

thingworx<sup>®</sup> keeware<sup>®</sup> edge

© 2019 PTC Inc. All Rights Reserved.

# Table of Contents

<b>Table of Contents</b> .....	<b>2</b>
Introduction .....	10
ThingWorx Kepware Edge System Requirements .....	10
API Architecture .....	11
Installing ThingWorx Kepware Edge .....	11
Application Data .....	12
ThingWorx Kepware Edge Licensing .....	12
Command Line Interface — edge_admin .....	14
<b>Getting Started</b> .....	<b>14</b>
Managing ThingWorx Kepware Edge Manually (Daemonless) .....	16
Documentation Endpoint .....	17
Configuration API Service — Endpoints .....	18
Enabling Interfaces .....	19
<b>Interfaces and Connectivity</b> .....	<b>20</b>
OPC UA Interface .....	20
ThingWorx Native Interface .....	21
Configuration API — Project Example .....	22
Configuration API Allen-Bradley ControlLogix Example .....	23
Configuration API Modbus Ethernet Example .....	24
Configuration API Siemens TCP/IP Ethernet Example .....	26
Configuration API ThingWorx Connection .....	28
<b>Configuring an IoT Gateway</b> .....	<b>29</b>
Configuring Self-Signed Certificates for MQTT Agent .....	31
<b>Components and Concepts</b> .....	<b>31</b>
What is a Channel? .....	32
What is a Device? .....	32
What is a Tag? .....	33
Tag Properties — General .....	33
System Tags .....	35
Property Tags .....	44
Statistics Tags .....	45
Dynamic Tags .....	47
Tag Properties — Scaling .....	48
What is a Tag Group? .....	49

Tag Group Properties .....	50
What is the Alias Map? .....	50
Alias Properties .....	50
What is the Event Log? .....	51
Properly Name a Channel, Device, Tag, and Tag Group .....	51
<b>Configuration API Service .....</b>	<b>51</b>
Security .....	51
Documentation .....	51
Configuration API Service — Invoking Services .....	52
Channel Properties — Configuration API .....	56
Configuration API Service — Creating a Channel .....	56
Configuration API Service — Updating a Channel .....	57
Configuration API Service — Removing Channel .....	58
Device Properties — Configuration API .....	58
Configuration API Service — Creating a Device .....	59
Configuration API Service — Updating a Device .....	60
Configuration API Service — Removing a Device .....	61
Configuration API Service — Creating a Tag .....	61
Configuration API Service — Updating a Tag .....	62
Configuration API Service — Removing a Tag .....	63
Configuration API Service — Creating a Tag Group .....	64
Configuration API Service — Updating a Tag Group .....	64
Configuration API Service — Removing a Tag Group .....	66
Configuration API Service — User Management .....	66
Configuration API Service — Creating a User .....	70
Configuration API Service — Creating a User Group .....	71
Configuration API Service — Updating a User .....	71
Configuration API Service — Updating a User Group .....	72
Configuration API Service — Configuring User Group Project Permissions .....	72
OPC UA Endpoint .....	73
Configuration API Service — Creating a UA Endpoint .....	76
Configuration API Service — Updating a UA Endpoint .....	76
Configuration API Service — Removing a UA Endpoint .....	77
Configuration API Service — Data .....	77
Configuration API Service — Content Retrieval .....	80
Configuration API Service — Concurrent Clients .....	86

<b>Using UaExpert</b> .....	<b>86</b>
<b>Event Log Messages</b> .....	<b>89</b>
Configuration API Service — Response Codes .....	89
Configuration API Service — Logging .....	89
The Config API SSL certificate contains a bad signature. ....	90
The Config API is unable to load the SSL certificate. ....	90
Unable to start the Config API Service. Possible problem binding to port. ....	90
The Config API SSL certificate has expired. ....	90
The Config API SSL certificate is self-signed. ....	90
The <name> device driver was not found or could not be loaded. ....	90
Unable to load the '<name>' driver because more than one copy exists ('<name>' and '<name>'). Remove the conflicting driver and restart the application. ....	91
Invalid project file. ....	91
Unable to add channel due to driver-level failure. ....	91
Unable to add device due to driver-level failure. ....	92
Version mismatch. ....	92
Invalid XML document: .....	92
Unable to load project <name>: .....	92
Unable to backup project file to '<path>' [<reason>]. The save operation has been aborted. Verify the destination file is not locked and has read/write access. To continue to save this project without a backup, deselect the backup option under Tools   Options   General and re-save the project. ....	93
<feature name> was not found or could not be loaded. ....	93
Unable to save project file <name>: .....	93
Device discovery has exceeded <count> maximum allowed devices. Limit the discovery range and try again. ....	93
<feature name> is required to load this project. ....	93
Unable to load the project due to a missing object.   Object = '<object>'. ....	94
Invalid Model encountered while trying to load the project.   Device = '<device>'. ....	94
Cannot add device. A duplicate device may already exist in this channel. ....	94
Auto-generated tag '<tag>' already exists and will not be overwritten. ....	94
Unable to generate a tag database for device '<device>'. The device is not responding. ....	94
Unable to generate a tag database for device '<device>': .....	95
Auto generation produced too many overwrites, stopped posting error messages. ....	95
Failed to add tag '<tag>' because the address is too long. The maximum address length is <num- ber>. ....	95
Line '<line>' is already in use. ....	95
Hardware error on line '<line>'. ....	96
No comm handle provided on connect for line '<line>'. ....	96
Unable to use network adapter '<adapter>' on channel '<name>'. Using default network adapter. ....	96

Rejecting attempt to change model type on a referenced device '<channel device>'. .....	96
Validation error on '<tag>': <error>. .....	97
Validation error on '<tag>': Invalid scaling parameters. ....	97
<Source>: Invalid Ethernet encapsulation IP '<address>'. .....	97
The '<product>' driver does not currently support XML persistence. Save using the default file format. ....	97
The time zone set for '<device>' is '<zone>'. This is not a valid time zone for the system. Defaulting the time zone to '<zone>'. ....	98
The specified network adapter is invalid on channel '%1'   Adapter = '%2'. ....	98
No tags were created by the tag generation request. See the event log for more information. ....	98
TAPI configuration has changed, reinitializing... ..	98
<Product> device driver loaded successfully. ....	98
Starting <name> device driver. ....	98
Stopping <name> device driver. ....	98
Attempting to automatically generate tags for device '<device>'. ....	99
Completed automatic tag generation for device '<device>'. ....	99
Data collection is enabled on device '<device>'. ....	99
Data collection is disabled on device '<device>'. ....	99
Object type '<name>' not allowed in project. ....	99
Created backup of project '<name>' to '<path>'. ....	99
Device '<device>' has been auto-promoted to determine if communications can be re-established. ....	99
Failed to load library: <name>. ....	99
Failed to read build manifest resource: <name>. ....	99
The project file was created with a more recent version of this software. ....	99
A client application has disabled auto-demotion on device '<device>'. ....	100
Access to object denied.   User = '<account>', Object = '<object path>', Permission = .....	100
Changing runtime operating mode. ....	100
Runtime operating mode change completed. ....	100
Shutting down to perform an installation. ....	100
User moved from user group.   User = '<name>', Old group = '<name>', New group '<name>'. ..	100
User group has been created.   Group = '<name>'. ....	100
User added to user group.   User = '<name>', Group = '<name>'. ....	100
User information replaced by import.   File imported = '<absolute file path>'. ....	100
User group has been renamed.   Old name = '<name>', New name = '<name>'. ....	101
Permissions definition has changed on user group.   Group = '<name>'. ....	101
User has been renamed.   Old name = '<name>', New name = '<name>'. ....	101
User has been disabled.   User = '<name>'. ....	101
User group has been disabled.   Group = '<name>'. ....	101

User has been enabled. | User = '<name>'. ..... 101

User group has been enabled. | Group = '<name>'. ..... 101

Failed to reset password for administrator. | Administrator name = '<name>'. ..... 101

Password for user has been changed. | User = '<name>'. ..... 101

Attempt to add item '<name>' failed. .... 101

No device driver DLLs were loaded. .... 102

Invalid project file: '<name>'. .... 102

Could not open project file: '<name>'. .... 102

Rejecting request to replace the project because it's the same as the one in use: '<name>'. .... 102

Filename must not overwrite an existing file: '<name>'. .... 102

Filename must not be empty. .... 102

Filename is expected to be of the form <subdir>/<name>.{json,opf} ..... 102

Filename contains one or more invalid characters. .... 102

Addition of object to '<name>' failed: <reason>. .... 102

Move object '<name>' failed: <reason>. .... 102

Update of object '<name>' failed: <reason>. .... 103

Delete object '<name>' failed: <reason>. .... 103

Unable to load startup project '<name>': <reason>. .... 103

Failed to update startup project '<name>': <reason>. .... 103

Runtime project replaced with startup project defined. Runtime project will be restored from '<name>' at next restart. .... 103

Ignoring user-defined startup project because a configuration session is active. .... 103

Write request rejected on read-only item reference '<name>'. .... 103

Unable to write to item '<name>'. .... 103

Write request failed on item '<name>'. The write data type '<type>' cannot be converted to the tag data type '<type>'. .... 103

Write request failed on item '<name>'. Error scaling the write data. .... 103

Write request rejected on item reference '<name>' since the device it belongs to is disabled. .... 104

<Name> successfully configured to run as a system service. .... 104

<Name> successfully removed from the service control manager database. .... 104

Runtime re-initialization started. .... 104

Runtime re-initialization completed. .... 104

Updated startup project '<name>'. .... 104

Runtime service started. .... 104

Runtime process started. .... 104

Runtime performing exit processing. .... 104

Runtime shutdown complete. .... 104

Shutting down to perform an installation. .... 105

Runtime project replaced from '<name>'. .... 105

Missing application data directory. ....	105
Runtime project saved as '<name>'. ....	105
Runtime project replaced. ....	105
Configuration session started by <name> (<name>). ....	105
Configuration session assigned to <name> has ended. ....	105
Configuration session assigned to <name> promoted to write access. ....	105
Configuration session assigned to <name> demoted to read only. ....	105
Permissions change applied on configuration session assigned to <name>. ....	105
Failed to start Script Engine server. Socket error occurred binding to local port.   Error = <error>, Details = '<information>'. ....	106
An unhandled exception was thrown from the script.   Function = '<function>', error = '<error>'. .	106
Script Engine service stopping. ....	106
Script Engine service starting. ....	106
Connection to ThingWorx failed.   Platform <host:port resource>, error: <reason>. ....	106
Error adding item.   Item name: '<item name>'. ....	107
Failed to trigger the autobind complete event on the platform. ....	107
Connection to ThingWorx failed for an unknown reason.   Platform <host:port resource>, error: <error>. ....	107
One or more value change updates lost due to insufficient space in the connection buffer.   Number of lost updates: <count>. ....	107
Item failed to publish; multidimensional arrays are not supported.   Item name: '%s'. ....	108
Connection to ThingWorx was closed.   Platform: <host:port resource>. ....	108
Failed to autobind property.   Name: '<property name>'. ....	108
Failed to restart Thing.   Name: '<thing name>'. ....	109
Write to property failed.   Property name: '<name>', reason: <reason>. ....	109
ThingWorx request to add item failed. The item was already added.   Item name: '<name>'. ....	109
ThingWorx request to remove item failed. The item doesn't exist.   Item name: '<name>'. ....	109
The server is configured to send an update for every scan, but the push type of one or more properties are set to push on value change only.   Count: <count>. ....	110
The push type of one or more properties are set to never push an update to the platform.   Count: <count>. ....	110
ThingWorx request to remove an item failed. The item is bound and the force flag is false.   Item name: '<name>'. ....	110
Write to property failed.   Thing name: '<name>', property name: '<name>', reason: <reason>. ..	110
Error pushing property updates to thing.   Thing name: '<name>'. ....	111
Connected to ThingWorx.   Platform: <host:port resource>, Thing name: '<name>'. ....	111
Reinitializing ThingWorx connection due to a project settings change initiated from the platform. ....	111
Dropping pending autobinds due to interface shutdown or reinitialize.   Count: <count>. ....	111
Serviced one or more autobind requests.   Count: <count>. ....	112
Reinitializing ThingWorx connection due to a project settings change initiated from the Con-	112

figuration API. .... 112

Resumed pushing property updates to thing: the error condition was resolved. | Thing name: '<name>'. .... 112

Configuration transfer from ThingWorx initiated. .... 112

Configuration transfer from ThingWorx aborted. .... 112

A socket error occurred listening for client connections. | Endpoint URL = '<endpoint URL>', Error = <error code>, Details = '<description>'. .... 112

The UA Server failed to register with the UA Discovery Server. | Endpoint URL: '<endpoint url>'. .... 112

The UA Server failed to unregister from the UA Discovery Server. | Endpoint URL: '<endpoint url>'. .... 113

The UA Server successfully registered with the UA Discovery Server. | Endpoint URL: '<endpoint url>'. .... 113

The UA Server successfully unregistered from the UA Discovery Server. | Endpoint URL: '<endpoint url>'. .... 113

Sample Profile Library event log message. Reason: <reason>. .... 113

Driver failed to initialize. .... 113

Unable to create serial I/O thread. .... 113

Connection failed. Unable to bind to adapter. | Adapter = '<name>'. .... 114

Device is not responding. .... 114

Device is not responding. | ID = '<device>'. .... 114

Unable to write to address on device. | Address = '<address>'. .... 115

Items on this page may not be changed while the driver is processing tags. .... 115

Specified address is not valid on device. | Invalid address = '<address>'. .... 115

Address '<address>' is not valid on device '<name>'. .... 116

This property may not be changed while the driver is processing tags. .... 116

Unable to write to address '<address>' on device '<name>'. .... 116

Socket error occurred connecting. | Error = <error>, Details = '<information>'. .... 116

Socket error occurred receiving data. | Error = <error>, Details = '<information>'. .... 116

Socket error occurred sending data. | Error = <error>, Details = '<information>'. .... 117

Socket error occurred checking for readability. | Error = <error>, Details = '<information>'. .... 117

Socket error occurred checking for writability. | Error = <error>, Details = '<information>'. .... 117

%s | .... 117

<Name> Device Driver '<name>' .... 118

Could not load item state data. Reason: <reason>. .... 118

Could not save item state data. Reason: <reason>. .... 118

Failed to load the license interface, possibly due to a missing third-party dependency. Run in Time Limited mode only. .... 118

Time Limited mode has expired. .... 119

Type <numeric type ID> limit of <maximum count> exceeded on feature '<name>'. .... 119

The <name> feature license has been removed. The server will enter Time Limited mode unless 119



the license is restored before the grace period expires. ....

Feature <name> is time limited and will expire at <date/time>. .... 120

Feature <name> is time limited and will expire at <date/time>. .... 120

Time limited usage period on feature <name> has expired. .... 120

Cannot add item. Requested count of <number> would exceed license limit of <maximum  
count>. .... 120

**Index** ..... **121**

## Introduction

---

Version 1.633

ThingWorx Kepware Edge is a connectivity server that enables users to connect diverse automation devices and sensors to a wide variety of digital solutions. It offers the stability, performance, and security that is essential for industrial environments. With support for popular and secure Linux operating systems, it supports distributed architectures that improve reliability and security and reduce cost. Built by the industrial connectivity experts, ThingWorx Kepware Edge eliminates the interoperability challenges associated with implementing digital solutions.


## ThingWorx Kepware Edge System Requirements

---


The product has been tested and verified on modern computer hardware running **Ubuntu X86\_64 version 18.04 LTS**. It currently only runs on X86\_64 platforms. Since we have not verified the server on systems with limited RAM, CPU, and disk space; we cannot guarantee its performance if tested in such an environment.


This user manual expects the user has a working knowledge of:

- Linux operating system and commands
- Command line interfaces
- Command line or API utilities, such as Postman or cURL
- ThingWorx Platform
- OPC UA configuration and connectivity
- Client interfaces and connectivity


 *If additional information is required, consult the vendors and websites related to those tools and technologies in use in your environment.*

## Prerequisites

- Ubuntu 18.04 LTS
  -  **Tip:** To install the Linux Standard Base components on Ubuntu, open a terminal and run the following command:  

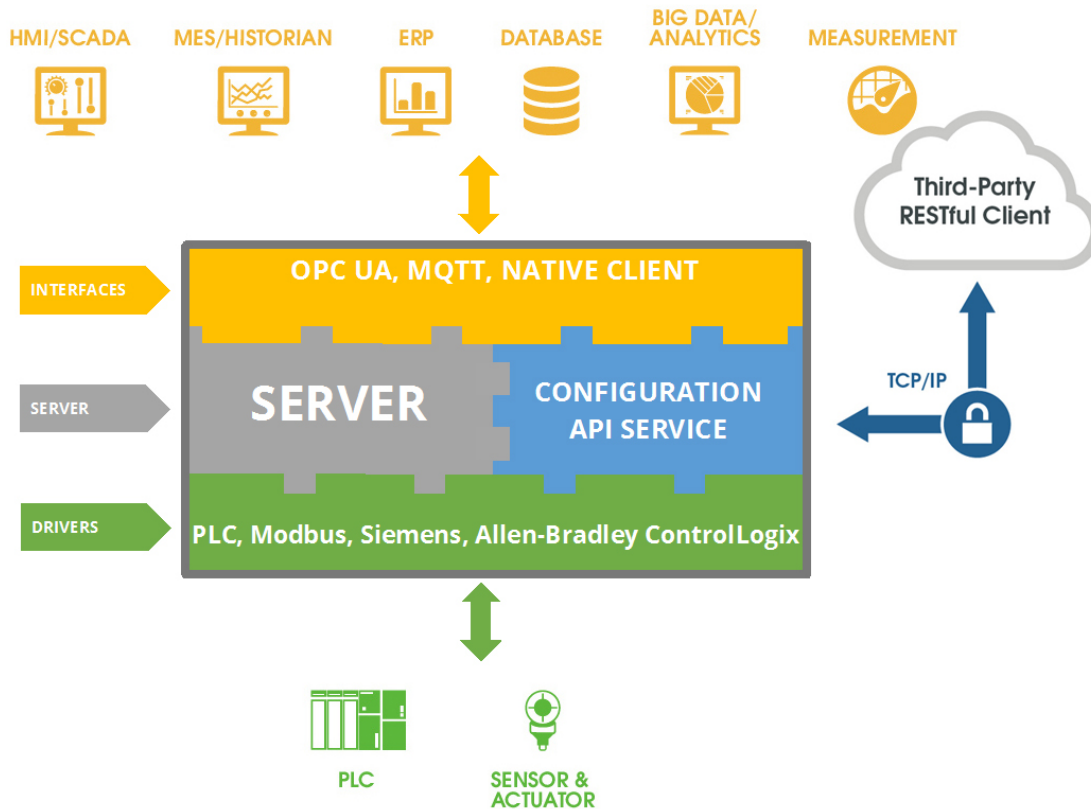
```
$ sudo apt install lsb
```
- Latest Linux Standard Base (LSB) package
  -  **Tip:** To install the Java runtime environment on Ubuntu, open a terminal and run the following command:  

```
$ sudo apt install default-jdk
```
- Java Runtime Environment for MQTT

 **Note:** OpenJDK and Amazon Corretto have been tested and validated for running the MQTT agent.

## API Architecture

The diagram below shows the layout of the components. The Configuration API Service is installed on the same machine with the server.



## Installing ThingWorx Kepware Edge

Before installing ThingWorx Kepware Edge, verify the installer hash to ensure it is the official, secure file. To generate the hash locally, run the following command and compare the results to the hash published [online](#).

```
$ sha256sum thingworx_kepware_edge*
```

The ThingWorx Kepware Edge must be installed by a user with root permissions. The installer supports both GUI and command line installations.

For all installation options, run the following command:

```
$ ./thingworx_kepware_edge*.run --help
```

**Note:** Ubuntu can place a lock on files needed to install software while it is checking for updates. Verify the system is updated before installing ThingWorx Kepware Edge by running the 'apt update' command.

A strong unique password should be set for the ThingWorx Kepware Edge Administrator account during installation. The password must be at least 14 characters and include a mix of uppercase and lowercase letters, numbers, and special characters. Avoid well-known, easily guessed, or common passwords.

● **Once installed, any Linux user accounts administering the ThingWorx Kepware Edge instance must be added to the user group created during the installation, tkedge by default.** This allows those accounts to use the edge\_admin tool and interact with the local file system to move files in and out of the secured data directory (<installation\_directory>/user\_data directory).

## Uninstalling ThingWorx Kepware Edge

To uninstall, run the uninstall command from the <installation\_directory> as root.

For a complete list of uninstall properties run the command:

```
$ sudo ./uninstall --help
```

● **Note:** the uninstall tool leaves the <installation\_directory> with the original install log and an uninstall log. This directory and these files may be removed manually.

● **See Also:** [Command Line Interface — edge\\_admin, Application Data](#)

To access it, run the following command:

```
$ sudo ./thingworx_kepware_edge*.run --help
```

## Application Data

During installation, a user\_data directory is created in the <installation\_directory> path. The user\_data directory is the relative path where all project files are saved to and loaded from, as well as where automatic tag generation (ATG) files should be placed.

● **Note:** All files in the user\_data directory must be world readable or owned by the ThingWorx Kepware Edge user and group that were created during installation, by default this is tkedge.

Any authorized Linux user should be added to the ThingWorx Kepware Edge user group that was created during installation to have the proper permissions to interact locally with this folder. All Configuration API commands that interact with this folder use the ThingWorx Kepware Edge user configured during installation, tkedge by default.

● **Note:** All directories created in the user\_data directory must be writeable by members of the ThingWorx Kepware Edge group created during installation, tkedge by default. Files in the user\_data directory must be either world readable or owned by the group that was setup during installation, tkedge by default.

## ThingWorx Kepware Edge Licensing

Licensing in ThingWorx Kepware Edge is provided on a per-tag basis across the set of supported drivers. This functionality is provided by a USB hardware key in association with a matching certificate file which indicates the licensing level. When no certificate file is present, ThingWorx Kepware Edge runs in a time-limited mode.

The following are the available licensing levels:

- 100 Tag Limit
- 750 Tag Limit
- 1500 Tag Limit
- Unlimited

The tag count is measured across all drivers and is determined at the time of subscription. A tag is not considered utilized for the purpose of licensing unless there is an active subscription for it. Simulator tags and system tags are not included in the tag limit. Subscribe to tags up to the limit established by the valid license. Tags beyond this limit may be added to the server, but will not be active.

## Installing a License

1. Login using a local Linux user account that is a member of the ThingWorx Kepware Edge user group configured during installation, tkedge by default.
2. Copy the .lic license file that corresponds to the hardware key attached to the machine into the <installation\_directory>/config/License directory.
3. Attach the hardware key to the machine and wait for the red light to light up on the key.
4. Restart the ThingWorx Kepware Edge runtime service and the server is now licensed.

## Time-Limited Operation

To operate the server in evaluation mode, ensure that all license files have been removed from the <installation\_directory>/config/License folder. The presence of files in this folder disables time-limited mode and cause immediate deactivation of all unlicensed features. After any change in license status, the server must be restarted for the change to take effect (systemctl restart tkedge\_runtime.service).

## Troubleshooting Licensing

If the .lic license file is located in <installation\_directory>/config/License and the corresponding hardware key is attached to the system and ThingWorx Kepware Edge is still going into expired or demo mode, run the listat command line utility from the install directory to troubleshoot the issue.

Listat doesn't list the license.

Verify that the license has been copied to the <installation\_directory>/config/License directory and is readable by the current user and the ThingWorx Kepware Edge user (default tkedge).

Listat indicates that the license is not valid.

Verify that the usb key is connected to the system. Verify that the license has not been altered.

The USB key is not detected, or not detected as HASP:

Verify that the USB key daemon has been installed (version 7.90-1 expected).

On Ubuntu: apt list -installed

If the USB Key daemon has not been installed, it can be installed manually from <installation\_directory>/dependencies:

On Ubuntu: dpkg -li aksusbd\*.deb

Listat will not start or returns an error.

Verify that the lsb package is installed on your system.

On Ubuntu: lsb\_release -a

If the lsb package is not installed, it can be installed using the following command:

On Ubuntu: apt install lsb

● **Note:** All ThingWorx Kepware Edge services must be restarted after installing LSB.

The server is running but all tags are disabled.

If you are trying to operate in demo mode, make sure that all licenses have been removed from the <installation\_directory>/config/License directory. Verify that the USB key is connected.

## Command Line Interface — edge\_admin

The edge\_admin Command Line Interface (CLI) application is used to manage configuration API settings and certificates for the server and from the command line. The documentation for the edge\_admin CLI may be obtained using the -help option. The following areas of functionality can be accessed through the command line.

### Certificates

- Trust Store: Import / Delete / List / Trust / Reject certificates for various interfaces.
- Instance Certificate: Import / Export / Reissue instance certificates for various interfaces.
- Configuration API Settings: Enable / Disable the Configuration API and manage the ports it listens on.

Linux user accounts that interact with the edge\_admin must be members of the ThingWorx Kepware Edge group that was created during installation. The edge\_admin can be found at the installation location and run from the command line.

### Examples

Obtain general help information and list the areas of the product which can be managed using the CLI:

```
./edge_admin --help
```

View commands related to managing the configuration API:

```
./edge_admin manage-cfgapi --help
```

View commands related to managing the certificates:

```
./edge_admin manage-certificate --help
```

View commands related to managing the trust store:

```
./edge_admin manage-truststore --help
```

To import an OPC UA certificate into the truststore:

```
./edge_admin manage-truststore -i MyCertificateName.der uaserver
```

## Getting Started

ThingWorx Kepware Edge does not have a graphical user interface. Configuration of the server can only be performed using the Configuration API accessed via a REST client, and the edge\_admin command line interface tool. The Configuration API is used to modify all project settings and most administrative settings. The edge\_admin is used to manage certificates and configure the Configuration API administrative settings.

• Additional help for the [edge\\_admin](#) tool may be found by running the tool with the '--help' option:

```
$ <installation_directory>/edge_admin --help
```

Additional help for the Configuration API may be accessed by a web browser at the following [URL](#):

Endpoint:

```
https://<hostname_or_ip>:<port>/config/v1/doc
```

**Tip:** The default port numbers are below.

**Note:** This version includes support for JSON-formatted documentation.

The initial API login credentials use a username of Administrator and the password configured during installation. For best security, a new [group](#) and [user](#) should be created via the Configuration API with only the appropriate permissions enabled.

### Services:

- tkedge\_configapi.service
- tkedge\_eventlog.service
- tkedge\_iotgateway.service
- tkedge\_runtime.service

**Tip:** Once ThingWorx Kepware Edge is installed as a Daemon, verify the processes are running using the following command:

```
$ systemctl | grep tkedge
```

### Ports:

- Configuration API HTTPS interface (Enabled): 57513
- Configuration API HTTP interface (Disabled by default): 57413
- OPC UA interface (Enabled by default): 49330
- Server Runtime service to IoT Gateway service (localhost only): 57213
- Server Runtime service to Configuration API service (localhost only): 32403
- Event Log service (localhost only): 56221

### Service Logs

ThingWorx Kepware Edge services log information to the system journal. To view log information, run:

```
$ journalctl -u <service_name>
```

All service logs may be viewed together by running:

```
$ journalctl -u tkedge*
```

To save the log files to disk, can run the following command:

```
$ journalctl -u tkedge* >> ~/tkedgelog.txt
```

### REST Client Settings

- Endpoint: `https://<hostname_or_ip>:<port>/config/`
- Port: 57513 for HTTPS (57413 for HTTP)
- Authentication: Username and password of the administrator account created during installation

🔥 For security reasons, the administrator account should have a strong unique password. The password must be at least 14 characters and include a mix of uppercase and lowercase letters, numbers, and special characters. Avoid well-known, easily guessed, or common passwords.

## Setting up a Project

During installation, there is an option to load a sample project. If that option was not selected, the default project file is blank. To configure a project, use the API commands in this section to create new channels, devices, and tags. If a baseline project is helpful, the example project may be loaded after installation using these steps:

### Reloading the Sample Project

1. Ensure the services are running.
2. Login using a local Linux user account that is a member of the ThingWorx Kepware Edge user group configured during installation, `tkedge` by default.
3. Copy the example project from `<installation_directory>/examples/tke_simdemo.lpf` to the `<installation_directory>/user_data` directory.
4. Use the configuration API to load the project using the instructions below.

### Project Load Example

Load the project by performing a PUT command from a REST client to invoke request on the ProjectLoad endpoint. The name of the project file is included in the body of the request. Use basic authentication for the request. The response should include the message "Accepted" to indicate the project has been loaded.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectLoad
```

Body:

```
{
  "common.ALLTYPES_NAME": "ProjectLoad",
  "servermain.PROJECT_FILENAME": "tke_simdemo.lpf"
}
```

Authentication:

Basic Authentication with a username of administrator and the password created during installation.

🔥 Do not try to load a JSON project file generated from a Windows environment. ThingWorx Kepware Edge supports only a subset of features; unsupported features in the project file may prevent the project from loading.

## Managing ThingWorx Kepware Edge Manually (Daemonless)

When installing the server, the option of installing the server as a daemon is available. While it is recommended to install as a daemon, it is possible to manage the services manually. If the server is not installed as daemons, the user must start each of the processes correctly. In addition to starting these services manu-



ally, the user is responsible for restarting the services should a critical error occur while the process is running.

## Services

ThingWorx Kepware Edge is comprised of four services:

- **Runtime:** The Runtime is the main server process. This service hosts the current project, communicates with edge devices, and provides access to data over interfaces such as OPC UA, or ThingWorx Native Interface.
- **Event Log:** The Event Log aggregates and manages log entries created by the other services.
- **Configuration API:** The Configuration API service provides a REST web service used to interact with and configure the Runtime. It also provides the ability to retrieve logs from the Event Log service.
- **IoT Gateway:** The IoT Gateway service manages MQTT agents that publish data updates from the Runtime to a MQTT broker.

## Service Startup Order

The services should be started in the correct order or important messages regarding the startup errors could be lost. The correct order the services should be started:

1. Event Log
2. Runtime
3. Configuration API
4. IoT Gateway

## Starting Each Service

The table below lists each service and the required arguments to start that service. These services all require that the working directory of the process be set to directory where the product was installed.

Service	Command to Invoke (Shell)
Service	Command to invoke (shell)
Event Log	<code>./server_eventlog</code>
Runtime	<code>./server_runtime</code>
Configuration API	<code>./config_api_service</code>
IoT Gateway*	<code>java -cp &lt;installdir&gt;/iotg/server-1.0.jar:&lt;installdir&gt;/iotg/lib:&lt;installdir&gt;/config/IoTGateway com.Kepware.Main -port 57213</code>

• **\*Note:** The IoT Gateway must be started with port 57213. Using an alternative port number is not supported at this time.

## Documentation Endpoint

The documentation endpoint can be used to retrieve information about the various endpoints, including:

- Supported properties of the endpoint
- Child nodes of the endpoint
- Property meta data (default values, state, data ranges, etc.)

• **Note:** Documentation served from the landing page is currently only available in JSON encoding.

## Supported Actions

HTTP(S) Verb	Action
GET	Retrieves the current server properties

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/doc
```

🔴 Accessing the documentation endpoint URL via a web browser prompts for authentication. Valid ThingWorx Kepware Edge user credentials must be used to access the documentation.

## Configuration API Service — Endpoints

The Configuration API allows uses the following endpoint mapping scheme:

### Project Connectivity Elements

```
/config/{version}/project
/config/{version}/project/aliases
/config/{version}/project/aliases/{alias_name}
/config/{version}/project/channels
/config/{version}/project/channels/{channel_name}
/config/{version}/project/channels/{channel_name}/devices
/config/{version}/project/channels/{channel_name}/devices/{device_name}
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tags
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tag_groups
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tag_groups/{group_name}
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tag_groups/{group_name}/tags
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tag_groups/{group_name}/tags/{tag_name}
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tag_groups/{group_name}/.../tag_groups
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tag_groups/{group_name}/.../tag_groups/{group_name}/tags
/config/{version}/project/channels/{channel_name}/devices/{device_name}/tag_groups/{group_name}/.../tag_groups/{group_name}/tags/{tag_name}
```

### Plug-in Endpoints

Plug-ins are considered project extensions and are managed under the Project endpoint:

```
/config/{version}/project/{namespace}
/config/{version}/project/{namespace}/{collection}
/config/{version}/project/{namespace}/{collection}/{object_name}
```

### Server Administration Endpoints

```
/config/{version}/admin
/config/{version}/admin/server_usergroups
/config/{version}/admin/server-users
/config/{version}/admin/ua_endpoints
```

### Log Endpoints

```
/config/{version}/log
/config/{version}/event_log
/config/{version}/transaction_log
```

## Documentation Endpoints

```
/config/{version}/doc  
/config/{version}/doc/drivers/{driver_name}/channels  
/config/{version}/doc/drivers/{driver_name}/devices  
/config/{version}/doc/drivers/{driver_name}/tags  
/config/{version}/doc/drivers
```

## Enabling Interfaces

---

For security reasons, only the HTTPS Configuration API endpoint and a secured OPC UA endpoint are enabled by default. The ThingWorx Native Interface and MQTT Agent are disabled by default. Interfaces are enabled or disabled using the Configuration API.

Performing a GET on the project endpoint will return a unique project ID that will be needed to perform a PUT successfully.

### UA Server

The UA Server interface can be enabled from the project endpoint, as shown below:

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project
```

Body:

```
{  
  "project_id": <project_ID_from_GET>,  
  "uaserverinterface.PROJECT_OPC_UA_ENABLE": true  
}
```

### ThingWorx Native Interface

The ThingWorx Native Interface can be enabled from the project endpoint, as shown below:

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project
```

Body:

```
{  
  "project_id": <project_ID_from_GET>,  
  "thingworxinterface.ENABLED": true  
}
```

### MQTT Agent

After creating an MQTT Agent, it can be enabled or disabled in the MQTT Client endpoint, as shown below:

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/_iot_gateway/mqtt_clients/<MQTTAgent_name>
```

Body:

```
{
  "project_id": <project_ID_from_GET>,
  "iot_gateway.AGENTTYPES_ENABLED": true
}
```

## Interfaces and Connectivity

---

This communications server simultaneously supports the client / server technologies listed below.

**Server** - a software application designed to bridge the communication between a device, controller, or data source with a client application. Servers can only respond to requests made by a client.

**Client** - a software program that is used to contact and obtain data from a server (either on the same computer or on another computer). A client makes a request and the server fulfills the request. An example of a client would be an e-mail program connecting to a mail server or an Internet browser client connecting to a web server.

**Human Machine Interface (HMI)** - a software application (typically a Graphical User Interface or GUI) that presents information to the operator about the state of a process and to accept and implement the operator control instructions. It may also interpret the plant information and guide the interaction of the operator with the system.

**Man Machine Interface (MMI)** - a software application (typically a Graphical User Interface or GUI) that presents information to the operator about the state of a process and to accept and implement the operator control instructions. It may also interpret the plant information and guide the interaction of the operator with the system.

For more information on a specific interface, select a link from the list below.

[DDE](#)

[FastDDE/SuiteLink](#)

[iFIX Native Interfaces](#)

[OPC .NET](#)

[OPC AE](#)

[OPC DA](#)

[OPC UA](#)

[ThingWorx Native Interface](#)

## OPC UA Interface

---

### Supported Version

1.02 optimized binary TCP

### Overview

OPC Open Connectivity via Open Standards (OPC) is a set of standard interfaces based on Microsoft's OLE / COM technology. The application of the OPC standard interface makes possible interoperability between automation / control applications and field systems / devices. Unified Architecture (UA User Administration or Unified Architecture) provides a platform independent interoperability standard. It is not a replacement for OPC Data Access (DA Data Access) technologies: for most industrial applications, UA complements or enhances an existing DA architecture. The OPC UA OPC Unified Architecture will replace, modernize, and enhance the functionality of the existing OPC defined interfaces. OPC UA is described in a layered set of

specifications broken into parts. It is purposely described in abstract terms and in later parts married to existing technology on which software can be built. This layering helps isolate changes in OPC UA from changes in the technology used to implement it.

• **See Also:** [Project Properties — OPC UA, OPC UA Endpoints](#)

## OPC UA Profiles

OPC UA is a multi-part specification that defines a number of services and information models referred to as features. Features are grouped into profiles, which are then used to describe the functionality supported by a UA server or client.

• *For a full list and a description of each OPC UA profile, refer to <https://www.opcfoundation.org/profilereporting/index.htm>.*

## Fully Supported OPC UA Profiles

- Standard UA Server Profile
- Core Server Facet
- Data Access Server Facet
- SecurityPolicy - Basic128Rsa15 (Deprecated)
- SecurityPolicy - Basic256 (Deprecated)
- SecurityPolicy - Basic256Sha256
- SecurityPolicy - None (Insecure)
- UA-TCP UA-SC UA Binary

• **CAUTION:** Security policies Basic128Rsa15 and Basic256 have been deprecated by the OPC Foundation as of OPC UA specification version 1.04. The encryption provided by these policies is considered less secure and usage should be limited to providing backward compatibility.

## Partially Supported OPC UA Profiles

- Base Server Behavior Facet

• **Note:** This profile does not support the Security Administrator – XML Schema.

## ThingWorx Native Interface

---

### Overview

ThingWorx is a connectivity platform that allows users to create actionable intelligence based on their device data. The ThingWorx Native Interface allows a user to provide data to the ThingWorx Platform with little additional configuration using the ThingWorx Always On technology. With the introduction of the ThingWorx Next Gen Composer, the ThingWorx Native interface has been updated to allow a better user interface integration with the Composer.

• As noted in the ThingWorx documentation, configuration of a ThingWorx Application Key is crucial to providing a secured environment. The Application Key that is used should provide the appropriate privileges to allow the proper exchange of data between the server instance and the ThingWorx Platform.

• **See Also:**

[Project Properties – ThingWorx Native Interface](#)

[Industrial Internet of Things](#)

[ThingWorx Platform](#)

## Configuration API — Project Example

Project files control the communications and data collection of the server and all the connected devices. Channel and device properties are defined and saved in the project files and how they are configured can impact performance (see *Optimization*). Tag and tag group settings saved in the project can impact how the data is available in control and monitoring displays and reports. There must always be one active open project.

Project saving and loading is restricted to the <installation\_directory>/user\_data directory. A local user must be a member of the ThingWorx Kepware Edge user group created during installation, tkedge by default, to be able to place files in this directory. The <installation\_directory>/user\_data directory is also used for loading of automatic tag generation (ATG) files.

● **Note:** All files in the user\_data directory must be world readable or owned by the ThingWorx Kepware Edge user and group that were created during installation, by default this is tkedge.

● **See Also:** [Application Data](#)

### Save a Project

Use a "PUT" command from a REST client to invoke the ProjectSave service and provide a unique file name for the new file. All files are loaded from and saved to the <installation\_directory>/user\_data directory.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectSave
```

Body:

```
{
  "common.ALLTYPES_NAME": "ProjectSave",
  "servermain.PROJECT_FILENAME": "myProject.json"
}
```

● **Note:** The project is saved to: <installation\_directory>/user\_data/. A path may be included in the file name, such as 'projects/MyProject.json'. Any directory that does not exist within the <installation\_directory>/user\_data/ directory will be created upon successfully saving a project file.

### Update a Project

The typical work flow for editing a project is to read the properties using a GET, modify the properties, then write them into the body of the message using a PUT.

### Read Available Device Properties Example

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/<channel_name>/devices
```

Return:

```
[
  {
    "PROJECT_ID": <project_ID_from_GET>,
    "common.ALLTYPES_NAME": <device_name>,
    "common.ALLTYPES_DESCRIPTION": "",
    "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "<driver>",
    "servermain.DEVICE_MODEL": 0,
    "servermain.DEVICE_UNIQUE_ID": <ID>,
  }
]
```

```
"servermain.DEVICE_CHANNEL_ASSIGNMENT": "<channel_name>",
"servermain.DEVICE_ID_FORMAT": 0,
"servermain.DEVICE_ID_STRING": "<nnn.nnn.n.n>.0",
...
}
]
```

where *nnn.nnn.n.n* is the Device ID address.

### Update Specific Device Properties Example

Only the properties you wish to change are needed for this step.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/<channel_
name>/devices/<device_name>
```

Body:

```
{
"project_id": <project_ID_from_GET>,
"servermain.DEVICE_ID_STRING": "<nnn.nnn.n.n>.0"
}
```

where *nnn.nnn.n.n* is the Device ID address.

## Configuration API Allen-Bradley ControlLogix Example

For a list of channel and device definitions and enumerations, access the following endpoints with the REST client.

### Channel Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/doc/drivers/Allen-Brad-
ley%20ControlLogix%20Ethernet/channels
```

### Device Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/doc/drivers/Allen-Brad-
ley%20ControlLogix%20Ethernet/devices
```

The following API commands are the minimum required to create an AllenBradley ControlLogix Ethernet channel, device, and tag.

• For more information regarding general project configuration, see the server help file.

### Create Allen-Bradley ControlLogix Channel

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyChannel",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Allen-Bradley ControlLogix Ethernet"
}
```

• **See Also:** [Appendix A](#) for a list of channel properties.

## Create Allen-Bradley ControlLogix Device

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/MyChannel/devices
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyDevice",
  "servermain.DEVICE_ID_STRING": "<IP>,0,1",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Allen-Bradley ControlLogix Ethernet",
  "servermain.DEVICE_MODEL": <model enumeration>
}
```

• **Note:** The format of the value for `servermain.DEVICE_ID_STRING` may vary depending on the model enumeration specified for `servermain.DEVICE_MODEL`. The device ID string format shown above is for the ControlLogix 5500 model.

• **See Also:** [Appendix B](#) for a list of device properties.

## Create Allen-Bradley ControlLogix Tags

Endpoint (POST):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/MyChannel/devices/MyDevice/tags
```

Body:

```
[
  {
    "common.ALLTYPES_NAME": "MyTag1",
    "servermain.TAG_ADDRESS": "40001"
  },
  {
    "common.ALLTYPES_NAME": "MyTag2",
    "servermain.TAG_ADDRESS": "40002"
  }
]
```

• **See Also:** [Appendix C](#) for a list of tag properties.

• See server help for more information on configuring tags and tag groups using the Configuration API.

## Configuration API Modbus Ethernet Example

For a list of channel and device definitions and enumerations, access the following endpoints with the REST client.



## Channel Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<-
port>/config/v1/doc/drivers/Modbus%20TCP%20FIP%20Ethernet/channels
```

## Device Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<-
port>/config/v1/doc/drivers/Modbus%20TCP%20FIP%20Ethernet/devices
```

## Create Modbus Channel

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyChannel",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Modbus TCP/IP Ethernet"
}
```

• **See Also:** [Appendix A](#) for a list of channel properties.

## Create Modbus Device

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/MyChannel/devices
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyDevice",
  "servermain.DEVICE_ID_STRING": "<192.160.0.1>.0",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Modbus TCP/IP Ethernet"
}
```

• **See Also:** [Appendix B](#) for a list of device properties.

## Device ID Update

Update the Device ID using a "PUT" command from a REST client.

The Endpoint example below references the "demo-project.json" project configuration with "ModbusTCPIP" channel name and "ModbusDevice" device name.

### Device ID Example

Endpoint (PUT):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/ModbusTCPIP/devices/ModbusDevice
```

Body:

```
{
  "project_id": <project_ID_from_GET>,
  "servermain.DEVICE_ID_STRING": "<IP Address>"
}
```

## Create Modbus Tags

Endpoint (POST):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/MyChannel/devices/MyDevice/tags
```

Body:

```
[
  {
    "common.ALLTYPES_NAME": "MyTag1",
    "servermain.TAG_ADDRESS": "40001"
  },
  {
    "common.ALLTYPES_NAME": "MyTag2",
    "servermain.TAG_ADDRESS": "40002"
  }
]
```

• See Also: [Appendix C](#) for a list of tag properties.

• See server help for more information on configuring projects over the Configuration API.

## Configuration API Siemens TCP/IP Ethernet Example

For a list of channel and device definitions and enumerations, access the following endpoints with the REST client:

### Channel Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<-
port>/config/v1/doc/drivers/Siemens%20TCP%2FIP%20Ethernet/channels
```

### Device Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<-
port>/config/v1/doc/drivers/Siemens%20TCP%2FIP%20Ethernet/devices
```

## Create Siemens Channel

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyChannel",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Siemens TCP/IP Ethernet"
}
```

• **See Also:** [Appendix A](#) for a list of channel properties.

## Create Siemens Device

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/MyChannel/devices
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyDevice",
  "servermain.DEVICE_ID_STRING": "<192.160.0.1>.0",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Siemens TCP/IP Ethernet",
  "servermain.DEVICE_MODEL": <model enumeration>
}
```

• **See Also:** [Appendix B](#) for a list of device properties.

## Create Siemens Tags Example

Endpoint (POST):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/MyChannel/devices/MyDevice/tags
```

Body:

```
[
  {
    "common.ALLTYPES_NAME": "MyTag1",
    "servermain.TAG_ADDRESS": "DB1,INT00"
  },
  {
    "common.ALLTYPES_NAME": "MyTag2",
    "servermain.TAG_ADDRESS": "DB1,INT01"
  }
]
```

• **See Also:** [Appendix C](#) for a list of tag properties.

• **See server help** for more information on configuring projects over the Configuration API.

## Configuration API ThingWorx Connection

To configure the ThingWorx connection, collect the following information from the ThingWorx platform instance to connect:

- **HOSTNAME:** Hostname or IP of machine running ThingWorx
- **PORT:** Port configured to run ThingWorx, typically port 80 for HTTP and 443 for HTTPS
- **APPKEY:** Application key configured in ThingWorx
- **THING\_NAME:** Industrial server name

For a list of ThingWorx interface definitions and enumerations, access the following endpoints with the REST client:

### Project definitions:

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project
```

**Tip:** Enabling ThingWorx and configuring the connection settings can be done at the same time.

### Enable ThingWorx Interface

**Tip:** This is already enabled if the instructions in the Quick Start Guide have been followed.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/
```

Body:

```
{
  "project_id": <project_ID_from_GET>,
  "thingworxinterface.ENABLED": true
}
```

### Configure ThingWorx Test Connection Example

**Note:** This is a testing configuration and the use of certificates and other security measures are suggested for production systems.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project
```

Body:

```
{
  "project_id": <project_ID_from_GET>,
  "thingworxinterface.ENABLED": true,
  "thingworxinterface.HOSTNAME": "<hostname or IP>",
  "thingworxinterface.PORT": <Port Number>,
  "thingworxinterface.RESOURCE": "/Thingworx/WS",
  "thingworxinterface.APPKEY": "<App Key>",
}
```

```

"thingworxinterface.ALLOW_SELF_SIGNED_CERTIFICATE": false,
"thingworxinterface.TRUST_ALL_CERTIFICATES": true,
"thingworxinterface.DISABLE_ENCRYPTION": true,
"thingworxinterface.THING_NAME": "<ThingName>"
}

```

## Configuring an IoT Gateway

The IoT Gateway allows information to be conveyed to an MQTT agent. The section below describes how to configure the IoT Gateway.

### IoT Gateway MQTT Agent Prerequisites

**Caution:** For the most secure configuration, enable ONLY those features that are being used or tested. As such, if MQTT is not being used, this section should be skipped.

1. Install a Java JRE on the machine (if this has not already been installed):

```
apt install default-jdk
```

**Tip:** OpenJDK and Amazon Corretto have been tested.

2. Once installed, verify the Java JRE version using the terminal command:

```
java -version
```

3. Stop and restart all the ThingWorx Kepware Edge services.

4. Install Version 8 Java JRE on the machine.

**Tip:** OpenJDK 8 has been tested. Using Ubuntu, OpenJDK 8 is available via the terminal using the command:

```
apt-get install openjdk-8-jdk
```

or the CentOS command:

```
yum install java-1.8.0-openjdk
```

5. Once installed, verify the Java JRE version using the terminal command: `java -version`

6. Stop and restart the server.

7. Confirm the MQTT agent starts automatically with the startup.sh script.

### Disable IoT Gateway Automatic Start

The IoT Gateway service starts with the startup script even if no MQTT agent is configured. The following instructions disable that behavior if desired.

1. Navigate to the ThingWorx Kepware Edge installation folder.
2. Edit the startup.sh file with a text editor.
3. Add # to the beginning the line that starts with "(java -jar". Corrected version:  

```
#(java -jar ./iotg/server-1.0.jar -port 57312 >> java.trace 2>&1 &)
```
4. Stop and restart the server.

## MQTT Examples

### Create MQTT Agent

Endpoint: (POST)

```
https://<hostname_or_ip>:<port>/config/v1/project/_iot_gateway/mqtt_clients
```

Body:

```
{
  "common.ALLTYPES_NAME": "NewMqttClient",
  "common.ALLTYPES_DESCRIPTION": "",
  "iot_gateway.AGENTTYPES_TYPE": "MQTT Client",
  "iot_gateway.AGENTTYPES_ENABLED": true
}
```

### View MQTT Agents

Endpoint: (GET)

```
https://<hostname_or_ip>:<port>/config/v1/project/_iot_gateway/mqtt_clients
```

### Create MQTT Agent Tag

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/_iot_gateway/mqtt_clients/NewMqttClient/iot_items
```

Body:

```
{
  "common.ALLTYPES_NAME": "Simulator_Word1",
  "iot_gateway.IOT_ITEM_SERVER_TAG": "Simulator.SimulatorDevice.R Registers.Word1",
  "iot_gateway.IOT_ITEM_ENABLED": true
}
```

### View MQTT Agent Tags

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/_iot_gateway/mqtt_clients/NewMqttClient/iot_items
```

### Update MQTT Agent

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/_iot_gateway/mqtt_clients/NewMqttClient
```

Body:

```
{
  "project_id": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "NewMqttClient_updated",
  "common.ALLTYPES_DESCRIPTION": "Update test"
}
```

## Delete MQTT Agent

Endpoint (DEL):

```
https://<hostname_or_ip>:<port>/config/v1/project/_iot_gateway/mqtt_clients/NewMqttClient_updated
```

Body:

```
{
  "project_id": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "NewMqttClient_updated",
  "common.ALLTYPES_DESCRIPTION": "Update test"
}
```

## Configuring Self-Signed Certificates for MQTT Agent

The IoT Gateway supports self-signed certificates with the MQTT agent. These agents use the Java KeyStore to manage these certificates. Use the commands below to import, list, or delete a certificate from the KeyStore.

• These instructions assume the Java keytool is installed.

• The default Java cacerts truststore password is “changeit”

Import certificate into the java store

```
sudo keytool -import -trustcacerts -keystore /usr/lib/jvm/<java_version>/lib/security/cacerts -alias <alias> -file <certificate>
```

List the contents of the certificate

```
keytool -list -keystore /usr/lib/jvm/<java_version>/lib/security/cacerts -alias <alias>
```

Delete the certificate

```
sudo keytool -delete -keystore /usr/lib/jvm/<java_version>/lib/security/cacerts -alias <alias>
```

• The location of the Java Key Store used in the above commands may vary. Use the location appropriate for the local Java installation.

• For more information about working with certificates using the Java keytool, consult the documentation found on the [Oracle Java website](#).

## Components and Concepts

For more information on a specific server component, select a link from the list below.

[What is a Channel?](#)

[What is a Device?](#)

[What is a Tag?](#)

[What is a Tag Group?](#)

[What is the Alias Map?](#)

## [What is the Event Log?](#)

## What is a Channel?

---

A channel represents a communication medium from the PC to one or more external devices. A channel can be used to represent a serial port, a card installed in the PC, or an Ethernet socket.

Before adding devices to a project, users must define the channel to be used when communicating with devices. A channel and a device driver are closely tied. After creating a channel, only devices that the selected driver supports can be added to this channel.

## Creating a Channel

Channels are defined by a set of properties based on the communication methods. Channels are created through the [configuration API service](#).

Channel names must be unique among all channels and devices defined in the project. *For information on reserved characters, refer to [How To... Properly Name a Channel, Device, Tag, and Tag Group](#).*

## Removing a Channel

To remove a channel from the project use the [Configuration API Service](#).

## Displaying Channel Properties

To review the channel properties of a specific channel via the Configuration API, access the [documentation channel endpoint](#).

• See Also: [Channel Properties — General](#)

## What is a Device?

---

Devices represent the PLCs, controllers, or other hardware with which the server communicates. The device driver that the channel is using restricts device selection.

## Adding a Device

Devices are defined by a set of properties based on the protocol, make, and model. Devices are created through the [Configuration API Service](#).

Device names are user-defined and should be logical for the device. This is the browser branch name used in OPC links to access the device's assigned tags.

• *For information on reserved characters, refer to [How To... Properly Name a Channel, Device, Tag, and Tag Group](#).*

The Network ID is a number or string that uniquely identifies the device on the device's network. Networked, multi-dropped devices must have a unique identifier so that the server's data requests are routed correctly. If devices that are not multi-dropped, they do not need an ID, so this setting is not available.

## Removing a Device

To remove a device from the project use the [Configuration API Service](#).

## Displaying Device Properties

To review the channel properties of a specific channel via the Configuration API, access the [documentation channel endpoint](#).

• *For more information, refer to [Device Properties](#).*



---

## What is a Tag?

A tag represents addresses within the device with which the server communicates. The server allows both Dynamic tags and user-defined Static tags. Dynamic tags are entered directly in the OPC client and specify device data. User-defined Static tags are created in the server and support tag scaling. They can be browsed from OPC clients that support tag browsing.

### Adding a Tag

Tags are defined by a set of properties based on the data. Tags are defined through the [Configuration API Service](#).

Tag names are user-defined and should be logical for reporting and data analysis.

• For information on reserved characters, refer to [How To... Properly Name a Channel, Device, Tag, and Tag Group](#).

### Removing a Tag

To remove a tag from the project; use the [Configuration API Service](#).

### Displaying Tag Properties

To review the tag properties of a specific channel via the Configuration API, access the [documentation channel endpoint](#).

---

## Tag Properties — General

A tag represents addresses in the PLC or other hardware device with which the server communicates. The server allows both Dynamic tags and user-defined Static tags. Dynamic tags are entered directly in the OPC client and specify device data. User-defined Static tags are created in the server and support tag scaling. They can be browsed from OPC clients that support tag browsing.

• For more information, refer to [Dynamic Tags and Static User-Defined Tags](#).

**Name:** Enter a string to represent this tag. The tag name can be up to 256 characters in length. While using descriptive names is generally a good idea, some OPC client applications may have a limited display window when browsing the tag space of an OPC server. The tag name is part of the OPC browse data tag names must be unique within a given device branch or tag group branch. For information on reserved characters, refer to [How To... Properly Name a Channel, Device, Tag, and Tag Group](#).

• **Tip:** If the application is best suited for using blocks of tags with the same names, use tag groups to segregate the tags. For more information, refer to [Tag Group Properties](#).

**Description:** Apply a comment to the tag. A string of up to 255 characters can be entered for the description. When using an OPC client that supports Data Access 2.0 tag properties, the description property is accessible from the tag's item Description properties.

**Address:** Enter the target tag's driver address. The address's format is based on the driver protocol. The address can be up to 128 characters.

• **Tip:** For hints about how an address should be entered, click the browse (...) button. If the driver accepts the address as entered, no messages are displayed. A popup informs of any errors. Some errors are related to the data type selection and not the address string.

**Data Type:** Specify the format of this tag's data as it is found in the physical device. In most cases, this is also the format of the data as it returned to the client. The data type setting is an important part of how a

communication driver reads and writes data to a device. For many drivers, the data type of a particular piece of data is rigidly fixed and the driver knows what format needs to be used when reading the device's data. In some cases, however, the interpretation of device data is largely in the user's hands. An example would be a device that uses 16-bit data registers. Normally this would indicate that the data is either a Short or Word. Many register-based devices also support values that span two registers. In these cases the double register values could be a Long, DWord or Float. When the driver being used supports this level of flexibility, users must tell it how to read data for this tag. By selecting the appropriate data type, the driver is being told to read one, two, four, eight, or sixteen registers or possibly a Boolean value. The driver governs the data format being chosen.

- **Default** - Uses the driver default data type
- **Boolean** - Binary value of true or false
- **Char** - Signed 8-bit integer data
- **Byte** - Unsigned 8-bit integer data
- **Short** - Signed 16-bit integer data
- **Word** - Unsigned 16-bit integer data
- **Long** - Signed 32-bit integer data
- **DWord** - Unsigned 32-bit integer data
- **LLong** - Signed 64-bit integer data
- **QWord** - Unsigned 64-bit integer data
- **Float** - 32-bit real value IEEE-754 standard definition
- **Double** - 64-bit real value IEEE-754 standard definition
- **String** - Null-terminated Unicode string
- **BCD** - Two byte-packed BCD value range is 0-9999
- **LBCD** - Four byte-packed BCD value range is 0-99999999
- **Date** - See [Microsoft® Knowledge Base](#).

**Client Access:** Specify whether the tag is **Read Only** or **Read / Write**. By selecting Read Only, users can prevent client applications from changing the data contained in this tag. By selecting **Read / Write**, users allow client applications to change this tag's value as needed. The **Client Access** selection also affects how the tag appears in the browse space of an OPC client. Many OPC client applications allow filtering tags based on attributes. Changing the access method of this tag may change how and when the tag appears in the browse space of the OPC client.

**Scan Rate:** Specify the update interval for this tag when used with a non-OPC client. OPC clients can control the rate at which data is scanned by using the update rate that is part of all OPC groups. Normally non-OPC clients don't have that luxury. The server is used to specify an update rate on a tag per tag basis for non-OPC clients. Using the scan rate, users can tailor the bandwidth requirements of the server to suit the needs of the application. If, for example, data that changes very slowly needs to be read, there is no reason to read the value very often. Using the scan rate this tag can be forced to read at a slower rate reducing the demand on the communications channel. The valid range is 10 to 99999990 milliseconds (ms), with a 10 ms increment. The default is 100 milliseconds.

• With the server's online full-time operation, these properties can be changed at any time. Changes made to tag properties take effect immediately; however, OPC clients that have already connected to this tag are not affected until they release and attempt to reacquire it. Utilize the User Manager to restrict access rights to server features and prevent operators from changing the properties.

## System Tags

System tags provide general error feedback to client applications, allow operational control when a device is actively collecting data, and allow a channel or device's standard properties to be changed by an OPC client application when needed.

The number of system tags available at both the channel level and device level depends on the nature of the driver being used. In addition, application-level system tags allow client applications to monitor the server's status. System tags can also be grouped according to their purpose as both status and control or property manipulation. Descriptions are as follows:

- **Status Tags** Status tags are read-only tags that provide data on server operation.
- **Parameter Control Tags:** Parameter control tags can be used to modify the server application's operational characteristics. This provides a great deal of flexibility in the OPC applications. By using the property control tags, users can implement redundancy by switching communications links or changing the device ID of a target device. Users can also provide access to the tags through special supervisory screens that allow a plant engineer to make changes to the communication parameters of the server if needed.

● **Note:** If there are errors when writing to read / write system tags, verify that the authenticated user has the appropriate permissions.

The tables below include descriptions of the following:

### [Application-Level System Tags](#)

### [Channel-Level System Tags for Ethernet Drivers](#)

### [Device-Level System Tags for both Serial and Ethernet Drivers](#)

## Application-Level System Tags

Syntax Example: <Channel Name>.<Device Name>.\_System.\_ActiveTagCount

Tag	Class	Description
_ActiveTagCount	Status Tag	The _ActiveTagCount tag indicates the number of tags that are currently active in the server.  This is a read-only tag.
_ClientCount	Status Tag	The _ClientCount tag indicates the number of clients that are currently connected to the server.  This is a read-only tag.
_Date	Status Tag	The _Date tag indicates the current date of the system that the server is running on. The format of this string is defined by the operating system date/time settings.  This is a read-only tag.
_DateTime	Status Tag	The _DateTime tag indicates the GMT date and time of the system that the server is running on. The format of the string is '2004-05-21T20:39:07.000'.  This is a read-only tag.

Tag	Class	Description
_DateTimeLocal	Status Tag	The <code>_DateTimeLocal</code> tag indicates the localized date and time of the system that the server is running on. The format of the string is '2004-05-21T16:39:07.000'.  This is a read-only tag.
_Date_Day	Status Tag	The <code>_Date_Day</code> tag indicates the current day of the month of the system on which the server is running.  This is a read-only tag.
_Date_DayOfWeek	Status Tag	The <code>_Date_DayOfWeek</code> tag indicates the current day of the week of the system on which the server is running. The format of the string is a number from 0 (Sunday) to 6 (Saturday).  This is a read-only tag.
_Date_Month	Status Tag	The <code>_Date_Month</code> tag indicates the current month of the system on which the server is running. The format of the string is a number (such as "9" instead of "September").  This is a read-only tag.
_Date_Year2	Status Tag	The <code>_Date_Year2</code> tag indicates the last two digits of the current year of the system on which the server is running.  This is a read-only tag.
_Date_Year4	Status Tag	The <code>_Date_Year4</code> tag indicates the current year of the system on which the server is running.  This is a read-only tag.
_ExpiredFeatures	Status Tag	The <code>_ExpiredFeatures</code> tag provides a list of all server features whose time-limited usage has expired. These features are no longer operational.  This is a read-only tag.
_FullProjectName	Status Tag	The <code>_FullProjectName</code> tag indicates the fully qualified path and file name to the currently loaded project.  This is a read-only tag.
_IsDemo	Status Tag	The <code>_IsDemo</code> tag is no longer available as the runtime will not enter Time Limited mode in version 6.0 or higher. See the <code>_TimeLimitedFeatures</code> , <code>_LicensedFeatures</code> , and <code>_ExpiredFeatures</code> tags to monitor the status of server features.
_LicensedFeatures	Status Tag	The <code>_LicensedFeatures</code> tag provides a list of all server features in use that have a valid license. These features are not subject to a time limit and will continue normal operation after any time-limited features expire.  This is a read-only tag.
_OpcClientNames	Status Tag	The <code>_OpcClientNames</code> tag is a String Array that lists the names of all OPC

Tag	Class	Description
		clients that connect to the server and register their name through the IOPCCommon::SetClientName method.  This is a read-only tag.
_ProductName	Status Tag	The _ProductName tag indicates the name of the underlying communication server.  This is a read-only tag.
_ProductVersion	Status Tag	The _ProductVersion tag indicates the version of the underlying communication server.  This is a read-only tag.
_ProjectName	Status Tag	The _ProjectName tag indicates the currently loaded project file name and does not include path information.  This is a read-only tag.
_ProjectTitle	Status Tag	The _ProjectTitle tag is a String tag that indicates the title of the project that is currently loaded.  This is a read-only tag.
_Time	Status Tag	The _Time tag indicates the current time of the system that the server is running on. The format of this string is defined by the operating system date/time settings.  This is a read-only tag.
_Time_Hour	Status Tag	The _Time_Hour tag indicates the current hour of the system on which the server is running.  This is a read-only tag.
_Time_Hour24	Status Tag	The _Time_Hour24 tag indicates the current hour of the system on which the server is running in a 24 hour format.  This is a read-only tag.
_Time_Minute	Status Tag	The _Time_Minute tag indicates the current minute of the system on which the server is running.  This is a read-only tag.
_Time_PM	Status Tag	The _Time_PM tag indicates the current AM/PM status of the system on which the server is running. This is a Boolean tag: 0 (False) indicates AM, and 1 (True) indicates PM.  This is a read-only tag.
_Time_Second	Status Tag	The _Time_Second tag indicates the current second of the system on which the server is running.  This is a read-only tag.

Tag	Class	Description
_TimeLimitedFeatures	Status Tag	<p>The _TimeLimitedFeatures tag provides a list of all server features that are time-limited and the time remaining (in seconds). When the time remaining expires, the feature will cease operation.</p> <p>This is a read-only tag.</p>
_TotalTagCount	Status Tag	<p>The _TotalTagCount tag indicates the total number of tags that are currently being accessed. These tags can be active or inactive.</p> <p><b>Note:</b> This count does not represent the number of tags configured in the project.</p> <p>This is a read-only tag.</p>

### Channel-Level System Tags for Ethernet Drivers

Syntax Example: <Channel name>.\_System.\_NetworkAdapter

Tag	Class	Description
_AvailableNetworkAdapters	Status Tag	<p>The _AvailableNetworkAdapters tag lists the available NICs and includes both unique NIC cards and NICs that have multiple IPs assigned to them. Additionally this tag also displays any WAN connections that are active, such as a dial-up connection. This tag is provided as a string tag and can be used to determine the network adapters available for use on this PC. The string returned contains all of the NIC names and their IP assignments. A semi-colon separates each unique NIC to allow the names to be parsed within an OPC application. For a serial driver, this tag is only used if Ethernet Encapsulation is selected.</p> <p>This is a read-only tag.</p>
_Description	Status Tag	<p>The _Description tag indicates the current user-defined text description for the channel it is referencing.</p> <p>This is a read-only tag.</p>
_EnableDiagnostics	Parameter Control Tag	<p>The _EnableDiagnostics tag allows the diagnostic system of the driver to be enabled and disabled. The diagnostic system places a little additional burden on the driver while enabled. As such the server allows diagnostics to be enabled or disabled to improve the driver's performance. When disabled, the Diagnostics tags will not be available. For more information, refer to <a href="#">Statistics Tags</a>.</p> <p>This is a read / write tag.</p>
_EncapsulationPort	Parameter Control	<p>The _EncapsulationPort tag controls the port used for Ethernet connections. The valid range is 0 to 65535.</p>

Tag	Class	Description
	Tag	This is a read / write tag.
_EncapsulationProtocol prop	Parameter Control Tag	The _EncapsulationProtocol tag controls the protocol used for Ethernet connections. Options include TCP/IP and UDP.  This is a read / write tag.
_FloatHandlingType	Parameter Control Tag	The _FloatHandlingType tag allows the current channel-level float handling to be changed. It exists in the channel-level _System folder. For more information, refer to <a href="#">Channel Properties — Advanced</a> .  This is a read / write tag.
_InterDeviceDelayMS	Parameter Control Tag	The _InterDeviceDelayMS tag specifies the amount of time that the channel will delay sending a request to the next device after the data has been received from the current device on the same channel. The valid range is 0 to 60000 milliseconds. The default setting is 0.  ● <b>Note:</b> This tag is only available on channels that use protocols that utilize the Inter-Device Delay.  This tag is a read / write tag.
_NetworkAdapter	Parameter Control Tag	The _NetworkAdapter tag allows the current NIC adapter in use by the driver to be changed at will. As a string tag, the name of the newly desired NIC adapter must be written to this tag in string format. The string written must match the exact description of the desired NIC for the change to take effect. NIC names can be obtained from the _AvailableNetworkAdapters tag listed above. For a serial driver, this tag will only be used if Ethernet Encapsulation is selected.  ● <b>Note:</b> When changing the NIC selection, the driver is forced to break all current device connections and reconnect.  This is a read / write tag.
_UnsolicitedEncapsulationPort	Parameter Control Tag	The _UnsolicitedEncapsulationPort tag controls the Ethernet port that has been opened to allow connections. The valid range is 0 to 65535.  This is a read / write tag.
_UnsolicitedEncapsulationProtocol	Parameter Control Tag	The _UnsolicitedEncapsulationProtocol tag controls the Ethernet protocol used to connect to the Unsolicited Encapsulation Port. Options include TCP/IP and UDP.  This is a read / write tag.

Tag	Class	Description
_WriteOptimizationDutyCycle	Parameter Control Tag	<p>The _WriteOptimizationDutyCycle tag allows the duty cycle of the write to read ratio to be changed at will. The duty cycle controls how many writes the driver will do for each read it performs. The _WriteOptimizationDutyCycle is defined as an unsigned long value. The valid range is 1 to 10 write per read. For more information, refer to <a href="#">Channel Properties — Write Optimizations</a>.</p> <p>This is a read / write tag.</p>

### Device-Level System Tags for both Serial and Ethernet Drivers

Syntax Example: <Channel Name>.<Device Name>.\_System.\_Error

Tag	Class	Description
_AutoCreateTagDatabase	Parameter Control Tag	<p>The _AutoCreateTagDatabase tag is a Boolean tag that is used to initiate the automatic OPC tag database functions of this driver for the device to which this tag is attached. When this tag is set True, the communications driver will attempt to automatically generate an OPC tag database for this device. This tag will not appear for drivers that do not support Automatic OPC Tag Database Generation.</p> <p>This is a read / write tag.</p>
_AutoDemoted	Status Tag	<p>The _AutoDemoted tag is a Boolean tag that returns the current auto-demoted state of the device. When False, the device is not demoted and is being scanned by the driver. When set True, the device is in demoted and not being scanned by the driver.</p> <p>This is a read-only tag.</p>
_AutoDemotionDiscardWrites	Parameter Control Tag	<p>The _AutoDemotionDiscardWrites tag is a Boolean tag that specifies whether or not write requests should be discarded during the demotion period. When this tag is set to False, all writes requests are performed regardless of the _AutoDemoted state. When this tag is set to True, all writes are discarded during the demotion period.</p> <p>This is a read / write tag.</p>
_AutoDemotionEnabled	Parameter Control Tag	<p>The _AutoDemotionEnabled tag is a Boolean tag that allows the device to be automatically demoted for a specific time period when the device is unresponsive. When this tag is set False, the device will never be demoted. When this tag is set True, the device is demoted when the _AutoDemotedFailureCount has been reached.</p> <p>This is a read / write tag.</p>
_AutoDemotedFailureCount	Parameter	<p>The _AutoDemotedFailureCount tag specifies how many suc-</p>



Tag	Class	Description
	Control Tag	cessive failures it takes to demote a device. The <code>_AutoDemotedFailureCount</code> is defined as a long data type. The valid range is 1 to 30. This tag can only be written to if <code>_AutoDemotionEnabled</code> is set to True.  This is a read / write tag.
<code>_AutoDemotionIntervalMS</code>	Parameter Control Tag	The <code>_AutoDemotionIntervalMS</code> tag specifies how long, in milliseconds, a device is demoted before re-attempting to communicate with the device. The <code>_AutoDemotionIntervalMS</code> is defined as a long data type. The valid range is 100 to 3600000 milliseconds. This tag can only be written to if <code>_AutoDemotionEnabled</code> is set to True.  This is a read / write tag.
<code>_ConnectTimeout</code>	Parameter Control Tag	The <code>_ConnectTimeout</code> tag allows the timeout associated with making an IP connection to a device to be changed at will. This tag is available when either a native Ethernet driver is in use or a serial driver is in Ethernet Encapsulation mode. The <code>_ConnectTimeout</code> is defined as a Long data type. The valid range is 1 to 30 seconds.  This is a read / write tag.
<code>_DemandPoll</code>	Status / Control Tag	The <code>_DemandPoll</code> tag issues a device read to all the active client items associated with the device. This is the equivalent of a client performing an asynchronous device read for those items. It takes priority over any scheduled reads that are supposed to occur for items that are being actively scanned.  The <code>_DemandPoll</code> tag becomes True (1) when written to. It returns to False (0) when the final active tag signals that the read requests have completed. Subsequent writes to the <code>_DemandPoll</code> tag will fail until the tag value returns to False. The demand poll respects the read / write duty cycle for the channel.  This is a read / write tag.
<code>_Description</code>	Status Tag	The <code>_Description</code> tag indicates the current user-defined text description for the device it is referencing.  This is a read-only tag.
<code>_DeviceId</code>	Parameter Control Tag	The <code>_DeviceId</code> tag allows the ID of the device to be changed at will. The data format of the <code>_DeviceId</code> depends on the type of device. For most serial devices this tag is a Long data type. For Ethernet drivers the <code>_DeviceId</code> is formatted as a string tag, allowing the entry of an IP address. In either case, writing a new device ID to this tag will cause the driver to change the target field device. This will only occur if the device ID written to this tag is correctly formatted and within the valid range for

Tag	Class	Description
		<p>the given driver.</p> <p>This is a read / write tag.</p>
_Enabled	Parameter Control Tag	<p>The _Enabled tag is a Boolean tag that allows the active state of the device to be turned On or Off. When this tag is set False, all other user-defined tags and data from this device is marked as invalid and writes will not be accepted for the device. When this tag is set True, normal communications will occur with the device.</p> <p>This is a read / write tag.</p>
_EncapsulationIp	Parameter Control Tag	<p>The _EncapsulationIp tag allows the IP of a remote terminal server to be specified and changed at will. This tag is only available on serial drivers that support <a href="#">Device Properties — Ethernet Encapsulation</a> mode. The _EncapsulationIp is defined as a string data type, allowing the entry of an IP address number. The server will reject entry of invalid IP addresses. This tag is only valid for a serial driver in Ethernet Encapsulation mode.</p> <p>This is a read / write tag.</p>
_EncapsulationPort	Parameter Control Tag	<p>The _EncapsulationPort tag allows the port number of the remote terminal server to be specified and changed. The _EncapsulationPort is defined as a long data type. The valid range is 0 to 65535. The port number entered in this tag must match that of the desired remote terminal server for proper Ethernet Encapsulation to occur. This tag is only valid for a serial driver in Ethernet Encapsulation mode.</p> <p>This is a read / write tag.</p>
_EncapsulationProtocol	Parameter Control Tag	<p>The _EncapsulationProtocol tag allows the IP protocol used for Ethernet Encapsulation to be specified and changed. The _EncapsulationProtocol is defined as a string data type. Writing either "TCP/IP" or "UDP" to the tag specifies the IP protocol. The protocol used must match that of the remote terminal server for proper Ethernet Encapsulation to occur. This tag is only valid for a serial driver in Ethernet Encapsulation mode.</p> <p>This is a read / write tag.</p>
_Error	Status Tag	<p>The _Error tag is a Boolean tag that returns the current error state of the device. When False, the device is operating properly. When set True, the driver has detected an error when communicating with this device. A device enters an error state if it has completed the cycle of request timeouts and retries without a response.</p> <p><b>Note:</b> For more information, refer to <a href="#">Device Properties — Timing</a>.</p>

Tag	Class	Description
		This is a read-only tag.
_FailedConnection	Status Tag	The _FailedConnection tag specifies that the connection failed. It is only available to specific drivers.  This is a read-only tag.
_InterRequestDelay	Parameter Control Tag	The _InterRequestDelay tag allows the time interval between device transactions to be changed at will. The _InterRequestDelay is defined as a Long data type. The valid range is 0 to 30000 milliseconds. This tag only applies to drivers that support this feature.  This is a read / write tag.
_RequestAttempts	Parameter Control Tag	The _RequestAttempts tag allows the number of communication attempts to be changed. The _RequestAttempts is defined as a Long value. The valid range is 1 to 10 attempts. This tag applies to all drivers equally.  This is a read / write tag.
_RequestTimeout	Parameter Control Tag	The _RequestTimeout tag allows the timeout associated with a data request to be changed at will. The _RequestTimeout tag is defined as a Long value. The valid range is 100 to 30000 milliseconds. This tag applies to all drivers equally.  This is a read / write tag.
_NoError	Status Tag	The _NoError tag is a Boolean tag that returns the current error state of the device. When True, the device is operating properly. When False, the driver has detected an error when communicating with this device. A device enters an error state if it has completed the cycle of request timeouts and retries without a response.  ● <b>Note:</b> For more information, refer to <a href="#">Device Properties — Timing</a> .  This is a read-only tag.
_ScanMode	Status Tag	The _ScanMode tag allows clients to dictate the method used for updates. It is defined as a String value, and corresponds to the user-specified Scan Mode setting (located in device properties). "Respect client specified scan rate" has a value of "UseClientRate," "Request data no faster than x" has a value of "UseFloorRate," and "Request all data at x" has a value of "ForceAllToFloorRate." The default setting is "Respect client specified scan rate."  This is a read-only tag.
_ScanRateMs	Status Tag	The _ScanRateMs tag corresponds to the _ScanMode tag, and

Tag	Class	Description
		is used when the Scan Mode is set to Request Data No Faster than Scan Rate or Request All Data at Scan Rate. This tag is defined as a DWord tag. The default setting is 1000 milliseconds.  This is a read-only tag.
_SecondsInError	Status Tag	The _SecondsInError tag is a DWord tag that displays the number of seconds since the device entered an error state. This tag displays 0 when the device is not in an error state.  This is a read-only tag.
_Simulated	Parameter Control Tag	The _Simulated tag is a Boolean tag that provides feedback about the simulation state of the current device. When read as True, this device is in a simulation mode. While in simulation mode, the server returns good data for this device, but does not attempt to communicate with the actual physical device. When tag is read as False, communication with the physical device is active. Changing the tag value allows clients to enable / disable simulated mode.  This is a read/write tag.

The \_System branch found under the DeviceName branch is always available. If referencing a system tag from a DDE application given the above example and the DDE defaults, the link would appear as "<DDE service name>|\_ddedata!Channel1.Device1.\_System.\_Error".

The \_Enabled tag provides a very flexible means of controlling the OPC applications. Additionally, using the \_Enable tag to allow the application to turn a particular device off while the physical device is being serviced can eliminate harmless but unwanted communications errors in the server's Event Log.

**See Also:**

- [Property Tags](#)
- [Statistics Tags](#)

### Property Tags


Property tags are used to provide read-only access to tag properties for client applications. To access a tag property, append the property name to the fully qualified tag address that has been defined in the server's tag database. For more information, refer to [Tag Properties — General](#).

If the fully qualified tag address is "Channel1.Device1.Tag1," its description can be accessed by appending the description property as "Channel1.Device1.Tag1.\_Description".

### Supported Property Tag Names

Tag Name	Description
_Name	The _Name property tag indicates the current name for the tag it is referencing.
_Address	The _Address property tag indicates the current address for the tag it is ref-

Tag Name	Description
	referencing.
_Description	The _Description property tag indicates the current description for the tag it is referencing.
_RawDataType	The _RawDataType property tag indicates the raw data type for the tag it is referencing.
_ScalingType	The _ScalingType property tag indicates the scaling type (None, Linear or Square Root) for the tag it is referencing.
_ScalingRawLow	The _ScalingRawLow property tag indicates the raw low range for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.
_ScalingRawHigh	The _ScalingRawHigh property tag indicates the raw high range for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.
_ScalingScaledDataType	The _ScalingScaledDataType property tag indicates the scaled to data type for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.
_ScalingScaledLow	The _ScalingScaledLow property tag indicates the scaled low range for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.
_ScalingScaledHigh	The _ScalingScaledHigh property tag indicates the scaled high range for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.
_ScalingClampLow	The _ScalingClampLow property tag indicates whether the scaled low value should be clamped for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.
_ScalingClampHigh	The _ScalingClampHigh property tag indicates whether the scaled high value should be clamped for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.
_ScalingUnits	The _ScalingUnits property tag indicates the scaling units for the tag it is referencing. If scaling is set to none this value contains the default value if scaling was applied.

 **See Also:**

[Statistics Tags](#)

[System Tags](#)

## Statistics Tags

Statistics tags are used to provide feedback to client applications regarding the operation of the channel communications in the server. Statistics tags are only available when diagnostics are enabled. *For more information, refer to Channel Diagnostics and OPC Diagnostics Viewer.*

Syntax Example: `<Channel Name>._Statistics._FailedReads`

## Supported Statistics Tag Names

Tag Name	Description
_SuccessfulReads	The _SuccessfulReads tag contains a count of the number of reads this channel has completed successfully since the start of the application or since the last time the _Reset tag was invoked. This tag is formatted as unsigned 32-bit integer and will eventually rollover. This tag is read only.
_SuccessfulWrites	The _SuccessfulWrites tag contains a count of the number of writes this channel has completed successfully since the start of the application or since the last time the _Reset tag was invoked. This tag is formatted as an unsigned 32-bit integer and will eventually rollover. This tag is read only.
_FailedReads	The _FailedReads tag contains a count of the number of reads this channel has failed to complete since the start of the application or since the last time the _Reset tag was invoked. This count is only incremented after the channel has failed the request based on the configured timeout and retry count for the device. This tag is formatted as an unsigned 32-bit integer and will eventually rollover. This tag is read only.
_FailedWrites	The _FailedWrites tag contains a count of the number of writes this channel has failed to complete since the start of the application or since the last time the _Reset tag was invoked. This count is only incremented after the channel has failed the request based on the configured timeout and retry count for the device. This tag is formatted as unsigned 32-bit integer and will eventually rollover. This tag is read only.
_RxBytes*	The _RxBytes tag contains a count of the number of bytes the channel has received from connected devices since the start of the application or since the last time the _Reset tag was invoked. This tag is formatted as unsigned 32-bit integer and will eventually rollover. This tag is read only.
_TxBytes	The _TxBytes tag contains a count of the number of bytes the channel has sent to connected devices since the start of the application or since the last time the _Reset tag was invoked. This tag is formatted as unsigned 32-bit integer and will eventually rollover. This tag is read only.
_Reset	The _Reset tag can be used to reset all diagnostic counters. The _Reset tag is formatted as a Boolean tag. Writing a non-zero value to the _Reset tag will cause the diagnostic counters to be reset. This tag is read / write.
_MaxPendingReads	The _MaxPendingReads tag contains a count of the maximum number of pending read requests for the channel since the start of the application (or the _Reset tag) was invoked. This tag is formatted as an unsigned 32-bit integer. The tag is read only.
_MaxPendingWrites	The _MaxPendingWrites tag contains a count of the maximum number of pending write requests for the channel since the start of the application (or the _Reset tag) was invoked. This tag is formatted as an unsigned 32-bit integer. The tag is read only.
_NextReadPriority	The _NextReadPriority is a channel-level system tag that reflects the priority level of the next read in the channel's pending read queue. Possible values are -1: No pending reads. 0: The next read is a result of a schedule-level demand poll or explicit read from a client. 1 - n: The next read is a result of scheduled read. This tag is read only.
_PendingReads	The _PendingReads tag contains a count of the current pending read requests for the channel. This tag is formatted as an unsigned 32-bit integer. The tag is read only.

Tag Name	Description
_PendingWrites	The _PendingWrites tag contains a count of the current pending write requests for the channel. This tag is formatted as an unsigned 32-bit integer. This tag is read only.

\* This statistical item is not updated in simulation mode ([See Device Properties](#)).

The \_Statistics branch (located beneath the channel branch) only appears when diagnostics are enabled for the channel. To reference a Diagnostics tag from a DDE application, given the above example and the DDE defaults, the link would appear as: "<DDE service name>|\_ddedata!Channel1.\_Statistics.\_SuccessfulReads".

The Diagnostics tag's value can also be viewed in the server by using the Communication Diagnostics Viewer. If Diagnostics Capture is enabled under Channel Properties, right-click on that channel and select **Diagnostics**.

 **See Also:**  
[System Tags](#)  
[Property Tags](#)

## Dynamic Tags

Dynamic tag addressing is a second method of defining tags that allows users to define tags only in the client application. As such, instead of creating a tag item in the client that addresses another tag item created in the server, users only need to create tag items in the client that directly accesses the device driver's addresses. On client connect, the server creates a virtual tag for that location and starts scanning for data automatically.

To specify an optional data type, append one of the following strings after the '@' symbol:

- BCD
- Boolean
- Byte
- Char
- Double
- DWord
- Float
- LBCD
- LLong
- Long
- QWord
- Short
- String
- Word

If the data type is omitted, the driver chooses a default data type based on the device and address being referenced. The default data types for all locations are documented in each individual driver's help documentation. If the data type specified is not valid for the device location, the server rejects the tag and an error posts in the Event Log.

## OPC Client Using Dynamic Addressing Example

Scan the 16-bit location "R0001" on the Simulator device. The following Dynamic tag examples assume that the project created is part of the example.

1. Start the OPC client application and connect to the server.
2. Using the Simulator Driver, create a channel and name it Channel1. Then, make a device and name it Device1.
3. In the client application, define an item name as "Channel1.Device1.R0001@Short."
4. The client project automatically starts receiving data. The default data type for address R0001 in the Simulator device is Word. To override this, the @Short has been appended to select a data type of Short.

**Note:** When utilizing Dynamic tags in an OPC client application, the use of the @[Data Type] modifier is not normally required. OPC clients can specify the desired data type as part of the request when registering a link for a specific data item. The data type specified by the OPC client is used if it is supported by the communications driver. The @[Data Type] modifier can be useful when ensuring that a communications driver interprets a piece of data exactly as needed.

### Non-OPC Client Example

Non-OPC clients can override the update rate on a per-tag basis by appending @[Update Rate].

For example, appending:

<DDE service name>|\_ddedata!Device1.R0001@500 overrides just the update rate.

<DDE service name>|\_ddedata!Device1.R0001@500,Short overrides both update rate and data type.

**Tips:**

1. The server creates a special Boolean tag for every device in a project that can be used by a client to determine whether a device is functioning properly. To use this tag, specify the item in the link as "Error." If the device is communicating properly, the tag's value is zero; otherwise, it is one.
2. If the device address is used as the item of a link such that the address matches the name of a user-defined tag in the server, the link references the address pointed to by the user-defined tag.
3. Static tags must be used to scale data in the server.

**See Also:**

[Static Tags \(User-Defined\)](#)

*Designing a Project: Adding User-Defined Tags*

### Tag Properties — Scaling

This server supports tag Scaling, which allows raw data from the device to be scaled to an appropriate range for the application.

**Type:** Select the method of scaling raw values. Select **Linear**, **Square Root**, or **None** to disable. The formulas for scaling types are shown below.

Type	Formula for Scaled Value
Linear	$\frac{((ScaledHigh - ScaledLow) / (RawHigh - RawLow)) * (RawValue - RawLow) + ScaledLow}$
Square root	$(Square\ root\ ((RawValue - RawLow) / (RawHigh - RawLow)) * (ScaledHigh - ScaledLow)) + ScaledLow$



**Raw Low:** Specify the lower end of the range of data from the device. The valid range depends on the raw tag data type. For example, if the raw value is Short, the valid range of the raw value would be from -32768 to 32767.

**Raw High:** Specify the upper end of the range of data from the device. The Raw High value must be greater than the Raw Low value. The valid range depends on the raw tag data type.

**Scaled Data Type:** Select the data type for the tag being scaled. The data type can be set to any valid OPC data type, including a raw data type, such as Short, to an engineering value with a data type of Long. The default scaled data type is Double.

**Scaled Low:** Specify the lower end of the range of valid resulting scaled data values. The valid range depends on the tag data type.

**Scaled High:** Specify the upper end of the range of valid resulting scaled data values. The valid range depends on the tag data type.

**Clamp Low:** Select **Yes** to prevent resulting data from exceeding the lower end of the range specified. Select **No** to allow data to fall outside of the established range.

**Clamp High:** Select **Yes** to prevent resulting data from exceeding the upper end of the range specified. Select **No** to allow data to fall outside of the established range.

**Negate Value:** Select **Yes** to force the resulting value to be negated before being passed to the client. Select **No** to pass the value to the client unmodified.

🔒 The server supports the OPC tag properties available in the 2.0 Data Access specifications. If the OPC client being used supports these properties, it can automatically configure the range of objects (such as user input objects or displays) by using the Scaling settings. Utilize the User Manager to restrict access rights to server features to prevent any unauthorized operator from changing these properties.

## What is a Tag Group?

---

This server allows tag groups to be added to the project. Tag groups are used to tailor the layout of OPC data into logical groupings that fit the application's needs. Tag groups allow multiple sets of identical tags to be added under the same device: this can be convenient when a single device handles a number of similar machine segments.

### Adding a Tag Group

Tag groups are defined by the set of tags contained. Tag groups are defined through the [Configuration API Service](#).

Tag group names are user-defined and should be logical for reporting and data analysis.

🔗 For information on reserved characters, refer to [How To... Properly Name a Channel, Device, Tag, and Tag Group](#).

### Removing a Tag Group

To remove a tag from the project; use the [Configuration API Service](#).

### Displaying Tag Group Properties

To review the tag group properties of a specific tag group via the Configuration API, access the [documentation channel endpoint](#).

---

## Tag Group Properties

From an OPC client standpoint, tag groups allow users to segregate OPC data into smaller tag lists, making finding specific tags easier.

Tag groups can be added at any level from the device-level down, and multiple tag groups can be nested together to fit the application's needs.

● **Note:** With the server's online full-time operation, these properties can be changed at any time. Any changes made to the tag groups take effect immediately. If the name is changed, OPC clients that have already used that tag group as part of an OPC item request are not affected until they release the item and attempt to reacquire it. New tag groups added to the project immediately allows browsing from an OPC client. Utilize the User Manager to restrict access rights to server features to prevent operators from changing the properties.

---

## What is the Alias Map?

The Alias Map provides both a mechanism for backwards compatibility with legacy server applications as well as a way to assign simple alias names to complex tag references. This is especially useful in client applications that limit the size of tag address paths. Although the latest version of the server automatically creates the alias map, users can add their own alias map entries to compliment those created by the server. Users can also filter the server created aliases so that the only ones visible are their own.

● **Note:** When enabled, the **Show auto-generated aliases** displays those alias maps created by the server automatically.

● **See Also:** [How to... Create and Use an Alias](#)

---

## Alias Properties

The Alias Map allows a way to assign alias names to complex tag references that can be used in client applications.

**Name:** Specify the alias name, which can be up to 256 characters long. It must be unique in the alias map. For information on reserved characters, refer to [How To... Properly Name a Channel, Device, Tag, and Tag Group](#).

**Description:** Enter a description of this alias to clarify data sources and reports (optional).

**Mapped to:** Specify or browse to the location of the alias. Because the alias map does not allow tag items to be browsed from the alias table, create a short nickname that replaces the address that leads up to the tag. This makes it easier to address items in a client application that does not support tag browsing.

**Scan Rate Override:** Specify an update rate to be applied to all non-OPC tags accessed using this alias map entry. The valid range is 0 to 99999990 milliseconds. The default is 0 milliseconds.

● **Tip:** This setting is equivalent to the topic update rate found in many DDE-only servers.

● **Note:** When set to 0 milliseconds, the server observes the scan rate set at the individual tag level.

---

## What is the Event Log?

The Event Log provides the date, time, and source of an error, warning, information, or security event. For more information, select a link from the list below.

### [Event Log Settings](#)

---

## Properly Name a Channel, Device, Tag, and Tag Group

When naming a channel, device, tag, or tag group, the following characters are reserved or restricted:

- Periods
- Double quotation marks
- Leading underscores
- Leading or trailing spaces

● **Note:** Some of the restricted characters can be used in specific situations. For more information, refer to the list below.

1. Periods are used in aliases to separate the original channel name and the device name. For example, a valid name is "Channel1.Device1".
2. Underscores can be used after the first character. For example, a valid name is "Tag\_1".
3. Spaces may be used within the name. For example, a valid name is "Tag 1".

---

## Configuration API Service

The Configuration API allows an HTTPS RESTful client to add, edit, read, and delete objects such as channels, devices, and tags in the server. The Configuration API offers the following features:

- Object definition in standard human-readable JSON data format
- Support for triggering and monitoring actions on some objects within the server
- Security via HTTP basic authentication and HTTP over SSL (HTTPS)
- Support for user-level access based on the User Manager and Security Policies Plug-In
- Transaction logging with configurable levels of verbosity and retention

● **Note:** This document assumes familiarity with HTTPS communication and REST concepts.

**Initialization** - The Configuration API is installed as a daemon and starts automatically with the system.

**Operation** - The Configuration API supports connections and commands between the server and REST clients.

If the Configuration API must be stopped, use the `systemctl` to stop the service.

### Security

REST clients to the Configuration API must use HTTPS Basic Authentication. The user credentials are defined in the server User Group.

Initial login to the Configuration API uses a username of administrator and the password set during installation. Additional users and should be created to allow the appropriate access.

### Documentation

● *Please consult additional information on properties, data ranges, endpoint mapping scheme, and acceptable actions for each endpoint is available at the Configuration API Landing Page at [https://<hostname\\_or\\_](https://<hostname_or_)*

*ip>:<port>/config/ (for default configurations).*

Documentation served from the landing page is HTML-encoded by default. To obtain JSON-encoded documentation, include an "Accept" request header with "application/json".

## Configuration API Service — Invoking Services

Objects may provide services if there are actions that can be invoked on the object beyond the standard CRUD (Create, Retrieve, Update, Delete) operations. Services provide an asynchronous programmatic interface through which remote clients can trigger and monitor these actions. Services can be found in a collection called 'services' underneath the object on which they operate. For example, the project load service is located at the `https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectLoad` endpoint as it operates on the project. Any object may provide services, so query if the service collection exists, then query the collection to see the available services.

### Service Architecture

Services are designed to provide stateless interaction with the object on which they operate. Services are comprised of two components: a service and a job. The job executes the work asynchronously and provides a mechanism through which a client can monitor the job for completion or for any errors that occurred during its operation. After a job completes, it is scheduled for deletion automatically by the server; no action is required by the client to clean up the job after it completes.

#### Service

The service is the interface through which an action is invoked. The service exposes all parameters that can be specified during its invocation as properties. To see the available parameters, perform a HTTPS GET on the service endpoint. All properties, besides the name and description of the service, are the parameters that can be included when invoking a service. Depending on the service, some or all parameters may be required.

Invocation of a service is accomplished by performing a HTTPS PUT request on the service endpoint with any parameters specified in the body of the request. Services may limit the total number of concurrent invocations. If the maximum number of concurrent invocations has been reached, the request is rejected with an "HTTPS 429 Too Many Requests" response. If the limit has not been reached, the server responds with an "HTTPS 202 Accepted" response and the body of the response including a link to the newly created job.

Successful PUT response example:

```
{
  "code": 202,
  "message": "Accepted",
  "href": "https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectLoad/jobs/job1"
}
```

Busy PUT response example:

```
{
  "code": 429,
  "message": "The server is busy. Retry the operation at a later time."
}
```

#### Job

The job represents a specific request accepted by the server. To check the status of a job perform a HTTPS GET request on the job endpoint. The **servermain.JOB\_COMPLETE** property represents the current state of the job as a Boolean. The value of this property remains false until the job has finished executing. If the job fails to execute for any reason, it provides the client with an appropriate error message in the **servermain.JOB\_STATUS\_MSG** property.

### Job Cleanup

Jobs are automatically deleted by the server after a configurable amount of time. By default, after a job has completed, the client has 30 seconds to interact with it before the job is deleted. If a longer amount of time is required by the client or the client is operating over a slow connection, the client can use the **servermain.JOB\_TIME\_TO\_LIVE\_SECOND** parameter when invoking the service to increase the time-to-live up to a maximum of five minutes. Each job has its own time-to-live and it may not be changed after a job has been created. Clients are not allowed to manually delete jobs from the server so it is best to choose the shortest time-to-live without compromising the client's ability to get the information from the job before it is deleted.

### Service Automatic Tag Generation

The Automatic Tag Generation service operates under a device endpoint for a driver that supports Automatic Tag Generation. The properties that support Automatic Tag Generation for the device must be configured prior to initiating Automatic Tag Generation. *See the driver specific documentation for related properties.*

To initiate Automatic Tag Generation, a PUT is sent to the TagGeneration endpoint with a defined empty payload. In the following example, Automatic Tag Generation is initiated on Channel1/Device1.

Endpoint (PUT):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/services/TagGeneration
```

The response should look something like the following.

Body:

```
{
  "code": 202,
  "message": "Accepted",
  "href": "https://<hostname_or_ip>:<-
port>/-
con-
fig/v1/project/channels/Channel1/devices/Device1/services/TagGeneration/jobs/job1"
}
```

This means the request was accepted and the job was created as job1. The status of the job can be seen by querying the job. This is done by sending a GET to the job's endpoint. The GET request should look like the following.

Endpoint (GET):

```
https://<hostname_or_ip>:<-
port>/-
con-
fig/v1/project/channels/Channel1/devices/Device1/services/TagGeneration/jobs/job1
```

Jobs are automatically cleaned up after their wait time has expired. This wait time is configurable.

• See the [Job Cleanup](#) section for more information.

**Note:** Not all drivers support Automatic Tag Generation.

**Tip:** Automatic Tag Generation files must be located in the <installation\_directory>/user\_data directory. All files in the user\_data directory must be world readable or owned by the ThingWorx Kepware Edge user and group that were created during installation, by default this is tkedge.

## Service Project Load

Projects can be loaded by interacting with the ProjectLoad service on the ProjectLoad endpoint. First a GET request must be sent to get the Project ID to later be used in the PUT request.

The GET request should look like the following.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/ProjectLoad
```

The server should respond with something similar to the following.

Body:

```
{
  "PROJECT_ID": 3531905431,
  "common.ALLTYPES_NAME": "ProjectLoad",
  "servermain.JOB_TIME_TO_LIVE_SECONDS": 30,
  "servermain.PROJECT_FILENAME": "",
  "servermain.PROJECT_PASSWORD": ""
}
```

To initiate the project load, a PUT request is sent to the server with the absolute path to the project file, the project file password, and the Project ID. If there is no password on the project, that field is not required. Project loading supports SLPF, LPF, and JSON file types. The request should look similar to the following.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectLoad
```

Body:

```
{
  "PROJECT_ID": 3531905431,
  "servermain.PROJECT_FILENAME": "/Absolute/Path/To/MyProject.json",
  "servermain.PROJECT_PASSWORD": ""
}
```

The server should respond with something similar to the following.

Body:

```
{
  "code": 202,
  "message": "Accepted",
  "href": "https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectLoad/jobs/job1"
}
```

This means the request was accepted and the job was created as job1. The status of the job can be seen by querying the job. This is done by sending a GET to the job's endpoint. The GET request should look like the following.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/ProjectLoad/jobs/job1
```

Jobs are automatically cleaned up after their wait time has expired. This wait time is configurable.

• See the [Job Cleanup](#) section for more information.

## Service Project Save

Projects can be loaded by interacting with the ProjectSave service on the ProjectSave endpoint. A GET request must be sent to get the Project ID to later be used in the PUT request. The GET request should look similar to the following.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/ProjectSave
```

The server should respond with something similar to the following.

Body:

```
{
  "PROJECT_ID": 2401921849,
  "common.ALLTYPES_NAME": "ProjectSave",
  "servermain.JOB_TIME_TO_LIVE_SECONDS": 30,
  "servermain.PROJECT_FILENAME": ""
}
```

To initiate the project save a PUT request is sent with the project file path and name of the file with the extension (SLPF, LPF, or JSON), the password to encrypt it with, and the Project ID. The password property is required for SLPF file and ignored otherwise. The path is relative to the Application Data Folder. The PUT request should look similar to the following.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectSave
```

Body:

```
{
  "PROJECT_ID": 2401921849,
  "servermain.PROJECT_FILENAME": "Projects/MyProject.SLPF",
  "servermain.PROJECT_PASSWORD": "MyPassword"
}
```

The server should respond with something similar to the following.

Body:

```
{
  "code": 202,
  "message": "Accepted",
  "href": "https://<hostname_or_ip>:<port>/config/v1/project/services/ProjectSave/jobs/job1"
}
```

This means the request was accepted and the job was created as job1. The status of the job can be seen by querying the job. This is done by sending a GET to the job's endpoint. The GET request should look like the following.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/ProjectLoad/jobs/job1
```

Jobs are automatically cleaned up after their wait time has expired. This wait time is configurable.

• See the [Job Cleanup](#) section for more information.

---

## Channel Properties — Configuration API

The following properties define a channel using the Configuration API service.

### General Properties

`common.ALLTYPES_NAME` \* Required parameter.

• **Note:** Changing this property causes the API endpoint URL to change.

`common.ALLTYPES_DESCRIPTION`

`servermain.MULTIPLE_TYPES_DEVICE_DRIVER` \* Required parameter

### Ethernet Communication Properties

`servermain.CHANNEL_ETHERNET_COMMUNICATIONS_NETWORK_ADAPTER_STRING`

### Advanced Properties

`servermain.CHANNEL_NON_NORMALIZED_FLOATING_POINT_HANDLING` \* Required parameter

### Write Optimizations

`servermain.CHANNEL_WRITE_OPTIMIZATIONS_METHOD`

`servermain.CHANNEL_WRITE_OPTIMIZATIONS_DUTY_CYCLE`

• **See Also:** *The server help system Configuration API Service section.*

---

## Configuration API Service — Creating a Channel

To create a channel via the Configuration API service, only a minimum set of properties are required; all others are set to the default value. Once a channel is defined, its properties and settings are used by all devices assigned to that channel. The specific properties are dependent on the protocol or driver selected.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the channel endpoint.

The example below creates a channel named Channel1 that uses the Simulator driver on a server running on the local host.



Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels
```

Body:

```
{
  "common.ALLTYPES_NAME": "Channel1",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator"
}
```

Refer to the driver specific help documentation to find out what properties are required to create a channel for that driver.

## Configuration API Service — Updating a Channel

To update a property or collection of properties on a channel, a GET request must first be sent to the endpoint to be updated to get the Project ID.

For more information about the Project ID see the Concurrent Clients section.

In the example below, the channel being updated is Channel1.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/Channel1
```

The GET request will return a JSON blob similar to the following.

Body:

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Channel1",
  "common.ALLTYPES_DESCRIPTION": "",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator",
  "servermain.CHANNEL_UNIQUE_ID": 2154899492,
  "servermain.CHANNEL_WRITE_OPTIMIZATIONS_METHOD": 2,
  ...
}
```

To update or change a channel property, a PUT request is sent to the channel with the Project ID and the new property value defined. In the following example, the channel name will change from Channel1 (from above) to Simulator.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/Channel1
```

Body:

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Simulator"
}
```

Following the PUT, a GET can be sent to the channel's endpoint to validate that the property changed. In this case, because the name was changed, the endpoint also changed and the GET request would be the following.

● **Note:** Some properties are client restricted and cannot be changed when a client is connected.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/Simulator
```

The response from the GET request should show the property value has changed. The response to the GET above should look similar to the following:

Body:

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Simulator",
  "common.ALLTYPES_DESCRIPTION": "",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator",
  "servermain.CHANNEL_UNIQUE_ID": 2154899492,
  "servermain.CHANNEL_WRITE_OPTIMIZATIONS_METHOD": 2,
  ...
}
```

---

## Configuration API Service — Removing Channel

To remove a channel, send a DELETE command to the channel endpoint to be removed. This causes the channel and all of its children to be removed.

In the example below, the channel Simulator will be removed.

Endpoint (DELETE):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/Simulator
```

This can be verified by sending a GET to the removed endpoint. The server will respond with an error. It can also be verified with a GET to the "channels" endpoint; the removed channel will not be in the list of channels returned from the GET request.

---

## Device Properties — Configuration API

The following properties define a device using the Configuration API service.

### General Properties

common.ALLTYPES\_NAME

common.ALLTYPES\_DESCRIPTION

servermain.DEVICE\_CHANNEL\_ASSIGNMENT

servermain.MULTIPLE\_TYPES\_DEVICE\_DRIVER

servermain.DEVICE\_MODEL

servermain.DEVICE\_ID\_STRING

servermain.DEVICE\_DATA\_COLLECTION

```
servermain.DEVICE_SIMULATED
```

## Scan Mode

```
servermain.DEVICE_SCAN_MODE * Required parameter
```

```
servermain.DEVICE_SCAN_MODE_RATE_MS
```

```
servermain.DEVICE_SCAN_MODE_RATE_MS
```

```
servermain.DEVICE_SCAN_MODE_PROVIDE_INITIAL_UPDATES_FROM_CACHE
```

## Auto Demotion

```
servermain.DEVICE_AUTO_DEMOTION_ENABLE_ON_COMMUNICATIONS_FAILURES
```

```
servermain.DEVICE_AUTO_DEMOTION_DEMOTE_AFTER_SUCESSIVE_TIMEOUTS
```

```
servermain.DEVICE_AUTO_DEMOTION_PERIOD_MS
```

```
servermain.DEVICE_AUTO_DEMOTION_DISCARD_WRITES
```


## Tag Generation


```
servermain.DEVICE_TAG_GENERATION_ON_STARTUP * Required parameter
```

```
servermain.DEVICE_TAG_GENERATION_DUPLICATE_HANDLING * Required parameter
```

```
servermain.DEVICE_TAG_GENERATION_GROUP
```

```
servermain.DEVICE_TAG_GENERATION_ALLOW_SUB_GROUPS
```

 **Tip:** To Invoke Automatic Tag Generation, send a PUT with an empty body to the TagGeneration service endpoint on the device.

 **Note:** All files in the user\_data directory must be world readable or owned by the ThingWorx Kepware Edge user and group that were created during installation, by default tkedge.

 **See Also:** For more information see *Services help*.


## Timing

```
servermain.DEVICE_CONNECTION_TIMEOUT_SECONDS
```

```
servermain.DEVICE_REQUEST_TIMEOUT_MILLISECONDS
```

```
servermain.DEVICE_RETRY_ATTEMPTS
```

```
servermain.DEVICE_INTER_REQUEST_DELAY_MILLISECONDS
```

 **See Also:** The server help system *Configuration API Service section*.

## Configuration API Service — Creating a Device

To create a device via the Configuration API service, only a minimum set of properties are required; all others are set to the default value. The specific properties are dependent on the protocol or driver selected.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the devices endpoint under a channel.

The example below will create a device named Device1 under Channel1 that uses the Simulator driver on a server running on the local host.

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/Channel1/devices
```

Body:

```
{
  "common.ALLTYPES_NAME": "Device1",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator"
}
```

Refer to the driver specific help documentation to find out what properties are required to create a device for that driver.

## Configuration API Service — Updating a Device

To update a property or collection of properties on a device, a GET request must first be sent to the endpoint to be updated to get the Project ID.

For more information about the Project ID, see the [Concurrent Clients](#) section.

In the example below, the device being updated is Device1 under Channel1.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/Channel1/devices/Device1
```

The GET request will return a JSON blob similar to the following.

Body:

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Device1",
  "common.ALLTYPES_DESCRIPTION": "",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator",
  "servermain.DEVICE_MODEL": 0,
  "servermain.DEVICE_UNIQUE_ID": <project_ID_from_GET>,
  "servermain.DEVICE_CHANNEL_ASSIGNMENT": "Channel1",
  ...
}
```

To update or change a device property a PUT request is sent to the device with the Project ID and the new property value defined. In the following example the device name will change from Device1 (from above) to Simulator.

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/project/channels/Channel1/devices/Device1
```

Body:

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Simulator"
}
```

Following the PUT, a GET can be sent to the device endpoint to validate that the property changed. In this case, because the name was changed, the endpoint also changed and the GET request would be the following.

● **Note:** Some properties are client restricted and cannot be changed when a client is connected.

Endpoint (GET):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Simulator
```

The response from the GET request will show the property value has changed. The response to the GET above should look similar to the following.

Body:

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Simulator",
  "common.ALLTYPES_DESCRIPTION": "",
  "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator",
  "servermain.DEVICE_MODEL": 0,
  "servermain.DEVICE_UNIQUE_ID": <device_ID_from_GET>,
  "servermain.DEVICE_CHANNEL_ASSIGNMENT": "Channel1",
  ...
}
```

## Configuration API Service — Removing a Device

To remove a device, send a DELETE to the device endpoint to be removed. This will cause the device and all of its children to be removed.

In the example below, the device Simulator will be removed.

Endpoint (DELETE):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Simulator
```

This can be verified by sending a GET to the removed endpoint. The server will respond with an error. It can also be verified with a get to the devices endpoint and the removed device will not be in the list of devices returned from the GET request.

## Configuration API Service — Creating a Tag

To create a tag via the Configuration API service, only a minimum set of properties are required; all others are set to the default value. The specific properties are dependent on the protocol or driver selected.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the tags endpoint under a device.

The example below will create a tag named MyTag for address R5 under Channel1/Device1 that uses the Simulator driver on a server running on the local host.

Endpoint (POST):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tags
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyTag",
  "servermain.TAG_ADDRESS": "R5"
}
```

Tags can also be created within a tag group. The process for adding a tag group is the same except the URL will change to include the tag\_group endpoint and the group name.

In the following example, the tag group RampTags already exists and a tag named MyTag will be created under it with the address R5.

• For more information on creating a tag group [see Creating a Tag Group](#) section.

Endpoint (POST):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tag_group/RampTags/tags
```

Body:

```
{
  "common.ALLTYPES_NAME": "MyTag",
  "servermain.TAG_ADDRESS": "R5"
}
```

• Refer to the driver specific help documentation to find out what properties are required to create a tag for that driver.

## Configuration API Service — Updating a Tag

To update a property or collection of properties on a tag, a GET request must first be sent to the endpoint to be updated to get the Project ID.

• For more information about the Project ID see the [Concurrent Clients](#) section.

In the example below, the tag being updated is MyTag under Channel1/Device1.

Endpoint (GET):

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tags/MyTag
```

The GET request will return a JSON blob similar to the following.

Body:

```
{
  "PROJECT_ID": <project_ID_from_GET>,
}
```

```

"common.ALLTYPES_NAME": "MyTag",
"common.ALLTYPES_DESCRIPTION": "",
"servermain.TAG_ADDRESS": "R0005",
"servermain.TAG_DATA_TYPE": 5,
"servermain.TAG_READ_WRITE_ACCESS": 1,
"servermain.TAG_SCAN_RATE_MILLISECONDS": 100,
...

```

To update or change a tag property, a PUT request is sent to the tag with the Project ID and the new property value defined.

In the following example, the tag name will change from MyTag (from above) to Tag1.

Endpoint (PUT):

```

https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tags/MyTag

```

Body:

```

{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Tag1"
}

```

Following the PUT a GET can be sent to the tag's endpoint to validate that the property changed. In this case, because the name was changed, the endpoint also changed and the GET request would be the following.

Endpoint (GET):

```

https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tags/Tag1

```

The response from the GET request will show the property value has changed. The response to the GET above should look similar to the following.

Body:

```

{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "Tag1",
  "common.ALLTYPES_DESCRIPTION": "",
  "servermain.TAG_ADDRESS": "R0005",
  "servermain.TAG_DATA_TYPE": 5,
  "servermain.TAG_READ_WRITE_ACCESS": 1,
  "servermain.TAG_SCAN_RATE_MILLISECONDS": 100,
  ...
}

```

## Configuration API Service — Removing a Tag

To remove a tag, send a DELETE to the tag's endpoint to be removed. This will cause the tag and all of its children to be removed.

In the example below, the tag Tag1 will be removed.

Endpoint (DELETE):

```
https://<hostname_or_ip>:<-  
port>/config/v1/project/channels/Channel1/devices/Device1/tags/Tag1
```

This can be verified by sending a GET to the removed endpoint. The server will respond with an error. It can also be verified with a get to the tags endpoint and the removed tag will not be in the list of tags returned from the GET request.

## Configuration API Service — Creating a Tag Group

To create a tag group via the Configuration API service, only a group name is required.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the tag\_groups endpoint under a device.

The example below will create a tag group named RampTags under Channel1/Device1 that uses the Simulator driver on a server running on the local host.

Endpoint (POST):

```
https://<hostname_or_ip>:<-  
port>/config/v1/project/channels/Channel1/devices/Device1/tag_groups
```

Body:

```
{  
  "common.ALLTYPES_NAME": "RampTags"  
}
```

Tag groups can have tags and more tag groups nested under them. *To add a Tag, see the [Creating a Tag](#) section.*

To nest a Tag Group within another group, another POST action is required to add the existing group name and the tag\_groups endpoint to the end of the URL.

Continuing the example above, the new request would look like the following.

Endpoint (POST):

```
https://<hostname_or_ip>:<-  
port>/config/v1/project/channels/Channel1/devices/Device1/tag_groups/RampTags/tag_  
groups
```

Body:

```
{  
  "common.ALLTYPES_NAME": "1-10"  
}
```

## Configuration API Service — Updating a Tag Group

To update a property or collection of properties on a tag, a GET request must first be sent to the endpoint to be updated to get the Project ID.

• For more information about the Project ID, see the [Concurrent Clients](#) section.

In the example below, the tag group being updated is RampTags under Channel1/Device1.



**Endpoint (GET):**

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tag_groups/RampTags
```

The GET request will return a JSON blob similar to the following.

**Body:**

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "RampTags",
  "common.ALLTYPES_DESCRIPTION": "",
  "servermain.TAGGROUP_LOCAL_TAG_COUNT": 0,
  "servermain.TAGGROUP_TOTAL_TAG_COUNT": 0,
  "servermain.TAGGROUP_AUTOGENERATED": false
}
```

To update or change a tag group property, a PUT request is sent to the tag group with the Project ID and the new property value defined.

In the following example, the tag group name will change from RampTags (from above) to RampGroup.

**Endpoint (PUT):**

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tags/MyTag
```

**Body:**

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "RampGroup"
}
```

Following the PUT, a GET can be sent to the tag group endpoint to validate that the property changed. In this case, because the name was changed, the endpoint also changed and the GET request would be the following.

**Endpoint (GET):**

```
https://<hostname_or_ip>:<-
port>/config/v1/project/channels/Channel1/devices/Device1/tag_groups/RampGroup
```

The response from the GET request will show the property value has changed. The response to the GET above should look similar to the following.

**Body:**

```
{
  "PROJECT_ID": <project_ID_from_GET>,
  "common.ALLTYPES_NAME": "RampTags",
  "common.ALLTYPES_DESCRIPTION": "",
  "servermain.TAGGROUP_LOCAL_TAG_COUNT": 0,
  "servermain.TAGGROUP_TOTAL_TAG_COUNT": 0,
  "servermain.TAGGROUP_AUTOGENERATED": false
}
```

## Configuration API Service — Removing a Tag Group

To remove a tag group, send a DELETE to the tag group endpoint to be removed. This will cause the tag group and all of its children to be removed. In the example below the tag group RampGroup will be removed.

Endpoint (DELETE):

```
https://<hostname_or_ip>:<-  
port>/config/v1/project/channels/Channel1/devices/Device1/tag_groups/RampGroup
```

This can be verified by sending a GET to the removed endpoint. The server will respond with an error. It can also be verified with a get to the tag\_groups endpoint and the removed tag group will not be in the list of tag groups returned from the GET request.

## Configuration API Service — User Management

The User Manager controls client access to the project's objects (which are the channels, devices, tags, etc.) and their corresponding functions. The User Manager allows permissions to be specified by user groups. For example, the User Manager can restrict user access to project tag data based on its permissions from the parent user group.

The User Manager has four built-in groups that each contain a built-in user. The default groups are Administrators, Server Users, Anonymous Clients, and ThingWorx Interface Users. The default users in these groups are Administrator, Default User, Data Client, and ThingWorx Interface. Users cannot rename or change the description fields of built-in user groups or users. Neither the default groups nor the default users can be disabled.

To allow adequate access for data transfer between the server and the ThingWorx Platform, project modification must be enabled for the ThingWorx Interface Users group. The request to grant the correct access for this functionality should look similar to the following:

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/admin/server_usergroups/ThingWorx Inter-  
face Users/project_permissions/Servermain Project
```

Body:

```
{  
  "libadminsettings.USERMANAGER_PROJECTMOD_EDIT": true  
}
```

### Notes:

1. Although the Administrator's group settings cannot be changed, additional administrative users can be added to the group.
2. A project cannot load without correct user information.
3. There is no "Project\_ID" property on the User Management endpoints. All PUTs are accepted and the last PUT to a given endpoint is applied.

## User Groups

**Endpoint:** https://<hostname\_or\_ip>:<port>/config/v1/admin/server\_usergroups

**Supported Actions**

HTTP(S) Verb	Action
POST	Create the specified group
GET	Retrieves a list of all groups

**Endpoint:** https://<hostname\_or\_ip>:<port>/config/v1/admin/server\_usergroups/<GroupName>

**Supported Actions**

HTTP(S) Verb	Action
GET	Retrieves the specified group
PUT	Updates the specified group

**Properties**

Property Name	Type	Required	Description
common.ALLTYPES_NAME	String	Yes	Specify the identity of this object.
common.ALLTYPES_DESCRIPTION	String	No	Provide a brief summary of this object or its use.
libadminsettings.USERMANAGER_GROUP_ENABLED	Enable/Disable	No	The group's enabled-state takes precedence over the users enabled state.
libadminsettings.USERMANAGER_IO_TAG_READ	Enable/Disable	No	Allow/deny clients belonging to the group to access I/O tag data.
libadminsettings.USERMANAGER_IO_TAG_WRITE	Enable/Disable	No	Allow/deny clients belonging to the group to modify I/O tag data. Note: When USERMANAGER_IO_TAG_READ is set to false, this property is also set to false and disabled to prevent write-only tags.
libadminsettings.USERMANAGER_IO_TAG_DYNAMIC_ADDRESSING	Enable/Disable	No	Allow/deny clients belonging to the group to add items using dynamic addressing.
libadminsettings.USERMANAGER_SYSTEM_TAG_READ	Enable/Disable	No	Allow/deny clients belonging to the group to access system tag data.
libadminsettings.USERMANAGER_SYSTEM_TAG_WRITE	Enable/Disable	No	Allow/deny clients belonging to the group to modify system tag data. Note: When USERMANAGER_SYSTEM_TAG_READ is set to false, this property is also set to false and disabled to prevent write-only tags.
libadminsettings.USERMANAGER_INTERNAL_TAG_READ	Enable/Disable	No	Allow/deny clients belonging to the group to access internal tag data.
libadminsettings.USERMANAGER_INTERNAL_TAG_WRITE	Enable/Disable	No	Allow/deny clients belonging to the group to modify internal tag data. Note: When USERMANAGER_INTERNAL_TAG_READ is set to false, this property is also set to false and disabled to prevent write-only tags.

Property Name	Type	Required	Description
libadminsettings.USERMANAGER_SERVER_MANAGE_LICENSES	Enable/Disable	No	Allow/deny users belonging to the group to access the license manager.
libadminsettings.USERMANAGER_SERVER_MODIFY_SERVER_SETTINGS	Enable/Disable	No	Allow/deny users belonging to the group to access this property sheet.
libadminsettings.USERMANAGER_SERVER_DISCONNECT_CLIENTS	Enable/Disable	No	Allow/deny users belonging to the group to take action that can cause data clients to be disconnected.
libadminsettings.USERMANAGER_SERVER_RESET_EVENT_LOG	Enable/Disable	No	Allow/deny users belonging to the group to clear all logged event messages.
libadminsettings.USERMANAGER_SERVER_OPCTUA_DOTNET_CONFIGURATION	Enable/Disable	No	Allow/deny users belonging to the group to access the OPC UA or XI configuration manager.
libadminsettings.USERMANAGER_SERVER_CONFIG_API_LOG_ACCESS	Enable/Disable	No	Allow/deny users belonging to the group to access the Configuration API Transaction Log.
libadminsettings.USERMANAGER_SERVER_REPLACE_RUNTIME_PROJECT	Enable/Disable	No	Allow/deny users belonging to the group to replace the running project.
libadminsettings.USERMANAGER_BROWSE_BROWSENAMESPACE	Enable/Disable	No	Allow/deny clients belonging to the user group to browse the project namespace.

### Project Permissions

**Endpoint:** [https://<hostname\\_or\\_ip>:<port>/config/v1/admin/server\\_usergroups/<GroupName>/project\\_permissions](https://<hostname_or_ip>:<port>/config/v1/admin/server_usergroups/<GroupName>/project_permissions)

#### Supported Actions

HTTP(S) Verb	Action
GET	Retrieves a list of all project permissions

### Child Endpoints

#### Properties

Endpoint	Description
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Alias	Configure default 'Servermain Alias' access permissions for the selected user group.
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Channel	Configure default 'Servermain Channel' access permissions for the selected user group.
/config/v1/admin/server_usergroups/<GroupName>/project_per-	Configure default 'Servermain Device' access permissions for the selected user group.

Endpoint	Description
missions/Servermain Device	
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Meter Order	Configure default 'Servermain Meter Order' access permissions for the selected user group. ● <b>Note:</b> Add and delete properties are disabled for this endpoint.
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Phone Number	Configure default 'Servermain Phone Number' access permissions for the selected user group.
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Phone Priority	Configure default 'Servermain Phone Priority' access permissions for the selected user group. ● <b>Note:</b> Add and delete properties are disabled for this endpoint.
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Project	Configure default 'Servermain Project' access permissions for the selected user group. ● <b>Note:</b> Add and delete properties are disabled for this endpoint.
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Tag	Configure default 'Servermain Tag' access permissions for the selected user group.
/config/v1/admin/server_usergroups/<GroupName>/project_permissions/Servermain Tag Group	Configure default 'Servermain Tag Group' access permissions for the selected user group.

**Endpoint:** https://<hostname\_or\_ip>:<port>/config/v1/admin/server\_usergroups/<GroupName>/project\_permissions/<PermissionName>

### Supported Actions

HTTP(S) Verb	Action
GET	Retrieves the specified project permission
PUT	Updates the specified project permission

### Properties

Property Name	Type	Description
common.ALLTYPES_NAME	String	Specify the identity of this object.
common.ALLTYPES_DESCRIPTION	String	Provide a brief summary of this object or its use.
libadminsettings.USERMANAGER_PROJECTMOD_ADD	Enable/Disable	Allow/deny users belonging to the group to add this type of object.
libadminsettings.USERMANAGER_PROJECTMOD_EDIT	Enable/Disable	Allow/deny users belonging to the group to edit this type of object.
libadminsettings.USERMANAGER_PROJECTMOD_DELETE	Enable/Disable	Allow/deny users belonging to the group to delete this type of object.

### Users

**Endpoint:** https://<hostname\_or\_ip>:<port>/config/v1/admin/server\_users

**Supported Actions**

HTTP(S) Verb	Action
POST	Create the specified user
GET	Retrieves a list of all users

**Endpoint:** https://<hostname\_or\_ip>:<port>/config/v1/admin/server\_users/<UserName>

**Supported Actions**

HTTP(S) Verb	Action
GET	Retrieves the specified user
PUT	Updates the specified user

**Properties**

Property Name	Type	Required	Description
common.ALLTYPES_NAME	String	Yes	Specify the identity of this object.
common.ALLTYPES_DESCRIPTION	String	No	Provide a brief summary of this object or its use.
libadminsettings.USERMANAGER_USER_GROUPNAME	String	Yes	The name of the parent group.
libadminsettings.USERMANAGER_USER_ENABLED	Enable/Disable	No	The group's enabled-state takes precedence over the users enabled state.
libadminsettings.USERMANAGER_USER_PASSWORD	Password	No	The user's password. This is case-sensitive. ● <b>Note:</b> The password must be at least 14 characters and include a mix of uppercase and lowercase letters, numbers, and special characters. Avoid well-known, easily guessed, or common passwords.

● **Note:** If there are errors when writing to read / write system tags, verify that the authenticated user has the appropriate permissions.

**Configuration API Service — Creating a User**

To create a user via the Configuration API service, only a minimum set of properties are required; all others are set to the default value.

● Only members of the Administrators group can create users.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the server\_users endpoint.

The example below creates a user named User1 that is a member of the Administrators user group:

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/admin/server_users
```

Body:

```
{
  "common.ALLTYPES_NAME": "User1",
  "libadminsettings.USERMANAGER_USER_GROUPNAME": "Administrators",
  "libadminsettings.USERMANAGER_USER_PASSWORD": "<Password>"
}
```

## Configuration API Service — Creating a User Group

To create a group via the Configuration API service, only a minimum set of properties are required; all others are set to the default value. Once a user group is defined, its permissions are used by all users assigned to that user group.

🟢 Only members of the Administrators group can create user groups.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the `server_usergroups` endpoint.

The example below creates a user group named Operators:

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/admin/server_usergroups
```

Body:

```
{
  "common.ALLTYPES_NAME": "Operators",
}
```

## Configuration API Service — Updating a User

To update a user via the Configuration API service, provide new values for the properties that require updating.

🟢 Only members of the Administrators group can update users.

🔒 There is no `PROJECT_ID` field for users.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the `server_users/<username>` endpoint.

The example below updates the user named User1 to add a description and move it to a different user group:

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/admin/server_users/User1
```

Body:

```
{
  "common.ALLTYPES_DESCRIPTION": "The user account of User1",
  "libadminsettings.USERMANAGER_USER_GROUPNAME": "Operators"
}
```

## Configuration API Service — Updating a User Group

To edit a user group via the Configuration API service, provide new values for the properties that require updating.

- Only members of the Administrators group can update user groups.
- There is no PROJECT\_ID field for user groups.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a PUT request to the `server_usergroups/<groupname>` endpoint.

The example below updates the user group named Operators to have permissions to modify server settings, cause clients to be disconnected, and loading new runtime projects; it also updates the description of the group:

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/admin/server_usergroups/Operators
```

Body:

```
{
  "common.ALLTYPES_DESCRIPTION": "User group for standard operators",
  "libadminsettings.USERMANAGER_SERVER_MODIFY_SERVER_SETTINGS": true,
  "libadminsettings.USERMANAGER_SERVER_DISCONNECT_CLIENTS": true,
  "libadminsettings.USERMANAGER_SERVER_REPLACE_RUNTIME_PROJECT": true
}
```

- **Note:** Group permissions for the administrator group are locked open and can't be modified by any user to prevent an administrator from accidentally disabling a permission that could prevent administrators from modifying any users permissions. Only users in the administrators group can modify permissions for other groups.

## Configuration API Service — Configuring User Group Project Permissions

All user groups contain a collection of project permissions. Each project permission corresponds to a specific permission applied when interacting with objects in the project. All permissions are always present under a user group (and therefore cannot be created nor deleted). An individual project permission can be granted or denied by updating that specific project permission under the desired User Group.

- Only members of the Administrators group can update a user group's project permissions.
- There is no PROJECT\_ID field for project permissions.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a PUT request to the `project_permissions/<permission_name>` endpoint.

The example below updates the user-created user group named Operators to grant permission to users of that group to add, edit, and delete channels:

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/admin/server_usergroups/Operators/project_permissions/Servermain Channel
```

Body:

```
{
```



```

"libadminsettings.USERMANAGER_PROJECTMOD_ADD": true,
"libadminsettings.USERMANAGER_PROJECTMOD_EDIT": true,
"libadminsettings.USERMANAGER_PROJECTMOD_DELETE": true
}

```

## OPC UA Endpoint

While the majority of the OPC UA configuration is located under the Projects endpoint, the ua-endpoints are configured under the admin endpoint:

• **See Also:** [Project Properties — OPC UA](#)

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/admin/ua_endpoints
```

### Supported Actions

HTTP(S) Verb	Action
GET	Retrieves a list of all UA endpoint objects
POST	Creates a new UA endpoint

Endpoint:

```
https://<hostname_or_ip>:<port>/config/v1/admin/ua_endpoints/<endpointName>
```

### Supported Actions:

HTTP(S) Verb	Action
GET	Retrieves the specified UA endpoint
PUT	Updates the specified UA endpoint

### Properties

Name	Type	Required	Default	Description
common.ALLTYPES_NAME	String	Yes	NA	Specifies the identity of this object
common.ALLTYPES_DESCRIPTION	String	No	""	Lists available network adapters found on the system. Adapters without assigned IP address are listed as disconnected.
libadminsettings.UACONFIGMANAGER_ENDPOINT_ENABLE	Enable/Disable	No	True	Defines if the endpoint is enabled or disabled
libadminsettings.UACONFIGMANAGER_ENDPOINT_ADAPTER	String	No	"Default"	Specifies the network adapter to which the endpoint will be bound. A list of network adapters

Name	Type	Required	Default	Description
				<p>installed on the system is provided in the endpoint Description property.</p> <p>The "Default" adapter indicates that the endpoint can bind to any adapter.</p> <p><b>Note:</b> Network adapters that do not have a valid IPv4 address can be used for configuring a UA Endpoint; however, and endpoint is only used when there is a valid IPv4 address during startup. The server needs to be reinitialized for endpoint configurations to be refreshed after configuration changes are made to the host's network adapters.</p>
libadminsettings.UACONFIGMANAGER_ENDPOINT_PORT	Integer	No	49330	The port number to which the endpoint will be bound
libadminsettings.UACONFIGMANAGER_ENDPOINT_URL	String	No	""	<p>The endpoint URL (READONLY).</p> <p>The property value is generated based on the selected network adapter and port property.</p> <p><b>Note:</b> The property is blank when the specified network adapter does not have a valid IPv4 address.</p>
libadminsettings.UACONFIGMANAGER_ENDPOINT_SECURITY_NONE	Enable/Disable	No	False	<p>The accepted endpoint security policy:</p> <p><b>None:</b> Endpoint accepts insecure connections</p> <p><b>Note:</b> {Insecure}</p>

Name	Type	Required	Default	Description
				This setting is insecure and not recommended.
libadminsettings.UACONFIGMANAGER_ENDPOINT_SECURITY_BASIC256_SHA256	Enum	No	2	<p>The accepted endpoint security policy:</p> <p><b>BASIC256_SHA256:</b> Endpoint accepts BASIC256_SHA256 encrypted connections</p> <p>The value determines the supported message mode or disabled if no message mode is selected:</p> <p>Enum=Disabled:0 Enum=Sign:1 Enum=Sign and Encrypt:2 Enum=Sign; Sign and Encrypt:3</p>
libadminsettings.UACONFIGMANAGER_ENDPOINT_SECURITY_BASIC128_RSA15	Enum	No	0	<p>The accepted endpoint security policy:</p> <p><b>BASIC128_RSA15:</b> Endpoint accepts BASIC128_RSA encrypted connections.</p> <p>The value determines the supported message mode or disabled if no message mode is selected:</p> <p>Enum=Disabled:0 Enum=Sign:1 Enum=Sign and Encrypt:2 Enum=Sign; Sign and Encrypt:3</p> <p><b>Note:</b> {Deprecated} This security policy is deprecated.</p>
libadminsettings.UACONFIGMANAGER_ENDPOINT_SECURITY_BASIC256	Enum	No	0	<p>The accepted endpoint security policy:</p> <p><b>BASIC256:</b> Endpoint accepts BASIC256 encrypted connections</p> <p>The value determines the supported mes-</p>

Name	Type	Required	Default	Description
				sage mode or disabled if no message mode is selected: Enum=Disabled:0 Enum=Sign:1 Enum=Sign and Encrypt:2 Enum=Sign; Sign and Encrypt:3 <b>Note:</b> {Deprecated} This security policy is deprecated.

● **Note:** A maximum of 100 OPC UA endpoints may be configured on a single instance of ThingWorx Kepware Edge.

### Configuration API Service — Creating a UA Endpoint

To create a UA endpoint via the Configuration API service, only a minimum set of properties are required; all others are set to their default value.

To create a new UA endpoint, use a REST-based API tool such as Postman, Insomnia, or Curl and make a POST request to the admin/ua\_endpoints endpoint.

Endpoint (POST):

```
https://<hostname_or_ip>:<port>/config/v1/admin/ua_endpoints
```

Body:

```
{
  "common.ALLTYPES_NAME": "Endpoint1"
}
```

### Configuration API Service — Updating a UA Endpoint

To update a UA endpoint via the Configuration API service, provide new values for the properties that require updating.

Using a REST-based API tool such as Postman, Insomnia, or Curl; make a POST request to the ua\_endpoints/<endpoint> endpoint.

The example below updates the endpoint named Endpoint1 with a new port number and security policy:

Endpoint (PUT):

```
https://<hostname_or_ip>:<port>/config/v1/admin/ua_endpoints/Endpoint1
```

Body:

```
{
  "libadminsettings.UACONFIGMANAGER_ENDPOINT_PORT": 49321,
}
```

```

"libadminsettings.UACONFIGMANAGER_ENDPOINT_SECURITY_BASIC256": 1
}

```

## Configuration API Service — Removing a UA Endpoint

To delete an existing UA endpoint make a DELETE request to the `ua_endpoints/<endpoint_name>` endpoint. A request body is not required:

Endpoint (DELETE):

```
https://<hostname_or_ip>:<port>/config/v1/admin/ua_endpoints/Endpoint1
```

Body:

```

{
}

```

## Configuration API Service — Data

The Configuration API Service receives requests in standard JSON format from the REST client. These requests are consumed by the server and broken down into create, read, update, or delete commands.

• Please consult additional information on properties, data ranges, endpoint mapping scheme, and acceptable actions for each endpoint is available at the Configuration API Landing Page at [https://<hostname\\_or\\_ip>:<port>/config/](https://<hostname_or_ip>:<port>/config/) (for default configurations).

• Documentation served from the landing page is HTML-encoded by default. To obtain JSON-encoded documentation, include an "Accept" request header with "application/json".

• Object names containing spaces, or other characters disallowed in URL formatting, must be percent-encoded to be correctly interpreted by the Configuration API. Percent encoding involves replacing disallowed characters with their hexadecimal representation. For example, an object named 'default object' is percent-encoded as `default%20object`. The following characters are not permitted in a URL and must be encoded:

*space*	!	#	\$	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%20	%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

• All leading and trailing spaces are removed from object names before the server validates them. This can create a discrepancy between the object name in the server and the object name a user provides via the Configuration API. Users can send a GET on the parent object after sending a PUT/POST to verify the new or modified object name in the server matches what was sent via the API.

• An attempt to perform a POST/PUT/DELETE with the API as a non-admin user fails if a user has the server configuration open at the same time. The error is a 401 status code (unauthorized). Only one user can write to the runtime at a time; the API cannot take permissions from the server configuration if it has insufficient credentials.

## Create an Object

An object can be created by sending an HTTPS POST request to the Configuration API. When creating a new object, the JSON must include required properties for the object (ex. each object must have a name), but doesn't require all properties. All properties not included in the JSON are set to the default value on creation.

Example POST JSON body:

```
{
  "<Property1_Name>": <Value>,
  "<Property2_Name>": <Value>,
  "<Property3_Name>": <Value>
}
```

## Create Multiple Objects

Multiple objects may be added to a given collection by including the JSON property objects in an array.

Example POST JSON body:

```
[
  {
    "<Property1_Name>": <Value>,
    "<Property2_Name>": <Value>,
    "<Property3_Name>": <Value>
  },
  {
    "<Property1_Name>": <Value>,
    "<Property2_Name>": <Value>,
    "<Property3_Name>": <Value>
  }
]
```

When a POST includes multiple objects, if one or more cannot be processed due to a property validation failure or some other error, the HTTPS status code 207 (Multi-Status) will be returned along with a JSON object array containing the status for each object in the request.

For example, if two objects are included in the request and the second one specifies the same name as the first:

```
[
  {
    "code": 201,
    "message": "Created"
  },
  {
    "code": 400,
    "message": "Validation failed on property common.ALLTYPES_NAME in object definition at line 7: The name 'Channell' is already used."
  }
]
```

## Create an Object with Child Hierarchy

An object may be created with a full child object hierarchy beneath it. To do this, include that hierarchy in the POST request just as it would appear when saved in a JSON project file.

For example, to create a channel with a device underneath it, the following JSON could be used:

```
[
  {
    "common.ALLTYPES_NAME": "Channel1",
    "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator",
    "devices":
      [
        {
          "common.ALLTYPES_NAME": "Device1",
          "servermain.MULTIPLE_TYPES_DEVICE_DRIVER": "Simulator",
          "servermain.DEVICE_MODEL": 0
        }
      ]
  }
]
```

## Read an Object

An object can be read by sending an HTTPS GET request to the Configuration API. All object properties are returned on every GET request and each object includes a Project\_ID. The Project\_ID property is used to track changes in the configuration and is updated on any change from the Configuration API or a server configuration client. This property should be saved and used in all PUT requests to prevent stale data manipulations.

Example response body:

```
{
  "<Property1_Name>": <Value>,
  "<Property2_Name>": <Value>,
  "PROJECT_ID": 12345678
}
```

• See Also: [Content Retrieval](#)

## Edit an Object

An object can be edited by sending an HTTPS PUT request to the Configuration API. PUT requests require the Project\_ID or Force\_Update property in the JSON body. Setting Force\_Update to True ignores Project\_ID validation.

Example PUT body:

```
{
  "<Property1_Name>": <Value>,
  "<Property2_Name>": <Value>,
  "PROJECT_ID": 12345678,
  "FORCE_UPDATE": true
}
```

Normally when a PUT request succeeds and all properties are assigned successfully, there is no response body returned to the client, there is simply a 200 status code to indicate success. There can be cases where a property is included in a PUT request that is not assigned to the object instance by the Server Runtime. In these cases, a response body will be generated as follows:

Body:

```
{,
  "not_applied":,
```

```
{,
  "servermain.CHANNEL_UNIQUE_ID": 2466304381
},
"code": 200,
"message": "Not all properties were applied. This could be due to active client
reference or property is disallowed/disabled/read-only."
}
```

The response indicates which property or properties were not applied to the object instance where each contains the value that is actually in use. There are several possible reasons why the property value could not be applied, such as:

- The property is read-only and cannot be changed.
- There is a client reference on the object that restricts what properties can be updated.
- The property is not allowed based on the values of other properties on which this condition depends.
- The property is not enabled based on the values of other properties on which this condition depends.
- The value was transformed in some way (ex. rounded or truncated).

## Delete an Object

An object can be deleted by sending an HTTPS DELETE request to the Configuration API. The Configuration API does not allow deleting multiple items on the same level with a single request (such as deleting all of the devices in a channel), but can delete an entire tree (such as deleting a device deletes all its child tags).

## Errors

All Configuration API Service requests return errors in JSON format.

Example:

```
{
  "code": 400,
  "message": "Invalid property: 'NAME'."
}
```

• See Also: [Troubleshooting](#)

## Configuration API Service — Content Retrieval

Content is retrieved from the server by issuing an HTTP(S) GET request. The URI specified in the request can target one of the following areas:

1. Online documentation (ex. [https://<hostname\\_or\\_ip>:<port>/config/v1/doc](https://<hostname_or_ip>:<port>/config/v1/doc) or [/config/v1/doc/drivers](https://<hostname_or_ip>:<port>/config/v1/doc/drivers))
2. Event log entries (ex. [https://<hostname\\_or\\_ip>:<port>/config/v1/event\\_log](https://<hostname_or_ip>:<port>/config/v1/event_log))
3. Transaction log entries (ex. [https://<hostname\\_or\\_ip>:<port>/config/v1/transaction\\_log](https://<hostname_or_ip>:<port>/config/v1/transaction_log))
4. Project configuration (ex. [https://<hostname\\_or\\_ip>:<port>/config/v1/project](https://<hostname_or_ip>:<port>/config/v1/project) or [/config/v1/project/channels/Channel1](https://<hostname_or_ip>:<port>/config/v1/project/channels/Channel1))



When targeting project configuration, a REST client can specify the type(s) of content that should be returned. In this context the word “content” refers to a category or categories of data about a collection or object instance.

By default, when a GET request is issued using an endpoint that identifies a collection, the server will return a JSON array that contains one value for each instance in the collection where each value is a JSON object that contains the properties of the instance.

By default, when a GET request is made using an endpoint that identifies an object instance, the server will return a JSON object that contains the properties of that instance.

The default behavior of these requests can be altered by specifying one or more “content” query parameters appended to the URL as in `https://<hostname>:<port>/config/v1/project?content=children`. The following table shows the available content types and their applicability to each endpoint type:

Content Type	Collection Endpoint	Object Instance Endpoint
properties	yes	yes
property_definitions	no	yes
property_states	no	yes
type_definition	yes	yes
children	yes	yes

The following table shows the structure of the JSON response for a given content type:

GET Request URI	JSON Response Structure
<code>https://&lt;hostname_or_ip&gt;:&lt;port&gt;/config/v1/project?content=properties</code>	<pre>{   &lt;property name&gt;: &lt;value&gt;,   &lt;property name&gt;: &lt;value&gt;,   ... }</pre>
<code>https://&lt;hostname_or_ip&gt;:&lt;port&gt;/config/v1/project?content=property_definitions</code>	<pre>[   {&lt;property definition&gt;},   {&lt;property definition&gt;},   ... ]</pre>
<code>https://&lt;hostname_or_ip&gt;:&lt;port&gt;/config/v1/project?content=property_states</code>	<pre>{   "allow":   {     &lt;property name&gt;: true/-     false,     &lt;property name&gt;: true/-     false,     ...   },   "enable":   {     &lt;property name&gt;: true/-     false,     &lt;property name&gt;: true/-     false,     ...   } }</pre>

GET Request URI	JSON Response Structure
<pre>https://&lt;hostname_or_ip&gt;:&lt;- port&gt;/config/v1/project?content=type_definition</pre>	<pre>{   "name": &lt;type name&gt;,   "collection": &lt;collection name&gt;,   "namespace": &lt;namespace name&gt;,   "can_create": true/false,   "can_delete": true/false,   "can_modify": true/false,   "auto_generated": true/- false,   "requires_driver": true/- false,   "access_controlled": true/- false,   "child_collections": [&lt;col- lection names&gt;] }</pre>
<pre>https://&lt;hostname_or_ip&gt;:&lt;- port&gt;/config/v1/project?content=children</pre>	<pre>{   &lt;collection name&gt;: [     {       "name": &lt;object instance name&gt;,       "href": &lt;object instance uri&gt;     },     ...   ],   &lt;collection name&gt;: [     {       "name": &lt;object instance name&gt;,       "href": &lt;object instance uri&gt;     },     ...   ],   ... }</pre>

Multiple content types can be specified in the same request by separating with a comma. For example, `https://<hostname>:<port>/config/v1/project?content=children,type_definition`. When multiple types are specified, the JSON response will contain a single object with a member for each requested content type as in:

```
{
  "properties": <properties response structure>,
  "property_definitions": <property definitions response structure>,
  "property_states": <property states response structure>,
  "type_definition": <type definition response structure>,
  "children": <children response structure>
}
```

### Type Definitions

The following table describes the members of the type definition JSON object.

Member	Type	Description
name	string	Object type name.
collection	string	Collection name. Identifies the collection in which objects of this type will exist. This name constitutes a valid endpoint that can be addressed using the REST interface.
namespace	string	Namespace that implements the object type. Objects that are implemented by the server exist in the "servermain" namespace. Other namespaces are defined by optional components such as drivers, plug-ins and client interfaces.
can_create	bool	Indicates whether or not instances of this type can be created by an end user. For example, this is false for the "Project" type because it's not something that can be created.
can_delete	bool	Indicates whether or not instances of this type can be deleted by an end user. Again, the "Project" type is not something that can be deleted.
can_modify	bool	Indicates whether or not instances of this type can be modified by an end user. For example, the server has some auto-generated objects that exist to create a child collection only and do not themselves have any modifiable properties.
auto_generated	bool	If true, instances of this type are auto-generated by the server. Typically objects of this type will have the previous three members defined as "false".
requires_driver	bool	True if instances of this type cannot be created without supplying the name of an installed driver.
access_controlled	bool	True if the server provides group-level access control over the CRUD operations that can be executed against an instance of this type (see <a href="#">User Manager</a> ).
child_collections	array	An array of collection names that are supported as children under an object of this type. For example, if a type includes "devices" in "child_collections", then object instances of that type will support one or more "Device" instance as a child.

## Property Definitions

A property definition identifies the characteristics of a given property, including the type of data it supports, applicable ranges, default value, etc. The JSON structure of a property definition object is defined as follows:

Member	Type	Description
symbolic_name	string	Identifies the property by canonical name in the form <namespace>.<-property name>.
display_name	localized string	The name the property would have if shown in the Server Configuration property editor. Value will be returned in the language the server is currently configured to use.
display_description	localized string	The description the property would have if shown in the Server Configuration property editor. Value will be returned in the language the server is currently configured to use.
read_only	Boolean	True if the property is informational, not expected to change once initially defined.
type	string	Determines the data type of the property value (see "Property Types" below).
minimum_value	number or null (applies to numeric types)	Minimum value the property can have to be considered valid. If null, there is no minimum.

Member	Type	Description
maximum_value	number or null (applies to numeric types)	Maximum value the property can have to be considered valid. If null, there is no maximum.
minimum_length	number (applies to strings only)	Minimum length a string value may have. 0 means no minimum.
maximum_length	number (applies to strings only)	Maximum length a string value may have. -1 means no maximum.
hints	arrays of strings (applies to strings only)	An array of possible choices that may be assigned to the property value. This member not included if no hints exist.
enumeration	object (applies to enumerations only)	For enumeration properties, this object identifies the valid name / value pairs the enumeration can have. Structure is as follows: <pre>{   &lt;name&gt;: number,   &lt;name&gt;: number,   ... }</pre>
allow	array of objects	Defines a conditional dependency on one or more other properties that determines whether this property is relevant. Properties that are not allowed are not shown in the Server Configuration property editor (see "Allow and Enable Conditions" below).
enable	array of objects	Defines a conditional dependency on one or more other properties that determines whether this property should be enabled for the client to change. Properties that are not enabled are grayed out in the Server Config property editor (see "Allow and Enable Conditions" below).

To get specific information about the property definitions of a specific endpoint, add "?content=property\_definitions" to the end of the URL of a GET request.

For example, to get the property definitions for a channel named Channel1 with the server running on the local host, the GET request would be sent to:

Endpoint:

```
https://<hostname_or_ip>:<port>/config/v1/channels/Channel1?content=property_definitions
```

The returned JSON block would look something like the following:

```
[
  {
    "symbolic_name": "common.ALLTYPES_NAME",
    "display_name": "Name",
    "display_description": "Specify the identity of this object.",
    "read_only": false,
    "type": "String",
    "default_value": null,
  }
]
```

```

    "minimum_length": 1,
    "maximum_length": 256
  },
  {
    "symbolic_name": "common.ALLTYPES_DESCRIPTION",
    "display_name": "Description",
    "display_description": "Provide a brief summary of this object or its use.",
    "read_only": false,
    "type": "String",
    "default_value": null,
    "minimum_length": 0,
    "maximum_length": 255
  },
  ...

```

## Property Types

The following table describes the different values that a property definition may contain for the “type” member. The “Value Type” identifies what JSON type the property value should have.

Type Name	Value Type	Description
AllowDeny	bool	Describes a property that would show a drop-down list that contains the choices “Allow”=true and “Deny”= false.
EnableDisable	bool	Describes a property that would show a drop-down list that contains the choices “Enable”=true and “Disable”= false.
YesNo	bool	Describes a property that would show a drop-down list that contains the choices “Yes”=true and “No”= false.
String	string	Generic string. Properties of this type will include minimum_length and maximum_length specifiers.
StringArray	array	Array of strings. Properties of this type will include minimum_length and maximum_length specifiers that apply to the strings themselves, not the length of the array.
Password	string	Obfuscated string that contains a password. When changing the value of a property of this type, a plain-text password is expected. Password values should only be changed over a secure connection. Passwords must be at least 14 characters.
LocalFileSpec	string	A fully qualified file specification in the local file system.
UncFileSpec	string	A fully qualified file specification in a network location.
LocalPathSpec	string	A fully qualified path specification in the local file system.
UncPathSpec	string	A fully qualified path specification to a network location.
StringWithBrowser	string	Describes a property that has a string value that would normally be chosen from a collection of dynamically generated strings.
Integer	number	Unsigned 32-bit integer value.
Hex	number	Unsigned 32-bit integer value intended to be displayed / edited in hexadecimal notation.
Octal	number	Unsigned 32-bit integer value intended to be displayed / edited in octal notation.
SignedInteger	number	Signed 32-bit integer value.

Type Name	Value Type	Description
Real4	number	Single precision floating point value.
Real8	number	Double precision floating point value.
Enumeration	number	One of the possible numeric values from the "enumeration" member of the property definition.
PropArray	object	Describes a structure containing members that each have a fixed-length array of values.
TimeOfDay	number	Integer value containing the number seconds since midnight that would define a specific time of day.
Date	number	Unix time value that specifies midnight on a given date.
DateAndTime	number	Unix time value that specifies a specific time on a given date.
Blob	array	Array of byte values that represents an opaque collection of data. Data of this type originates in the server and is hashed to prevent modification.

### Allow and Enable Conditions

For definitions that contain allow and/or enable conditions, this is the structure they would have in the JSON:

```
<condition>: [
  {
    "depends_on": <property name>
    "operation": "==" or "!="
    "value": <value>
  },
  ...
]
```

Each condition identifies another property that is a dependent and how it depends as equal or not equal to the value of that property. More than one dependency can exist, either on the same property or different ones. If multiple exist, the "operation" will always be the same. Evaluation of the expression to determine the state of the condition when multiple dependencies exist is a logical "or" for "==" and a logical "and" for "!=".

When using "content=property\_states", the returned JSON describes the outcome of the evaluation of these conditions (if they exist) for each property.

### Configuration API Service — Concurrent Clients

The Configuration API can serve multiple REST clients at the same time. To prevent a client from editing stale configurations, the Server Runtime maintains a numeric project ID. Each time an object is edited through the Configuration API or the local Configuration client, the Project ID changes. The current project ID is returned in each GET response. The current project ID must be specified by the client in all PUT requests.

The best practice is to issue a GET request, save the current project ID, and use that ID for the following PUT request. If only one client is used, the client may put the property "FORCE\_UPDATE": true in the PUT request body to force the Configuration API server to ignore the project ID.

### Using UaExpert

An application like Unified Automation UaExpert can be used to verify the flow of data from devices through ThingWorx Kepware Edge.

● The UaExpert tool is designed to be a general-purpose OPC UA test client; it is not meant for production. Below is a walkthrough of creating a secure user with specific data access rights to read and write tags.

## Default OPC UA Server Settings

- URL: opc.tcp://<hostname>:<port>
- Port: 49330
- Security Policies: Basic256Sha256
- Authentication: (Enabled by default)
- Server Interface Enabled: True

## Creating a User Group and User with Read / Write / Browse Access

1. Install ThingWorx Kepware Edge with default settings.
2. Add a new user group with data access and browse permissions via the Config API:

Endpoint (POST):

```
https://<hostname>:<port>/config/v1/admin/server_usergroups
```

Body:

```
{
  "common.ALLTYPES_NAME": "Group1",
  "libadminsettings.USERMANAGER_GROUP_ENABLED": true,
  "libadminsettings.USERMANAGER_IO_TAG_READ": true,
  "libadminsettings.USERMANAGER_IO_TAG_WRITE": true,
  "libadminsettings.USERMANAGER_BROWSE_BROWSENAMESPACE": true
}
```

3. Add a new user with a password to the group created in above.

Endpoint (POST):

```
https://<hostname>:<port>/config/v1/admin/server_users
```

Body:

```
{
  "common.ALLTYPES_NAME": "User1",
  "libadminsettings.USERMANAGER_USER_GROUPNAME": "Group1",
  "libadminsettings.USERMANAGER_USER_ENABLED": true,
  "libadminsettings.USERMANAGER_USER_PASSWORD": "<insert_password>"
}
```

## Adding Server Connection to UaExpert

1. Download, install, and launch UaExpert from Unified Automation.
2. Select the **Server | Add** drop-down menu option.

3. In the **Add Server** configuration window, double-click the **Add Server** option located under **Custom Discovery**.
4. Enter the URL and port for the machine to connect. For example: "opc.tcp://<hostname>:49330".
5. A new server connection is added in the Custom Discovery group.
6. Expand the new server connection for a list of valid endpoints. These are the available security options for the server. In this example, only one option is available.
7. Choose the **Basic256Sha256 – Sign & Encrypt** security option.
8. Set the user name and password using the settings used in the creation of the user above.
9. Check the **Store** checkbox to save the password or leave it unchecked and to be prompted for a password when connecting to the server.
10. Click **OK** to close the window.
11. Verify that "ThingWorxKepwareEdge/UA" appears under Servers.
12. Right-click on the server and select **Connect**.
13. A certificate validation window appears.
14. Click **Trust Server Certificate** for the client to trust the ThingWorxKepwareEdge/UA server.
15. Click **Continue**. There is an error until the server trusts the client certificate.
16. To trust the client certificate on the server, use the [edge\\_admin](#) tool.
17. The client certificate's thumbprint is required to trust it. To get the thumbprint, use the edge\_admin tool to list the certificates in the UA Server trust store:

```
$ <installation_directory>/edge_admin manage-truststore --list uaserver
```

18. The output of the list shows a thumbprint, a status, and a common name of the certificate.  
🔴 The UAExpert certificate will be Rejected. Use the thumbprint to trust the certificate.

```
$ <installation_directory>/edge_admin manage-truststore --trust-  
t=<certificate_thumbprint> uaserver
```

19. List the certificates of the UA Server to verify that the certificate is now trusted.
20. In UaExpert, right-click on the server and click **Connect**. The connection should succeed and the Address Space window in the lower right pane should be populated, which enables browsing for and adding tags.
21. Add a tag in the data access view to verify that the user has read access.
22. Change the value of the tag to verify that the user has write access.



# Event Log Messages

The following information concerns messages posted to the Event Log. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

## Configuration API Service — Response Codes

One of the following response codes may be returned from a REST request. Where possible, the body of the response contains specific error messages to help identify the cause of the error and possible solutions:

- HTTPS/1.1 200 OK
- HTTPS/1.1 201 Created
- HTTPS/1.1 202 Accepted
- HTTPS/1.1 207 Multi-Status
- HTTPS/1.1 400 Bad Request
- HTTPS/1.1 401 Unauthorized
- HTTPS/1.1 403 Forbidden
- HTTPS/1.1 404 Not Found
- HTTPS/1.1 429 Too Many Requests
- HTTPS/1.1 500 Internal Server Error
- HTTPS/1.1 503 Server Runtime Unavailable
- HTTPS/1.1 504 Gateway Timeout
- HTTPS/1.1 520 Unknown Error

Consult the [Configuration API Service Event Log Messages](#)

## Configuration API Service — Logging

Messages from the event log service can be read from a REST client by sending a GET to `https://<host-name>:<port>/config/v1/event_log`. The response contains comma-separated entries.

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/event_log
```

Example Return:

```
[
  {
    "timestamp": "2018-11-13T16:34:57.966",
    "event": "Security",
    "source": "ThingWorxKepwareEdge\\Runtime",
    "message": "Configuration session started by admin as Default User (R/W)."
  },
  {
    "timestamp": "2018-11-13T16:35:08.729",
    "event": "Warning",
    "source": "Licensing",
    "message": "Feature Modbus TCP/IP Ethernet is time limited and will expire at 11/13/2019 12:00 AM."
  }
]
```

```
}
...
]
```

**Filtering:** The Configuration API event log endpoint allows log items to be sorted or limited using filter parameters specified in the URI. The filters, which can be combined or used individually, allow the results of the log query to be restricted to a specific time period (e.g. events which occurred since a given date, events which occurred before a given date, or events that occurred between two dates). Example filtered log query:

Endpoint (GET):

```
https://<hostname_or_ip>:<port>/config/v1/event_log?limit=10&start=2016-01-01T00:00:00.000&end=2016-01-02T20:00:00.000
```

where:

- **Limit** = Maximum number of log entries to return. The default setting is 100 entries.
- **Start** = Earliest time to be returned in YYYY-MM-DDTHH:mm:ss.sss (UTC) format.
- **End** = Latest time to be returned in YYYY-MM-DDTHH:mm:ss.sss (UTC) format.

● **Note:** The Limit filter overrides the result of the specified time period. If there are more log entries in the time period than the Limit filter allows, only the newest specified quantity of records that match the filter criteria are displayed.

---

### The Config API SSL certificate contains a bad signature.

#### Error Type:

Error

---

### The Config API is unable to load the SSL certificate.

#### Error Type:

Error

---

### Unable to start the Config API Service. Possible problem binding to port.

#### Error Type:

Error

---

### The Config API SSL certificate has expired.

#### Error Type:

Warning

---

### The Config API SSL certificate is self-signed.

#### Error Type:

Warning

---

### The <name> device driver was not found or could not be loaded.

#### Error Type:

Error

**Possible Cause:**

1. If the project has been moved from one PC to another, the required drivers may have not been installed yet.
2. The specified driver may have been removed from the installed server.
3. The specified driver may be the wrong version for the installed server version.

**Possible Solution:**

1. Re-run the server install and add the required drivers.
2. Re-run the server install and re-install the specified drivers.
3. Ensure that a driver has not been placed in the installed server directory (which is out of sync with the server version).

**Unable to load the '<name>' driver because more than one copy exists ('<name>' and '<name>'). Remove the conflicting driver and restart the application.**

---

**Error Type:**

Error

**Possible Cause:**

Multiple versions of the driver DLL exist in the driver's folder in the server.

**Possible Solution:**

1. Re-run the server install and re-install the specified drivers.
2. Contact Technical support and verify the correct version. Remove the driver that is invalid and restart the server and load the project.

**Invalid project file.**

---

**Error Type:**

Error

**Unable to add channel due to driver-level failure.**

---

**Error Type:**

Error

**Possible Cause:**

Attempt failed due to issues in the driver.

**Possible Solution:**

Refer to the additional messages about the driver error and correct related issues.

---

**Unable to add device due to driver-level failure.**

---

**Error Type:**

Error

**Possible Cause:**

Attempt failed due to issues in the driver.

**Possible Solution:**

Refer to the additional messages about the driver error and correct related issues.

---

**Version mismatch.**

---

**Error Type:**

Error

---

**Invalid XML document:**

---

**Error Type:**

Error

**Possible Cause:**

The server is unable to parse the specified XML file.

**Possible Solution:**

If the server project was edited using a third-party XML editor, verify that the format is correct via the schemas for the server and drivers.

---

**Unable to load project <name>:**

---

**Error Type:**

Error

**Possible Cause:**

The project was created in a server version that is not compatible with the version trying to load it.

**Possible Solution:**

Typically this happens when a project was created in a newer version of the server and it is being opened in an older version.

**Note:**

Every attempt is made to ensure backwards compatibility in the server so that projects created in older versions may be loaded in newer versions. However, since new versions of the server and driver may have properties and configurations that do not exist in older versions, it may not be possible to open or load an older project in a newer version.

**Unable to backup project file to '<path>' [<reason>]. The save operation has been aborted. Verify the destination file is not locked and has read/write access. To continue to save this project without a backup, deselect the backup option under Tools | Options | General and re-save the project.**

---

**Error Type:**

Error

**Possible Cause:**

1. The destination file may be not locked by another application.
2. The destination file or the folder where it is located does not allow read/write access.

**Possible Solution:**

1. Ensure that the destination file is not locked by another application, unlock the file, or close the application.
2. Ensure that the destination file and with the folder where it is located allow read and write access.

**<feature name> was not found or could not be loaded.**

---

**Error Type:**

Error

**Possible Cause:**

The feature is not installed or is not in the expected location.

**Possible Solution:**

Re-run the server install and select the specified feature for installation.

**Unable to save project file <name>:**

---

**Error Type:**

Error

**Device discovery has exceeded <count> maximum allowed devices. Limit the discovery range and try again.**

---

**Error Type:**

Error

**<feature name> is required to load this project.**

---

**Error Type:**

Error

---

**Unable to load the project due to a missing object. | Object = '<object>'.**

---

**Error Type:**

Error

**Possible Cause:**

Editing the JSON project file may have left it in an invalid state.

**Possible Solution:**

Revert any changes made to the JSON project file.

---

**Invalid Model encountered while trying to load the project. | Device = '<device>'.**

---

**Error Type:**

Error

**Possible Cause:**

The specified device has a model that is not supported in this version of the server.

**Possible Solution:**

Open this project with a newer version of the server.

---

**Cannot add device. A duplicate device may already exist in this channel.**

---

**Error Type:**

Error

---

**Auto-generated tag '<tag>' already exists and will not be overwritten.**

---

**Error Type:**

Warning

**Possible Cause:**

Although the server is regenerating tags for the tag database, it has been set not to overwrite tags that already exist.

**Possible Solution:**

If this is not the desired action, change the setting of the "On Duplicate Tag" property for the device.

---

**Unable to generate a tag database for device '<device>'. The device is not responding.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The device did not respond to the communications request.
2. The specified device is not on, not connected, or in error.

**Possible Solution:**

1. Verify that the device is powered on and that the PC is on (so that the server can connect to it).
2. Verify that all cabling is correct.
3. Verify that the device IDs are correct.
4. Correct the device failure and retry the tag generation.

**Unable to generate a tag database for device '<device>':**

---

**Error Type:**

Warning

**Possible Cause:**

The specified device is not on, not connected, or in error.

**Possible Solution:**

Correct the device failure and retry the tag generation.

**Auto generation produced too many overwrites, stopped posting error messages.**

---

**Error Type:**

Warning

**Possible Cause:**

1. To keep from filling the error log, the server has stopped posting error messages on tags that cannot be overwritten during automatic tag generation.
2. Reduce the scope of the automatic tag generation or eliminate problematic tags.

**Failed to add tag '<tag>' because the address is too long. The maximum address length is <number>.**

---

**Error Type:**

Warning

**Line '<line>' is already in use.**

---

**Error Type:**

Warning

**Possible Cause:**

The target modem line is already open, likely because it is in use by another application.

**Possible Solution:**

Find the application holding the modem open and close or release it.

---

**Hardware error on line '<line>'.**

---

**Error Type:**

Warning

**Possible Cause:**

A hardware error was returned after a request was made for a tag in a device connected to the modem.

**Possible Solution:**

Disable data collection on the device. Enable it after the modem connects to the destination modem.

**Note:**

The error occurs on first scan and is not repeated.

---

**No comm handle provided on connect for line '<line>'.**

---

**Error Type:**

Warning

**Possible Cause:**

An attempt was made to connect to the modem line with no specified COMM handle.

**Possible Solution:**

Verify the modem is installed and initialized correctly.

---

**Unable to use network adapter '<adapter>' on channel '<name>'. Using default network adapter.**

---

**Error Type:**

Warning

**Possible Cause:**

The network adapter specified in the project does not exist on this PC. The server uses the default network adapter.

**Possible Solution:**

Select the network adapter to use for the PC and save the project.

**See Also:**

Channel Properties - Network Interface

---

**Rejecting attempt to change model type on a referenced device '<channel device>'.**

---

**Error Type:**

Warning



---

**Validation error on '<tag>': <error>.**

---

**Error Type:**

Warning

**Possible Cause:**

An attempt was made to set invalid parameters on the specified tag.

---

**Validation error on '<tag>': Invalid scaling parameters.**

---

**Error Type:**

Warning

**Possible Cause:**

An attempt was made to set invalid scaling parameters on the specified tag.

**See Also:**

Tag Properties - Scaling

---

**<Source>: Invalid Ethernet encapsulation IP '<address>'.**

---

**Error Type:**

Warning

**Possible Cause:**

The IP address specified for a device on an Ethernet encapsulated channel is not a valid IP address.

**Possible Solution:**

Correct the IP in the XML file and re-load the project.

**Note:**

This error can occur when loading XML formatted projects that were created or edited with third-party XML software.

---

**The '<product>' driver does not currently support XML persistence. Save using the default file format.**

---

**Error Type:**

Warning

**Possible Cause:**

The specified driver does not support XML formatting.

**Possible Solution:**

Save the project in .opf format.

---

**The time zone set for '<device>' is '<zone>'. This is not a valid time zone for the system. Defaulting the time zone to '<zone>'.**

---

**Error Type:**

Warning

---

**The specified network adapter is invalid on channel '%1' | Adapter = '%2'.**

---

**Error Type:**

Warning

**Possible Cause:**

The network adapter specified in the project does not exist on this PC.

**Possible Solution:**

Select the network adapter to use for the PC and save the project.

**See Also:**

Channel Properties - Network Interface

---

**No tags were created by the tag generation request. See the event log for more information.**

---

**Error Type:**

Warning

**Possible Cause:**

The driver produced no tag information but declined to provide a reason why.

**Possible Solution:**

Event log may contain information that will help troubleshoot the issue.

---

**TAPI configuration has changed, reinitializing...**

---

**Error Type:**

Informational

---

**<Product> device driver loaded successfully.**

---

**Error Type:**

Informational

---

**Starting <name> device driver.**

---

**Error Type:**

Informational

---

**Stopping <name> device driver.**

---

**Error Type:**

Informational

---

**Attempting to automatically generate tags for device '<device>'.**

---

**Error Type:**

Informational

**Completed automatic tag generation for device '<device>'.**

---

**Error Type:**

Informational

**Data collection is enabled on device '<device>'.**

---

**Error Type:**

Informational

**Data collection is disabled on device '<device>'.**

---

**Error Type:**

Informational

**Object type '<name>' not allowed in project.**

---

**Error Type:**

Informational

**Created backup of project '<name>' to '<path>'.**

---

**Error Type:**

Informational

**Device '<device>' has been auto-promoted to determine if communications can be re-established.**

---

**Error Type:**

Informational

**Failed to load library: <name>.**

---

**Error Type:**

Informational

**Failed to read build manifest resource: <name>.**

---

**Error Type:**

Informational

**The project file was created with a more recent version of this software.**

---

**Error Type:**

Informational

---

**A client application has disabled auto-demotion on device '<device>'.**

**Error Type:**

Informational

---

**Access to object denied. | User = '<account>', Object = '<object path>', Permission =**

**Error Type:**

Security

---

**Changing runtime operating mode.**

**Error Type:**

Informational

---

**Runtime operating mode change completed.**

**Error Type:**

Informational

---

**Shutting down to perform an installation.**

**Error Type:**

Informational

**Error Type:**

Security

---

**User moved from user group. | User = '<name>', Old group = '<name>', New group '<name>'.**

**Error Type:**

Security

---

**User group has been created. | Group = '<name>'.**

**Error Type:**

Security

---

**User added to user group. | User = '<name>', Group = '<name>'.**

**Error Type:**

Security

---

**User information replaced by import. | File imported = '<absolute file path>'.**

**Error Type:**

Security

---

---

**User group has been renamed. | Old name = '<name>', New name = '<name>'.**

---

**Error Type:**

Security

---

**Permissions definition has changed on user group. | Group = '<name>'.**

---

**Error Type:**

Security

---

**User has been renamed. | Old name = '<name>', New name = '<name>'.**

---

**Error Type:**

Security

---

**User has been disabled. | User = '<name>'.**

---

**Error Type:**

Security

---

**User group has been disabled. | Group = '<name>'.**

---

**Error Type:**

Security

---

**User has been enabled. | User = '<name>'.**

---

**Error Type:**

Security

---

**User group has been enabled. | Group = '<name>'.**

---

**Error Type:**

Security

---

**Failed to reset password for administrator. | Administrator name = '<name>'.**

---

**Error Type:**

Security

---

**Password for user has been changed. | User = '<name>'.**

---

**Error Type:**

Security

---

**Attempt to add item '<name>' failed.**

---

**Error Type:**

Error

---

---

**No device driver DLLs were loaded.**

---

**Error Type:**

Error

---

**Invalid project file: '<name>'.**

---

**Error Type:**

Error

---

**Could not open project file: '<name>'.**

---

**Error Type:**

Error

---

**Rejecting request to replace the project because it's the same as the one in use: '<name>'.**

---

**Error Type:**

Error

---

**Filename must not overwrite an existing file: '<name>'.**

---

**Error Type:**

Error

---

**Filename must not be empty.**

---

**Error Type:**

Error

---

**Filename is expected to be of the form <subdir>/<name>.{json,opf}**

---

**Error Type:**

Error

---

**Filename contains one or more invalid characters.**

---

**Error Type:**

Error

---

**Addition of object to '<name>' failed: <reason>.**

---

**Error Type:**

Warning

---

**Move object '<name>' failed: <reason>.**

---

**Error Type:**

Warning

---

**Update of object '<name>' failed: <reason>.**

**Error Type:**

Warning

---

**Delete object '<name>' failed: <reason>.**

**Error Type:**

Warning

---

**Unable to load startup project '<name>': <reason>.**

**Error Type:**

Warning

---

**Failed to update startup project '<name>': <reason>.**

**Error Type:**

Warning

---

**Runtime project replaced with startup project defined. Runtime project will be restored from '<name>' at next restart.**

**Error Type:**

Warning

---

**Ignoring user-defined startup project because a configuration session is active.**

**Error Type:**

Warning

---

**Write request rejected on read-only item reference '<name>'.**

**Error Type:**

Warning

---

**Unable to write to item '<name>'.**

**Error Type:**

Warning

---

**Write request failed on item '<name>'. The write data type '<type>' cannot be converted to the tag data type '<type>'.**

**Error Type:**

Warning

---

**Write request failed on item '<name>'. Error scaling the write data.**

**Error Type:**

Warning

---

**Write request rejected on item reference '<name>' since the device it belongs to is disabled.**

**Error Type:**

Warning

---

**<Name> successfully configured to run as a system service.**

**Error Type:**

Informational

---

**<Name> successfully removed from the service control manager database.**

**Error Type:**

Informational

---

**Runtime re-initialization started.**

**Error Type:**

Informational

---

**Runtime re-initialization completed.**

**Error Type:**

Informational

---

**Updated startup project '<name>'.**

**Error Type:**

Informational

---

**Runtime service started.**

**Error Type:**

Informational

---

**Runtime process started.**

**Error Type:**

Informational

---

**Runtime performing exit processing.**

**Error Type:**

Informational

---

**Runtime shutdown complete.**

**Error Type:**

Informational



---

**Shutting down to perform an installation.**

---

**Error Type:**

Informational

---

**Runtime project replaced from '<name>'.**

---

**Error Type:**

Informational

---

**Missing application data directory.**

---

**Error Type:**

Informational

---

**Runtime project saved as '<name>'.**

---

**Error Type:**

Informational

---

**Runtime project replaced.**

---

**Error Type:**

Informational

---

**Configuration session started by <name> (<name>).**

---

**Error Type:**

Security

---

**Configuration session assigned to <name> has ended.**

---

**Error Type:**

Security

---

**Configuration session assigned to <name> promoted to write access.**

---

**Error Type:**

Security

---

**Configuration session assigned to <name> demoted to read only.**

---

**Error Type:**

Security

---

**Permissions change applied on configuration session assigned to <name>.**

---

**Error Type:**

Security

---

**Failed to start Script Engine server. Socket error occurred binding to local port. | Error = <error>, Details = '<information>'.**

---

**Error Type:**

Error

**Possible Cause:**

The port conflicts with another application.

**Possible Solution:**

Use the server administration settings to update the Script Engine port.

---

**An unhandled exception was thrown from the script. | Function = '<function>', error = '<error>'.**

---

**Error Type:**

Error

**Possible Cause:**

An exception was thrown from the script.

**Possible Solution:**

Correct the condition that lead to the exception, or update the script logic.

---

**Script Engine service stopping.**

---

**Error Type:**

Informational

---

**Script Engine service starting.**

---

**Error Type:**

Informational

---

**Connection to ThingWorx failed. | Platform <host:port resource>, error: <reason>.**

---

**Error Type:**

Error

**Possible Cause:**

The connection to the ThingWorx platform could not be established.

**Possible Solution:**

1. Verify that the host, port, resource, and application key are all valid and correct.
2. Verify that the host machine can reach the composer on the ThingWorx platform.
3. Verify that the proper certificate settings are enabled if using a self-signed certificate or no encryption.

---

**Error adding item. | Item name: '<item name>'.**

---

**Error Type:**

Error

**Possible Cause:**

The item <TagName> could not be added to the server for scanning.

**Possible Solution:**

1. Verify that the tag exists on a valid channel and device.
2. Verify that the tag may be read using another client, such as the QuickClient.

---

**Failed to trigger the autobind complete event on the platform.**

---

**Error Type:**

Error

**Possible Cause:**

The ThingWorx connection was terminated before the autobind process completed.

**Possible Solution:**

Wait to reinitialize or alter the ThingWorx project properties until after all autobinds have completed.

---

**Connection to ThingWorx failed for an unknown reason. | Platform <host:port resource>, error: <error>.**

---

**Error Type:**

Error

**Possible Cause:**

The connection to the ThingWorx platform failed.

**Possible Solution:**

1. Verify that the host, port, resource, and application key are all valid and correct.
2. Verify that the host machine can reach the composer on the ThingWorx platform.
3. Verify that the proper certificate settings are enabled if using a self-signed certificate or no encryption.
4. Contact technical support with the error code and an application report.

---

**One or more value change updates lost due to insufficient space in the connection buffer. | Number of lost updates: <count>.**

---

**Error Type:**

Error

**Possible Cause:**

Data is being dropped because the ThingWorx platform is not available or too much data is being collected by the instance.

**Possible Solution:**

1. Verify that some data is updating on the ThingWorx Platform and that the platform is reachable.
2. Slow down the tag scan rate to move less data into the ThingWorx Platform.

**Item failed to publish; multidimensional arrays are not supported. | Item name: '%s'.**

---

**Error Type:**

Error

**Possible Cause:**

The item <ItemName> references a tag whose data is a multidimensional array.

**Possible Solution:**

Modify the item to reference a tag with a supported datatype.

**Connection to ThingWorx was closed. | Platform: <host:port resource>.**

---

**Error Type:**

Warning

**Possible Cause:**

The connection was closed. The service was stopped or the interface is no longer able to reach the platform.

**Possible Solution:**

1. Verify that the native interface is enabled in the project properties.
2. Verify that the host machine can reach the composer on the ThingWorx platform.

**Failed to autobind property. | Name: '<property name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

A property with this name already exists under this Thing.

**Possible Solution:**

1. Check the property to see if data is current.
2. If data is not current, delete the property under your Thing and run the addItem service once again.

---

**Failed to restart Thing. | Name: '<thing name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

When the AddItem service is complete, a restart service is called on the Thing. This allows the Composer to visualize the changes. Data changes are sent to the platform even when this error has been presented.

**Possible Solution:**

Relaunch the composer to restart the Thing.

---

**Write to property failed. | Property name: '<name>', reason: <reason>.**

---

**Error Type:**

Warning

**Possible Cause:**

Unable to write to a tag due to a conversion issue.

**Possible Solution:**

1. Verify that the data type of the tag in KEPServerEX, as well as in the ThingWorx Platform, is correct and consistent.
2. Verify that the value to be written is within the appropriate range for the data type.

---

**ThingWorx request to add item failed. The item was already added. | Item name: '<name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

The tag had already been added to this Thing.

**Possible Solution:**

1. Check the property to see if data is current.
2. If data is not current, delete the property under your Thing and run the addItem service once again.

---

**ThingWorx request to remove item failed. The item doesn't exist. | Item name: '<name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

The tag was already removed from the Thing or no such tag exists.

**Possible Solution:**

If the tag still shows under the properties of the Thing, delete that property in the ThingWorx Composer.

**The server is configured to send an update for every scan, but the push type of one or more properties are set to push on value change only. | Count: <count>.**

---

**Error Type:**

Warning

**Possible Cause:**

The push type in the ThingWorx platform is set to change only for some items. This push type only updates data on the platform when the data value changes.

**Possible Solution:**

To use the Send Every Scan option, set this value to Always.

**The push type of one or more properties are set to never push an update to the platform. | Count: <count>.**

---

**Error Type:**

Warning

**Possible Cause:**

The push type in the ThingWorx platform is set to Never for some items, which prevents any data changes from being automatically updated on the platform.

**Possible Solution:**

If this is not the desired behavior, change the push type in the ThingWorx platform.

**ThingWorx request to remove an item failed. The item is bound and the force flag is false. | Item name: '<name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

The RemoveItems service could not remove the item because it is bound to a property and the Force Flag is not set to True.

**Possible Solution:**

Re-run the service, explicitly calling the ForceRemove flag as True.

**Write to property failed. | Thing name: '<name>', property name: '<name>', reason: <reason>.**

---

**Error Type:**

Warning

**Possible Cause:**

Unable to write to a tag due to a conversion issue.

**Possible Solution:**

1. Verify that the data type of the tag in KEPServerEX, as well as in the ThingWorx Platform, is correct and consistent.
2. Verify that the value to be written is within the appropriate range for the data type.

---

**Error pushing property updates to thing. | Thing name: '<name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

Property updates for the named thing were not successfully published to the platform.

**Possible Solution:**

Check the platform's log for an indication of why property updates are failing, such as a permissions issue.

---

**Connected to ThingWorx. | Platform: <host:port resource>, Thing name: '<name>'.**

---

**Error Type:**

Informational

**Possible Cause:**

A connection was made to the ThingWorx platform.

---

**Reinitializing ThingWorx connection due to a project settings change initiated from the platform.**

---

**Error Type:**

Informational

**Possible Cause:**

When using the SetConfiguration service, this message informs an operator viewing the KEPServerEX event log that a change was made.

---

**Dropping pending autobinds due to interface shutdown or reinitialize. | Count: <count>.**

---

**Error Type:**

Informational

**Possible Cause:**

A server shutdown or initialization was called while auto-binding was in process from an AddItems service call.

**Possible Solution:**

Any Items not auto bound will need to be manually created and bound in the ThingWorx Composer.

---

**Serviced one or more autobind requests. | Count: <count>.**

---

**Error Type:**

Informational

**Possible Cause:**

Part of the AddItems service is the autobind action. This action may take more time than the actual adding of the item. This message alerts the operator to how many items have been autobound.

---

**Reinitializing ThingWorx connection due to a project settings change initiated from the Configuration API.**

---

**Error Type:**

Informational

**Possible Cause:**

When using the Configuration API, this message informs an operator viewing the KEPServerEX event log that a change was made.

---

**Resumed pushing property updates to thing: the error condition was resolved. | Thing name: '<name>'.**

---

**Error Type:**

Informational

---

**Configuration transfer from ThingWorx initiated.**

---

**Error Type:**

Informational

---

**Configuration transfer from ThingWorx aborted.**

---

**Error Type:**

Informational

---

**A socket error occurred listening for client connections. | Endpoint URL = '<endpoint URL>', Error = <error code>, Details = '<description>'.**

---

**Error Type:**

Error

---

**The UA Server failed to register with the UA Discovery Server. | Endpoint URL: '<endpoint url>'.**

---

**Error Type:**

Error



**The UA Server failed to unregister from the UA Discovery Server. | Endpoint URL: '<endpoint url>'.**

---

**Error Type:**

Warning

**The UA Server successfully registered with the UA Discovery Server. | Endpoint URL: '<endpoint url>'.**

---

**Error Type:**

Informational

**The UA Server successfully unregistered from the UA Discovery Server. | Endpoint URL: '<endpoint url>'.**

---

**Error Type:**

Informational

**Sample Profile Library event log message. Reason: <reason>.**

---

**Error Type:**

Warning

**Possible Cause:**

1. Sample Profile Library event log possible cause 1.
2. Sample Profile Library event log possible cause 2.
3. Sample Profile Library event log possible cause 3.
4. Sample Profile Library event log possible cause 4.
5. Sample Profile Library event log possible cause 5.

**Possible Solution:**

Sample Profile Library event log possible solution.

**Driver failed to initialize.**

---

**Error Type:**

Error

**Unable to create serial I/O thread.**

---

**Error Type:**

Error

**Possible Cause:**

The server process has no resources available to create new threads.

**Possible Solution:**

Each tag group consumes a thread. The typical limit for a single process is about 2000 threads. Reduce the number of tag groups in the project.

---

**Connection failed. Unable to bind to adapter. | Adapter = '<name>'.**

---

**Error Type:**

Error

**Possible Cause:**

Since the specified network adapter cannot be located in the system device list, it cannot be bound to for communications. This can occur when a project is moved from one PC to another (and when the project specifies a network adapter rather than using the default). The server reverts to the default adapter.

**Possible Solution:**

Change the Network Adapter property to Default (or select a new adapter), save the project, and retry.

---

**Device is not responding.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The connection between the device and the host PC is broken.
2. The communication parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect device ID.
4. The response from the device took longer to receive than allowed by the Request Timeout device setting.

**Possible Solution:**

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters match those of the device.
3. Verify that the device ID for the named device matches that of the actual device.
4. Increase the Request Timeout setting to allow the entire response to be handled.

---

**Device is not responding. | ID = '<device>'.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The network connection between the device and the host PC is broken.
2. The communication parameters configured for the device and driver do not match.

3. The response from the device took longer to receive than allowed by the Request Timeout device setting.

**Possible Solution:**

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communications parameters match those of the device.
3. Increase the Request Timeout setting to allow the entire response to be handled.

---

**Unable to write to address on device. | Address = '<address>'.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The connection between the device and the host PC is broken.
2. The communications parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect device ID.

**Possible Solution:**

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communication parameters match those of the device.
3. Verify that the device ID given to the named device matches that of the actual device.

---

**Items on this page may not be changed while the driver is processing tags.**

---

**Error Type:**

Warning

**Possible Cause:**

An attempt was made to change a channel or device configuration while data clients were connected to the server and receiving data from the channel/device.

**Possible Solution:**

Disconnect all data clients from the server before making changes.

---

**Specified address is not valid on device. | Invalid address = '<address>'.**

---

**Error Type:**

Warning

**Possible Cause:**

A tag address has been assigned an invalid address.

**Possible Solution:**

Modify the requested address in the client application.

**Address '<address>' is not valid on device '<name>'.**

---

**Error Type:**

Warning

**This property may not be changed while the driver is processing tags.**

---

**Error Type:**

Warning

**Unable to write to address '<address>' on device '<name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The connection between the device and the host PC is broken.
2. The communications parameters for the connection are incorrect.
3. The named device may have been assigned an incorrect device ID.

**Possible Solution:**

1. Verify the cabling between the PC and the PLC device.
2. Verify that the specified communication parameters match those of the device.
3. Verify that the device ID given to the named device matches that of the actual device.

**Socket error occurred connecting. | Error = <error>, Details = '<information>'.**

---

**Error Type:**

Warning

**Possible Cause:**

Communication with the device failed during the specified socket operation.

**Possible Solution:**

Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

**Socket error occurred receiving data. | Error = <error>, Details = '<information>'.**

---

**Error Type:**

Warning

**Possible Cause:**

Communication with the device failed during the specified socket operation.

**Possible Solution:**

Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

**Socket error occurred sending data. | Error = <error>, Details = '<information>'.**

---

**Error Type:**

Warning

**Possible Cause:**

Communication with the device failed during the specified socket operation.

**Possible Solution:**

Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

**Socket error occurred checking for readability. | Error = <error>, Details = '<information>'.**

---

**Error Type:**

Warning

**Possible Cause:**

Communication with the device failed during the specified socket operation.

**Possible Solution:**

Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

**Socket error occurred checking for writability. | Error = <error>, Details = '<information>'.**

---

**Error Type:**

Warning

**Possible Cause:**

Communication with the device failed during the specified socket operation.

**Possible Solution:**

Follow the guidance in the error and details, which explain why the error occurred and suggest a remedy when appropriate.

**%s |**

---

**Error Type:**

Informational

---

**<Name> Device Driver '<name>'**

---

**Error Type:**

Informational

**Could not load item state data. Reason: <reason>.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The driver could not load the item state data for the specified reason.
2. Corrupt data files.
3. Inadequate disk space.
4. Invalid drive in path.
5. Deleted or renamed data files.

**Possible Solution:**

Solution depends upon the reason given in the error message. In the case of file corruption or deletion, previous state data is lost.

**Could not save item state data. Reason: <reason>.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The driver could not save the item state data for the specified reason.
2. Corrupt data files.
3. Inadequate disk space.
4. Invalid drive in path.
5. Deleted or renamed data files.

**Possible Solution:**

Solution depends upon the reason given in the error message. In the case of file corruption or deletion, previous state data is lost.

**Failed to load the license interface, possibly due to a missing third-party dependency. Run in Time Limited mode only.**

---

**Error Type:**

Error

**Possible Cause:**

One or more required OEM licensing component is missing the system.

**Possible Solution:**

Contact a sales or support representative for assistance.

**Time Limited mode has expired.**

---

**Error Type:**

Warning

**Possible Cause:**

1. The product has not been purchased and licensed during Time Limited mode.
2. The server started in Time Limited mode with the specified time remaining in Time Limited mode.

**Possible Solution:**

1. If evaluating the server, no action needs to be taken.
2. If this is a production machine, activate the product licenses for the installed components before Time Limited mode expires.
3. Purchase a license for all features of the product that will be used.
4. Contact a sales or support representative for assistance.

**Type <numeric type ID> limit of <maximum count> exceeded on feature '<name>'.**

---

**Error Type:**

Warning

**Possible Cause:**

The installed feature license limits the number of items of the specified type that can be configured.

**Possible Solution:**

1. Contact customer solutions to determine what object type count should be reduced to remain within the limits of the license.
2. If more items are needed, contact a sales representative about upgrading the license.

**The <name> feature license has been removed. The server will enter Time Limited mode unless the license is restored before the grace period expires.**

---

**Error Type:**

Warning

**Possible Cause:**

The feature license has been deleted, moved to another machine, the hardware key has been removed, or trusted storage has been corrupted.

**Possible Solution:**

1. Consult the Licensing User Manual for instructions on activating an emergency licenses.
2. Contact a sales or support representative for assistance.

---

**Feature <name> is time limited and will expire at <date/time>.**

---

**Error Type:**

Warning

---

**Feature <name> is time limited and will expire at <date/time>.**

---

**Error Type:**

Warning

---

**Time limited usage period on feature <name> has expired.**

---

**Error Type:**

Warning

---

**Cannot add item. Requested count of <number> would exceed license limit of <maximum count>.**

---

**Error Type:**

Informational

**Possible Cause:**

The product license limits the number of items that can be configured.

**Possible Solution:**

1. Verify the number authorized by the license and correct the project to use only that number of items.
2. If more items are needed, contact a sales representative about upgrading the license.



# Index

## %

%s | 117

## <

<feature name> is required to load this project. 93

<feature name> was not found or could not be loaded. 93

<Name> Device Driver '<name>' 118

<Name> successfully configured to run as a system service. 104

<Name> successfully removed from the service control manager database. 104

<Product> device driver loaded successfully. 98

<Source>

Invalid Ethernet encapsulation IP '<address>'. 97

## A

A client application has disabled auto-demotion on device '<device>'. 100

A socket error occurred listening for client connections. | Endpoint URL = '<endpoint URL>', Error = '<error code>', Details = '<description>'. 112

Access to object denied. | User = '<account>', Object = '<object path>', Permission = 100

Addition of object to '<name>' failed

<reason>. 102

Address '<address>' is not valid on device '<name>'. 116

aksusbd 13

Alias Name 50

Alias Properties 50

An unhandled exception was thrown from the script. | Function = '<function>', error = '<error>'. 106

APPKEY 28

Application Data 12

Architecture 11, 52

Attempt to add item '<name>' failed. 101

Attempting to automatically generate tags for device '<device>'. 99

authentication 51

Authentication 87

Authorization 16

Auto-generated tag '<tag>' already exists and will not be overwritten. 94

Auto generation produced too many overwrites, stopped posting error messages. 95

Automatic Start 29

Automatic Tag Generation 53

## B

BCD 47

Boolean 47

Byte 47

## C

Cannot add device. A duplicate device may already exist in this channel. 94

Cannot add item. Requested count of <number> would exceed license limit of <maximum count>. 120

certificate 31

Changing runtime operating mode. 100

Char 47

Child Endpoints 68

Clamp 49

Command line 10

Command Line Interface 14

Completed automatic tag generation for device '<device>'. 99

Components and Concepts 31

Concurrent Clients 86

Configuration API Service 51

Configuration session assigned to <name> demoted to read only. 105

Configuration session assigned to <name> has ended. 105

Configuration session assigned to <name> promoted to write access. 105

Configuration session started by <name> (<name>). 105

Configuration transfer from ThingWorx aborted. 112

Configuration transfer from ThingWorx initiated. 112

Configuring User Group Project Permissions 72

Connected to ThingWorx. | Platform

```
<host  
  port resource>, Thing name  
  '<name>'. 111
```

Connection failed. Unable to bind to adapter. | Adapter = '<name>'. 114

```
Connection to ThingWorx failed for an unknown reason. | Platform <host  
  port resource>, error  
  <error>. 107
```

Connection to ThingWorx failed. | Platform <host  
port resource>, error  
<reason>. 106

Connection to ThingWorx was closed. | Platform  
<host  
port resource>. 108

Connectivity 18

Content Retrieval 80

Could not load item state data. Reason  
<reason>. 118

Could not open project file  
'<name>'. 102

Could not save item state data. Reason  
<reason>. 118

Create MQTT Agent 30

Create MQTT Agent Tag 30

Created backup of project '<name>' to '<path>'. 99

Creating a Channel 56

Creating a Device 59

Creating a Tag 61

Creating a UA Endpoint 76

Creating a User 70

Creating a User Group 71

cURL 10

Curl 56

## D

daemon 13

Data 77

Data collection is disabled on device '<device>'. 99

Data collection is enabled on device '<device>'. 99

DELETE 58, 61, 63, 66

Delete MQTT Agent 31

Delete object '<name>' failed  
<reason>. 103

Device '<device>' has been auto-promoted to determine if communications can be re-established. 99

Device discovery has exceeded <count> maximum allowed devices. Limit the discovery range and try again. 93

Device is not responding. 114

Device is not responding. | ID = '<device>'. 114

Documentation Endpoint 17

Double 47  
Driver failed to initialize. 113  
Dropping pending autobinds due to interface shutdown or reinitialize. | Count  
<count>. 111  
DWord 47  
Dynamic Tags 47

## E

edge\_admin 14  
Enable 87  
Endpoint 16  
Endpoint Mapping 18  
Error adding item. | Item name  
'<item name>'. 107  
Error pushing property updates to thing. | Thing name  
'<name>'. 111  
Event Log Messages 89

## F

Failed to add tag '<tag>' because the address is too long. The maximum address length is <number>. 95  
Failed to autobind property. | Name  
'<property name>'. 108  
Failed to load library  
<name>. 99  
Failed to load the license interface, possibly due to a missing third-party dependency. Run in Time Limited mode only. 118  
Failed to read build manifest resource  
<name>. 99  
Failed to reset password for administrator. | Administrator name = '<name>'. 101  
Failed to restart Thing. | Name  
'<thing name>'. 109  
Failed to start Script Engine server. Socket error occurred binding to local port. | Error = <error>, Details = '<information>'. 106  
Failed to trigger the autobind complete event on the platform. 107  
Failed to update startup project '<name>'  
<reason>. 103  
Feature <name> is time limited and will expire at <date/time>. 120  
Filename contains one or more invalid characters. 102  
Filename is expected to be of the form <subdir>/<name>.{json,opf} 102  
Filename must not be empty. 102

Filename must not overwrite an existing file

'<name>'. 102

Filtering 90

Float 47

## G

GET Request URI 81

Getting Started 14

## H

Hardware error on line '<line>'. 96

hardware key 12

HASP 13

Hierarchy 78

HOSTNAME 28

HTTP 51

HTTPS 51

Human Machine Interface (HMI) 20

## I

Ignoring user-defined startup project because a configuration session is active. 103

Initialization 51

Insomnia 56

installation directory 13

Installing 11

Instance Certificate 14

Interfaces and Connectivity 20

Introduction 10

Invalid Model encountered while trying to load the project. | Device = '<device>'. 94

Invalid project file

'<name>'. 102

Invalid project file. 91

Invalid XML document 92

IoT Gateway 29, 31

Item failed to publish 108

Items on this page may not be changed while the driver is processing tags. 115

**J**

Java Runtime 10  
Job 52  
Job Cleanup 53  
JSON Response Structure 81

**K**

KeyStore 31  
keytool 31

**L**

LBCD 47  
Licensing 12  
Line '<line>' is already in use. 95  
Linear 48  
Linux 10  
Listat 13  
LLong 47  
Logging 89  
Long 47  
LSB 10

**M**

Man Machine Interface (MMI) 20  
Mapped to 50  
Member 83  
Missing application data directory. 105  
Move object '<name>' failed  
    <reason>. 102  
MQTT Agent 19, 29, 31  
MQTT client 10  
multidimensional arrays are not supported. | Item name  
    '%s'. 108  
Multiple Objects 78

**N**

Negate 49

No comm handle provided on connect for line '<line>'. 96

No device driver DLLs were loaded. 102

No tags were created by the tag generation request. See the event log for more information. 98

**O**

Object 77

Object type '<name>' not allowed in project. 99

One or more value change updates lost due to insufficient space in the connection buffer. | Number of lost updates  
<count>. 107

OPC UA 20

OPC UA Endpoint 73

OPC UA server 87

OpenJDK 10

Operation 51

**P**

Password 85

Password for user has been changed. | User = '<name>'. 101

Permissions change applied on configuration session assigned to <name>. 105

Permissions definition has changed on user group. | Group = '<name>'. 101

Plug-in Endpoints 18

Port 16, 87

PORT 28

Postman 10, 56

Prerequisites 29

Project Permissions 68

Project Save 55

ProjectSave 22

Properly Name a Channel, Device, Tag, and Tag Group 51

Property Definitions 83

Property Tags 44

Property Types 85

**Q**

QWord 47

**R**

Raw 49

Reinitializing ThingWorx connection due to a project settings change initiated from the Configuration API. 112

Reinitializing ThingWorx connection due to a project settings change initiated from the platform. 111

Rejecting attempt to change model type on a referenced device '<channel device>'. 96

Rejecting request to replace the project because it's the same as the one in use '<name>'. 102

Removing a Device 61

Removing a Tag 63

Removing a Tag Group 66

Removing a UA Endpoint 77

Removing Channel 58

Response Codes 89

REST 15, 51, 56, 60-61

Resumed pushing property updates to thing  
the error condition was resolved. | Thing name  
'<name>'. 112

Runtime operating mode change completed. 100

Runtime performing exit processing. 104

Runtime process started. 104

Runtime project replaced from '<name>'. 105

Runtime project replaced with startup project defined. Runtime project will be restored from '<name>' at next restart. 103

Runtime project replaced. 105

Runtime project saved as '<name>'. 105

Runtime re-initialization completed. 104

Runtime re-initialization started. 104

Runtime service started. 104

Runtime shutdown complete. 104

**S**

Sample Profile Library event log message. Reason  
<reason>. 113



Scaled 49  
Scan rate override 50  
Script Engine service starting. 106  
Script Engine service stopping. 106  
security 16  
Security 51-52, 80, 87  
Self-Signed Certificates 31  
Service 52  
Serviced one or more autobind requests. | Count  
    <count>. 112  
Short 47  
Shutdown 51  
Shutting down to perform an installation. 100, 105  
Socket error occurred checking for readability. | Error = <error>, Details = '<information>'. 117  
Socket error occurred checking for writability. | Error = <error>, Details = '<information>'. 117  
Socket error occurred connecting. | Error = <error>, Details = '<information>'. 116  
Socket error occurred receiving data. | Error = <error>, Details = '<information>'. 116  
Socket error occurred sending data. | Error = <error>, Details = '<information>'. 117  
Specified address is not valid on device. | Invalid address = '<address>'. 115  
Square Root 48  
Starting <name> device driver. 98  
Statistics Tags 45  
Stopping <name> device driver. 98  
String 47  
System Requirements 10  
System Services 52  
System Tags 35  
systemctl 13

## T

Tag Group Properties 50  
Tag Properties — General 33  
Tag Properties — Scaling 48  
TAPI configuration has changed, reinitializing... 98  
The '<product>' driver does not currently support XML persistence. Save using the default file format. 97  
The <name> device driver was not found or could not be loaded. 90  
The <name> feature license has been removed. The server will enter Time Limited mode unless the  
    license is restored before the grace period expires. 119

The Config API is unable to load the SSL certificate. 90

The Config API SSL certificate contains a bad signature. 90

The Config API SSL certificate has expired. 90

The Config API SSL certificate is self-signed. 90

The project file was created with a more recent version of this software. 99

The push type of one or more properties are set to never push an update to the platform. | Count <count>. 110

The server is configured to send an update for every scan, but the push type of one or more properties are set to push on value change only. | Count <count>. 110

The specified network adapter is invalid on channel '%1' | Adapter = '%2'. 98

The time zone set for '<device>' is '<zone>'. This is not a valid time zone for the system. Defaulting the time zone to '<zone>'. 98

The UA Server failed to register with the UA Discovery Server. | Endpoint URL '<endpoint url>'. 112

The UA Server failed to unregister from the UA Discovery Server. | Endpoint URL '<endpoint url>'. 113

The UA Server successfully registered with the UA Discovery Server. | Endpoint URL '<endpoint url>'. 113

The UA Server successfully unregistered from the UA Discovery Server. | Endpoint URL '<endpoint url>'. 113

THING\_NAME 28

ThingWorx Example 28

ThingWorx Native Interface 19, 21

ThingWorx Platform 10

ThingWorx request to add item failed. The item was already added. | Item name '<name>'. 109

ThingWorx request to remove an item failed. The item is bound and the force flag is false. | Item name '<name>'. 110

ThingWorx request to remove item failed. The item doesn't exist. | Item name '<name>'. 109

This property may not be changed while the driver is processing tags. 116

Time-Limited 13

Time Limited mode has expired. 119

Time limited usage period on feature <name> has expired. 120

Trust Store 14

twxedge 13

Type <numeric type ID> limit of <maximum count> exceeded on feature '<name>'. 119

Type Definitions 82

## U

- UA Server 19
- UaExpert 86
- Ubuntu 10, 13
- Unable to add channel due to driver-level failure. 91
- Unable to add device due to driver-level failure. 92
- Unable to backup project file to '<path>' [<reason>]. The save operation has been aborted. Verify the destination file is not locked and has read/write access. To continue to save this project without a backup, deselect the backup option under Tools | Options | General and re-save the project. 93
- Unable to create serial I/O thread. 113
- Unable to generate a tag database for device '<device>' 95
- Unable to generate a tag database for device '<device>'. The device is not responding. 94
- Unable to load project <name> 92
- Unable to load startup project '<name>'<br><reason>. 103
- Unable to load the '<name>' driver because more than one copy exists ('<name>' and '<name>').<br>Remove the conflicting driver and restart the application. 91
- Unable to load the project due to a missing object. | Object = '<object>'. 94
- Unable to save project file <name> 93
- Unable to start the Config API Service. Possible problem binding to port. 90
- Unable to use network adapter '<adapter>' on channel '<name>'. Using default network adapter. 96
- Unable to write to address '<address>' on device '<name>'. 116
- Unable to write to address on device. | Address = '<address>'. 115
- Unable to write to item '<name>'. 103
- Update MQTT Agent 30
- Update of object '<name>' failed<br><reason>. 103
- Updated startup project '<name>'. 104
- Updating a Channel 57
- Updating a Device 60
- Updating a Tag 62
- Updating a Tag Group 64
- Updating a UA Endpoint 76
- Updating a User 71
- Updating a User Group 72
- URL 87
- USB Key 13
- User added to user group. | User = '<name>', Group = '<name>'. 100
- User group has been created. | Group = '<name>'. 100

User group has been disabled. | Group = '<name>'. 101  
User group has been enabled. | Group = '<name>'. 101  
User group has been renamed. | Old name = '<name>', New name = '<name>'. 101  
User Groups 66  
User has been disabled. | User = '<name>'. 101  
User has been enabled. | User = '<name>'. 101  
User has been renamed. | Old name = '<name>', New name = '<name>'. 101  
User information replaced by import. | File imported = '<absolute file path>'. 100  
User Management 66  
User moved from user group. | User = '<name>', Old group = '<name>', New group '<name>'. 100  
Users 69

## V

Validation error on '<tag>'  
    <error>. 97  
    Invalid scaling parameters. 97  
Version mismatch. 92  
View MQTT Agent Tags 30  
View MQTT Agents 30

## W

What is a Channel? 32  
What is a Device? 32  
What is a Tag Group? 49  
What is a Tag? 33  
What is the Alias Map? 50  
What is the Event Log? 51  
Word 47  
Write request failed on item '<name>'. Error scaling the write data. 103  
Write request failed on item '<name>'. The write data type '<type>' cannot be converted to the tag data type '<type>'. 103  
Write request rejected on item reference '<name>' since the device it belongs to is disabled. 104  
Write request rejected on read-only item reference '<name>'. 103  
Write to property failed. | Property name  
    '<name>', reason  
    <reason>. 109

Write to property failed. | Thing name

'<name>', property name

'<name>', reason

<reason>. 110