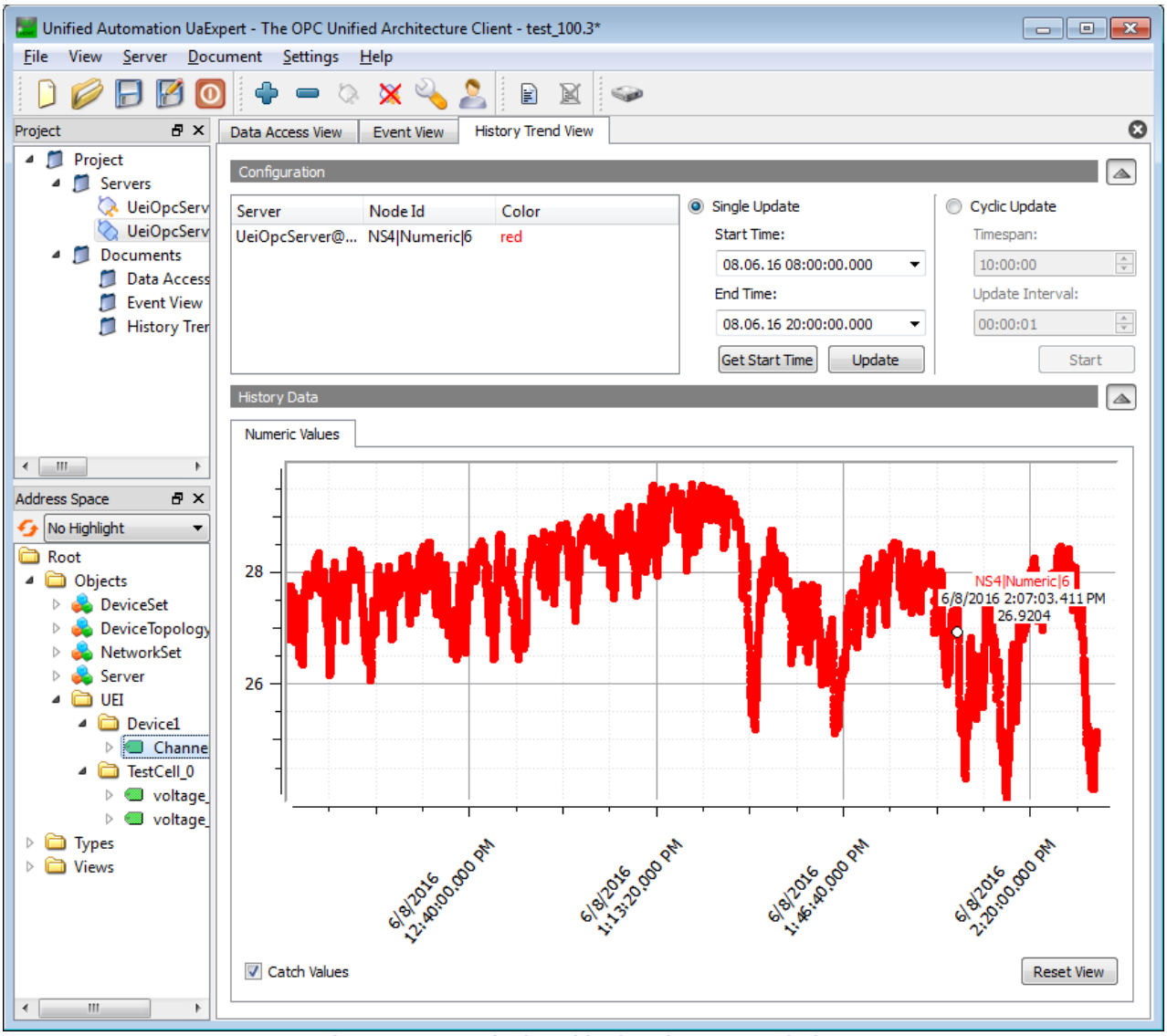


Figure 15 Data Displayed in the History Trend View