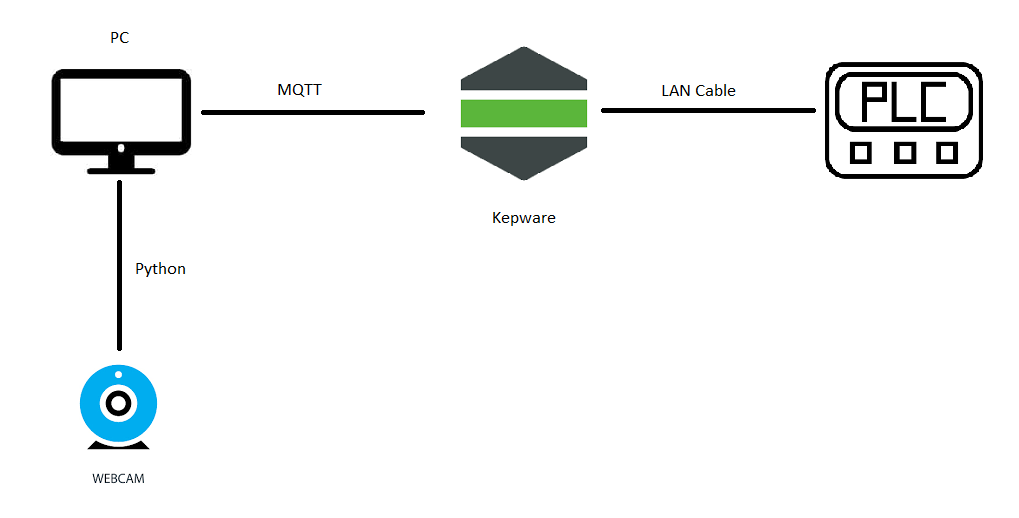
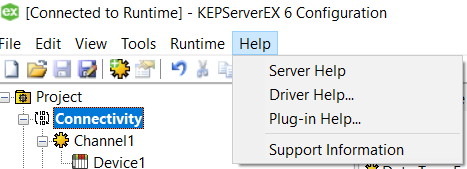
## **Flow of Kepware**



More advanced detail can be found in Help>Server Help

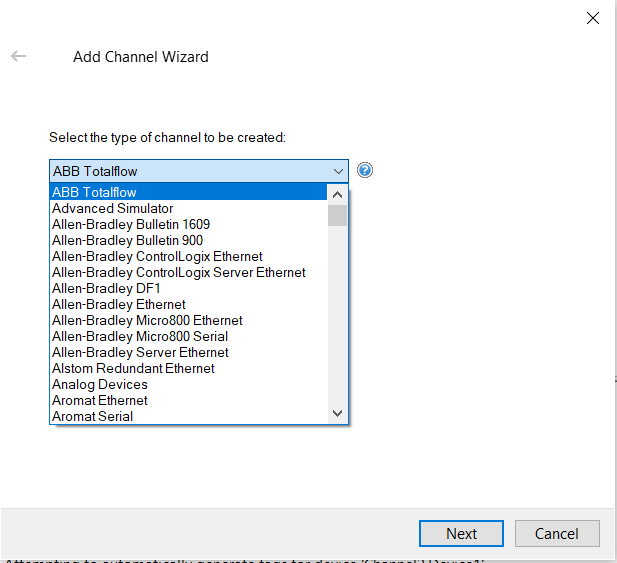
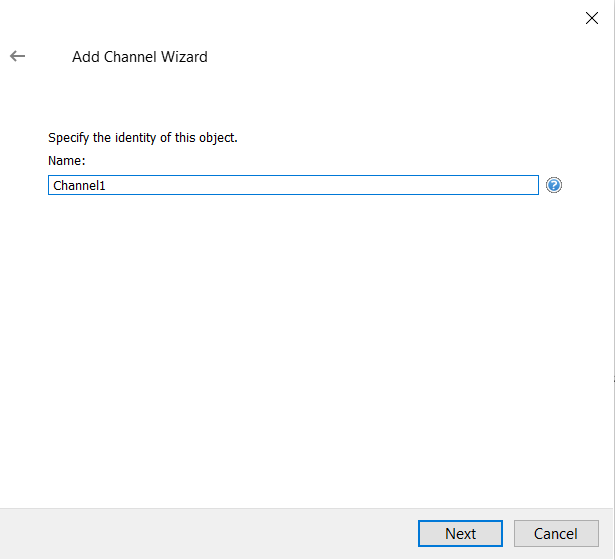


## **Installation**

1. Download the server at <https://www.kepware.com/en-us/products/kepserverex/>
2. Follow through the installation

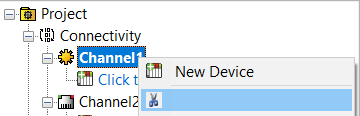
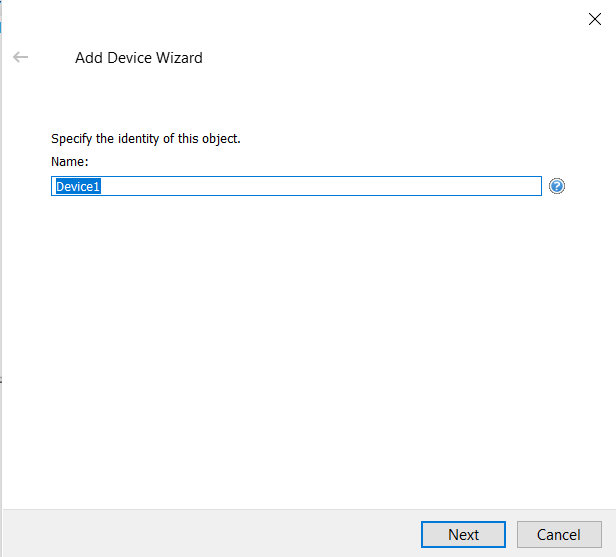
## **Adding and Configuring a Channel**

When creating a new project, users must first determine the communications driver that is required by the application: this is referred to as a channel in the server. A number of channels can be defined within a single project, depending on the driver or drivers installed.

1. In the tree view, right-click on the **Connectivity** node and select **New Channel** (or choose **Edit | Connectivity | New Channel**).  
   
2. Select the type of channel to be created from the drop-down list of available drivers.  
   
3. Click **Next**.
4. Enter a name for the channel to help identify it (used in tag paths, event log messages, and aliasing).  
   
5. Click **Next**.
6. Configure the channel properties according to the options and environment.
7. Review the summary for the new channel and choose **Back** to make changes or **Finish** to close.

## **Adding and Configuring a Device**

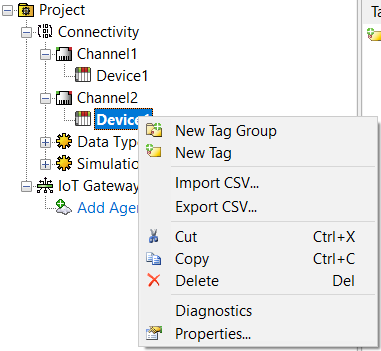
Once a channel has been defined, a device can be added. The device identifies a communication link's physical node or station, and can be thought of as a way to frame the connection's definition to a specific point of interest in the application. In this respect, a device is the correct term for describing the connection to a database object. As such, "device" refers to a specific device on a network, supports multiple device nodes, and allows users to simulate networked devices.

1. In the tree view, locate and select the channel to which device(s) are being added.
2. Right-click and select **New Device** or choose **Edit | Connectivity | New Device**).  
   
3. Enter a name for the device to help identify it (used in tag paths, event log messages, and aliasing).  
   
4. Click **Next**.
5. Configure the device properties according to the options and environment.
6. Review the summary for the new device and choose **Back** to make changes or **Finish** to close.

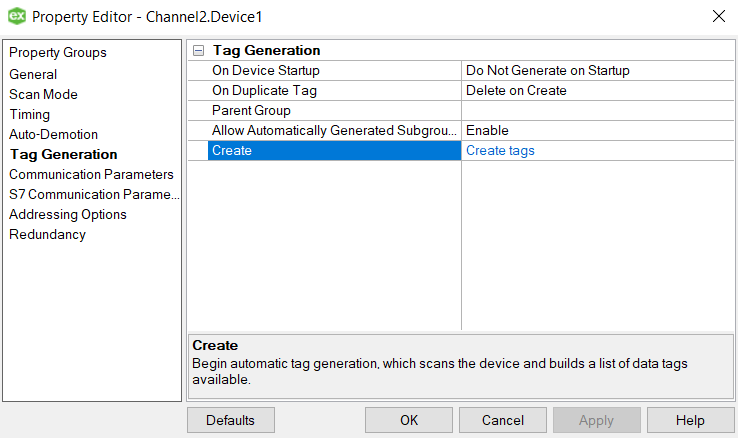
## **Tag Generation**

The OPC server's ability to automatically generate tags for select communication drivers brings OPC technology one step closer to Plug and Play operation. Tag information can be read directly from a device, and tags can also be generated from stored tag data. In either case, users no longer need to manually enter OPC tags into the server.

1. In the tree view, locate and select the device that has been added.
2. Right-click and select **Properties.**

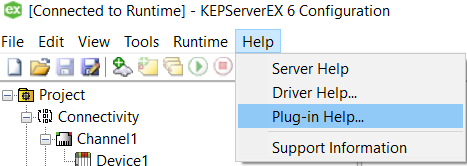


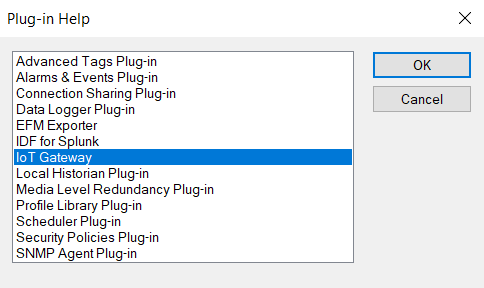
1. In the **Tag Generation**, click the **Create Tag** to scan and list the tag on the device.



## **IoT Gateway**

For more detail on IoT Gateway, Refer the advanced help in **Help**>**Plug-in Help**>**IoT Gateway**

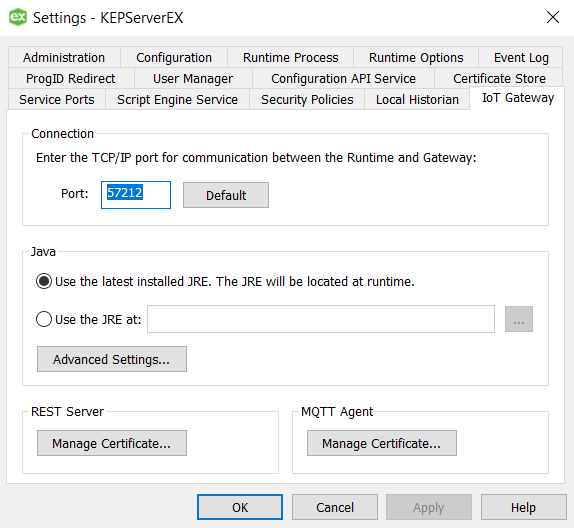




## **Configure the Gateway**

The IoT Gateway administrative settings are automatically configured on installation. The server\_iotgateway service is configured from the Settings sections of the Administration menu. This is where Java settings and gateway-level changes may be made. Generally, these settings do not need to be adjusted once established for the IoT Gateway to function properly. If the settings need to be adjusted, access the IoT Gateway system settings by right-clicking on the Administration icon located in the system tray and selecting Settings | IoT Gateway. The agents and tags themselves are configured from the IoT Gateway section of the user interface.

**Tip**: If the Administrative icon is not in the system tray, re-launch Settings... by navigating to the Administration folder from the Start menu.



In the Connection area:

**Port**: specifies the TCP/IP port that the server runtime and configuration use to communicate with the gateway service. The valid range is 1024 to 65535. The Default button populates the field with the default port number of 57212, configured by the server.

**Tips**:

1. The default port is recommended unless there is a conflict with another application using that port.
2. Before changing the port setting, verify there is no conflict with the new port number.
3. The gateway service does not accept remote connections, so there should be no firewall implications associated with this port assignment.

In the Java area:

Use the latest installed version of the JRE and utilizes the newest 64-bit JRE installed on the system when the IoT Gateway starts.

To specify a specific JRE, de-select this option and enter the path to the JRE or use the Browse (...) button to locate the JRE.

**Tip**: If Use latest installed version of the JRE is selected and the Java version is updated on the machine, the gateway service automatically starts using the updated version the next time the gateway is started. If this option is disabled, the gateway service continues to use the specified version.

Advanced Settings... allow Java-specific settings to be used. These settings should only be changed if instructed by Technical Support.

In the REST Server area:

Click on the Manage Certificate… button to configure security certificate use for the REST server.

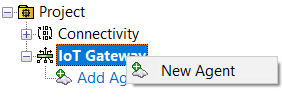
In the MQTT Agent area:

Click on the Manage Certificate… button to configure security certificate use for the MQTT agent.

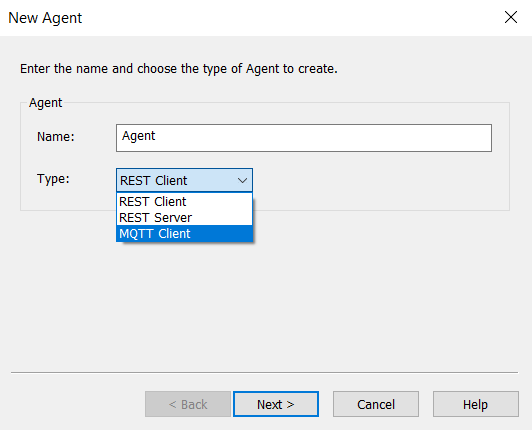
## **Configuring an Agent**

An agent configuration is required to begin publishing data to a third party endpoint. At least one agent needs to be configured with one active tag for the gateway service to start. Follow the steps below to configure a new agent.

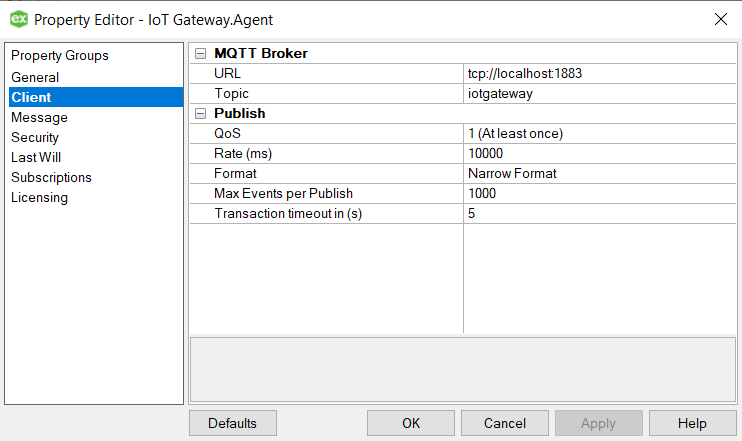
1. Click on the **Add Agent...** text or right-click any blank area in the agent pane and select **New Agent** from the pop up menu. Alternatively, click on the New Agent icon in the toolbar.



1. In the New Agent dialog, enter a name for the agent and select the type: MQTT Client, REST Client, or REST Server agent. Click **Next >**.



Once the agent type is selected as an MQTT client, the following properties must be defined.



**URL**: the IP address or URL and port of the endpoint for the agent connection. If the endpoint uses an SSL connection, adjust the URL to use “https://” or “ssl://”. It is the broker address.

**Note**: The URL field MUST be in the format "tcp://<host>:<port>" or "ssl://<host>:<port>" where <host> is the IP address or URL and <port> is the port of the end point being targeted. There should be no additional characters or "/" after the port of the end point.

**Topic**: the name used to filter or organize data published on the broker.

**QoS**: the MQTT setting for publishing data Quality of Service. Choices include: 0 (at most once), 1 (At least once), 2 (exactly once).

**Rate (ms)**: the frequency at which the agent pushes data to the endpoint. The range is from 10 to 99,999,990 milliseconds.

**Format**: select from the following options:

**Wide Format (every tag in every publish)**: produces an output that includes all enabled tags in the agent with every scan regardless of value or quality changes. This format guarantees a consistent data format on every publish. Wide format sends only the latest value for each tag and has no buffer. If a publish fails while using wide format, the next publish is the latest scanned values for each tag.

**Narrow Format**: produces output based on tags that have changed value or quality. This format buffers data and publishes all tag data changes to an endpoint.

**Max. events per**: adjusts the number of tag events the gateway packages in a single transmission when using narrow format.

**Transaction timeout**: set the time, in seconds, the server waits for a single transaction to complete. The range is from 1 to 60 seconds.

If needed, enter in the Client ID, Username, and Password for the broker being connected in the MQTT Client - Security dialog.

**Tip**: Confirm the MQTT connection is published to the broker and the broker is receiving. Check the Event Log for errors.

## **Agent Properties — Subscriptions**

The MQTT agent can be configured to subscribe to a topic on the broker, allowing other publishers to write to tags under that agent. When enabled, the agent listens to this topic with a QOS of 0 for properly formatted JSON to write to a tag.

Double-click on the agent name or right-click on the agent and select **Properties**. Then select **Subscriptions**. 

**Listen for Write Requests** Select Yes or No to allow agent to subscribe. The default setting is No.

**Topic**: Specify the topic to which the agent should subscribe. The default topic is iotgateway/write.

**Notes:**

* The MQTT agent checks the topic specified for JSON data in the proper format. Once the agent parses the data, it attempts to write the value to the specified tag.
* Only tags that have been added to the MQTT agent may be written.

Write formatting

To perform a write, the data needs to be in the following format:

[{"id": "Channel1.Device1.Tag1","v": 42},{"id": "Channel1.Device1.Tag2","v": 523}]

The “Channel1.Device1.Tag1” should be replaced by the tag to be written and “42” by the value to be written.

The example above shows a JSON array that should update both Tag1 and Tag2 when parsed by the MQTT agent.

Using Mosquitto\_sub.exe to update Tag1 from a DOS command line would look like:

mosquitto\_pub.exe -t iotgateway/write -m "[{\"id\": \"Channel1.Device1.Tag1\",\"v\": 42}]"

**Tip**: Any failures to update a tag using this method are posted to the Event Log.

**Note**: The MQTT subscription option does not check for user authorization against the User Manager or Security Policies Plug-In. Any valid JSON published to the configured topic will be written to the server. Configure the MQTT broker to verify that appropriate authentication is used at that level.

## **Standard Template Data Format**

The standard template pushes data in a JSON format via the REST and MQTT clients. The data structure for these agents looks like the following sample by default:

{"timestamp":1438011255230,

"values":

[

{"id":"Channel1.Device1.Tag1","v":250,"q":true,"t":1438011254668}

]

}

The components used in the above sample are defined as follows:

* timestamp = The time, in milliseconds, that the data was published to the endpoint since the UNIX epoch
* id = The unique name of the tag
* v = The value of the tag
* q = True means good quality update, false means bad (i.e. lost communications to the underlying device or invalid configuration)
* t = The time the tag data was sampled inf milliseconds since the UNIX epoch

The format of a 2 by 2 array is as follows:

{"timestamp":1438011255653,

"values":

[

{"id":"Channel1.Device1.Tag1","v":"[1,2][3,4]","q":true,"t":1438011254924}

]

}

**Note**: When writing an array, all opening and closing brackets, as well as commas between the array elements, must be included. All values for an array must be written at one time. A missing value in the array prevents the entire array from being written.

## **Paho.py**

This is the script for the connection from Python to Kepware. Any program can use this script to edit the tag inside of Kepware.

Install the necessary module:-

pip install paho-mqtt

The variables that can be changed are as below:

**Broker**: The URL to a managed or self hosted broker. It must be the same as in the IoT Gateway.

**Port**: The port of the broker.It must be the same as in the IoT Gateway.

**Topic**: The same topic as in the IoT Gateway topic

**Client\_id**: An identification for the connection. Can be put as any.

**Msgs**: The message that needs to be delivered to kepware. It must follow the Standard Template Data Format.

## **Main.py**

This is the script for a motion detection function using a webcam. When it detects motion, the script will send a boolean value of 1 to kepware via the paho.py script.

The necessary module:-

pip install opencv-python

pip install pandas

In Line 24, You can change the webcam code if you have multiple webcam.

In Line 61, the contour threshold can be changed here