

# kepware<sup>®</sup> edge

## Quick Start Guide

© 2026 Kepware. All Rights Reserved.

# Table of Contents

<b>Quick Start Guide</b> .....	<b>1</b>
<b>Table of Contents</b> .....	<b>2</b>
<b>Introduction</b> .....	<b>3</b>
Kepware Edge System Requirements .....	3
Installation .....	4
Licensing .....	4
<b>Getting Started</b> .....	<b>5</b>
Enabling Interfaces .....	6
Configuration API Service – Project Example .....	7
Configuration API Service – Creating a User .....	8
Configuring the ThingWorx Native Interface .....	8
Configuring the IoT Gateway .....	9
<b>Connecting with an OPC UA Client Using UaExpert</b> .....	<b>10</b>
Configuration API Service – Creating a UA Endpoint .....	12

## Introduction

---

Version 1.863

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. It is a Linux-based solution deployed via a container to accelerate deployment, reduce maintenance costs, and mitigate cyber security risks. Built by the industrial connectivity experts, Kepware Edge eliminates the interoperability challenges associated with implementing digital solutions.

Configuration of the server is performed using the web user interface, the Configuration API accessed via a REST client application / tool (not included), and the `edge_admin` command line interface tool. The web user interface or Configuration API can be used to modify project settings and administrative settings. The `edge_admin` is used to manage certificates and configure the Configuration API administrative settings.

The following drivers are supported:

- [Allen-Bradley ControlLogix Ethernet](#)
- [Mitsubishi Ethernet](#)
- [Modbus Ethernet](#)
- [OPC UA Client](#)
- [Siemens TCP/IP Ethernet](#)
- [Siemens S7 Plus Ethernet](#)

• **See Also:** [Help for Kepware Drivers](#)

Kepware Edge can also be managed online through the [Kepware+ Manager](#) online portal.

## Kepware Edge System Requirements

---

The product has been tested and verified on modern computer hardware running Red Hat Enterprise Linux. It currently only runs on X86\_64 platforms.

### Prerequisites

- Linux x86-64 CPU Architecture
- Container runtime: Any modern docker/OCI compliant container runtime (Docker, Kubernetes, Podman, Rancher, etc.)
- Host OS: Any modern x86-64 OS (i.e. RHEL, Ubuntu, Fedora, Windows, Windows Server, MacOS, etc.)

### Knowledge Requirements

- Linux operating system and commands
- RESTful interfaces
- Command line or API utilities, such as Postman or cURL
- Container orchestration tools (Docker, Podman, Kubernetes, etc.)
- ThingWorx Platform (if used)
- OPC UA configuration and connectivity (if used)
- MQTT Client interfaces and connectivity (if used)

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

• **See Also:** The [licensing server](#) user manual for related system requirements.

## Installation

Before installing 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 [Article - CS278624](#) (login required).

```
$ sha256sum keppure_edge*
```

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

To install, run the following command:

```
$ ./keppure_edge*.run
```

For all installation options, run the following command:

```
$ ./keppure_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 Kepware Edge by running the 'apt update' command.

● A password should be set for the Administrator account during installation. To skip setting a password significantly reduces the security of the installation. The Administrator account is specific to the product installation; it is not the general operating system Administrator account.

● Administrator passwords must be at least 14 characters and no more than 512 characters. Passwords should be at least 14 characters and include a mix of uppercase and lowercase letters, numbers, and special characters. Choose a strong unique password that avoids well-known, easily guessed, or common passwords. Passwords greater than 512 characters will be truncated.

● The Administrator user account password cannot be reset, but additional administrative users can be added to the Administrator user group. Best practices suggest each user with administrative access be assigned unique accounts and passwords to ensure audit integrity and continual access through role and staff changes.

● Once installed, any Linux user accounts administering the Kepware Edge instance must be added to the user group created during the installation, which is keppure 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).

● Verify Java and Python3 are correctly added to the Environment Variables path. If they are not, the following error may occur: "Unable to install ./producer-settings.xml - not presentScript stderr: child process exited abnormally. Error running install-systemd.sh".

## Licensing

Licensing is provided by a license server. If a license cannot be obtained from the license server, unlicensed functionality cannot be used.

● **See Also:** [Kepware Edge License Server User Manual](#)

### Installing a Demo License

Demo licenses are time-limited, but fully functional to allow evaluation of the software. They are distributed by the license server just as a purchased production license is.

### Configuring the License Server Connection

The license server connection can be configured using either the edge\_admin command line tool or the Configuration API.

1. Set the IP address or host name of the server where the license server is running:  
Using Edge Admin:

```
./edge_admin manage-licensing -l <server_address>
```

Using the Configuration API:

Endpoint: (PUT)

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

Body:

```
{
  "libadminsettings.LICENSING_SERVER_NAME": "192.168.1.1"
}
```

2. Import the license server certificate used when configuring the license server:  
Using Edge Admin:

```
./edge_admin manage-truststore -i <cert_file> licensing
```

3. Enable the license server connection:  
Using Edge Admin:

```
./edge_admin manage-licensing --lls-enable
```

Using the Configuration API:

Endpoint: (PUT)

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

Body:

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

**Note:** The server can be configured to run with a self-signed certificate. This configuration is recommended for testing only.

**See Also:** [Configuration API Service – Configuring Licensing Server](#)

## License Recheck

The server periodically checks the license state to verify it is up to date. The server reaches out to the license server requesting to borrow a license every specified check period when a feature in use requires a license. To trigger an immediate check of the license state, use the commands below. This feature might be helpful if new licenses have been added to the license server or if license parameters have changed.

**See Also:** [Kepware Edge License Server User Manual](#)

Using Edge Admin:

```
./edge_admin manage-licensing --force-recheck
```

Using the Configuration API:

Endpoint: (PUT)

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

## Getting Started

Configuration is performed through the user interface, Kepware+ Manager user interface, the Configuration API accessed via a REST client application / tool (not included), or the `edge_admin` command line interface tool. The user interface allows access to project settings, the Event Log and Audit log, and other administrative features from a single portal. 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:

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

**Additional help for the Configuration API** may be accessed by a 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 the Administrator username and password configured during installation. For best security, a new [user](#) should be created via the Configuration API with only the appropriate permissions enabled.

### Ports:

- Configuration API HTTPS interface (Enabled): 57513
- Configuration API HTTP interface (Disabled by default): 57413
- OPC UA interface (Enabled by default): 49330

### REST Configuration API Server 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

• A password should be set for the Kepware Edge Administrator account during installation. To skip setting a password significantly reduces the security of the installation. The Administrator account is specific to the product installation; it is not the general operating system Administrator account.

• The Administrator user account password cannot be reset, but additional administrative users can be added to the Administrator user group. Best practices suggest each user with administrative access be assigned unique accounts and passwords to ensure audit integrity and continual access through role and staff changes.

• Administrator passwords must be at least 14 characters and no more than 512 characters. Passwords should be at least 14 characters and include a mix of uppercase and lowercase letters, numbers, and special characters. Choose a strong unique password that avoids well-known, easily guessed, or common passwords. Passwords greater than 512 characters are truncated.

### Setting up a Project

During container deployment, there is an option to load a sample project by setting an environment variable `USE_SAMPLE_PROJECT` to the value of `TRUE`. If that option was not chosen, 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, re-deploy the container image and ensure that the environmental variable is used:

• **Note:** The sample project is located at `/opt/kepedge/v1/examples/tke_simdemo.lpf` in the containers file system.

### 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 server other than Kepware Edge as unsupported features in the project file may prevent the project from loading.

• **See Also:** [User Interface](#)

### 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 returns a unique project ID necessary to perform a PUT successfully without using the "FORCE\_UPDATE" override.

• **See Also:**

[Connecting with an OPC UA Client](#)

[Configuring the IoT Gateway](#)

[Configuring the ThingWorx Native Interface](#)

## Configuration API Service – Project Example

Project files control the communications and data collection of the server and all connected devices. Channel and device properties are defined and saved in the project file 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 /opt/kepedge/v1/user\_data directory. A local user must be a member of the Kepware Edge user group created during installation, kepedge by default, to be able to place files in this directory. The /opt/kepedge/v1/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 Kepware Edge user and group that were created during installation, kepedge by default.

• **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 /opt/kepedge/v1/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 /opt/kepedge/v1//user\_data/ directory are 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 server 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>"
}

```

● The Administrator user account password cannot be reset, but additional administrative users can be added to the Administrator user group. Best practices suggest each user with administrative access be assigned unique accounts and passwords to ensure audit integrity and continual access through role and staff changes.

● The product Administrator password must be at least 14 characters and no more than 512. Passwords should include a mix of uppercase and lowercase letters, numbers, and special characters. Choose a strong unique password that avoids well-known, easily guessed, or common passwords. Passwords greater than 512 characters are truncated.

## Configuring the ThingWorx Native Interface

To configure the ThingWorx Native Interface 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:** Name of the Industrial Connection defined in the platform.
  - **Tip:** If a name that does not yet exist on the platform is specified, an ephemeral thing is created. To complete the connection, navigate to the new Thing in the platform and save.

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 the ThingWorx Native Interface and configuring the connection settings can be done at the same time.

#### Enable ThingWorx Native 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 Native Test Interface 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 the IoT Gateway

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

#### 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.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
```

## Connecting with an OPC UA Client Using UaExpert

An application like Unified Automation's UaExpert can be used to verify the flow of data from devices through 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 walk-through 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 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_BROWSE_NAMESPACE": 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 "Kepware Edge/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 Kepware Edge/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, these instructions use the [edge\\_admin](#) tool (*see the server help for other methods*).
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:

```
$ ./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.

```
$ ./edge_admin manage-truststore --trust=  
<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"  
}
```