

thingworx® keeware® edge

Quick Start Guide

© 2019 PTC Inc. All Rights Reserved.

Table of Contents

Quick Start Guide	1
Table of Contents	2
ThingWorx Kepware Edge System Requirements	3
Installing ThingWorx Kepware Edge	3
ThingWorx Kepware Edge Licensing	4
Getting Started	5
Enabling Interfaces	7
Configuration API — Project Example	8
Configuration API Service — Creating a User	10
Configuration API Allen-Bradley ControlLogix Example	10
Configuration API Modbus Ethernet Example	12
Configuration API Siemens TCP/IP Ethernet Example	14
Configuration API ThingWorx Connection	15
Configuring an IoT Gateway	16
Using UaExpert	18
Configuration API Service — Creating a UA Endpoint	20
Configuration API Service — Logging	21

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.

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
```

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.

• *For Licensing troubleshooting, please refer to the server help documentation.*

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 -i 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.

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, `tkeedge` 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.

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
}
```

 For additional information on creating and configuring OPC UA endpoints, refer to the server help file.

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
}
```

 For a complete list of settings, refer to the server help documentation.

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 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 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-Bradley%20ControlLogix%20Ethernet/channels
```

Device Definitions

Endpoint (GET):

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

Device Definitions

Endpoint (GET):

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

Device Definitions

Endpoint (GET):

```
https://<hostname_or_ip>:<-  
port>/config/v1/doc/drivers/Siemens%20TCP%20FIP%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"
}
```

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.

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 — Logging

Messages from the event log service can be read from a REST client by sending a GET to `https://<hostname>:<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.