Technical Note

KEPServerEX® and Microsoft® Azure IoT Edge

This Technical Note discusses how to configure KEPServerEX to communicate with Azure IoT Hub over MQTT **through** Azure IoT Edge. Once these steps are complete, add modules in IoT Edge (such as Azure Stream Analytics, Azure ML, or custom code) to "process" the data before sending it to the IoT Hub.

The instructions are similar to creating an IoT Hub direct connection, with the major additions being making sure the server can resolve the edge gateway name, making sure the server trusts the edge gateway certificate so that it can open a TLS connection, and pointing to the Edge as the MQTT server endpoint instead of IoT Hub directly.

1. Configure IoT Edge as a Transparent Gateway

Follow the instructions from Microsoft (https://docs.microsoft.com/en-us/azure/iot-edge/how-to-create-transparent-gateway-windows) or Linux (https://docs.microsoft.com/en-us/azure/iot-edge/how-to-create-transparent-gateway-linux).

Notes:

1. The self-signed certs generated via the scripts in the instructions above are meant for test scenarios. Use "real" certificates for production.
2. Record the IoT Edge gateway name (e.g. iotedgegw.local). This is the name in the hostname parameter in the config.yaml file for IoT Edge.

2. Establish a Trust Relationship

A trust relationship must exist between KEPServerEX and the Edge. To create a trust relationship, certificates must be exchanged and accepted from the Edge device to the KEPServerEX device. If a certificate purchased from a third party or trusted corporate certificate authority already trusted by KEPServerEX was used on the IoT Edge device, skip this step; otherwise, proceed:

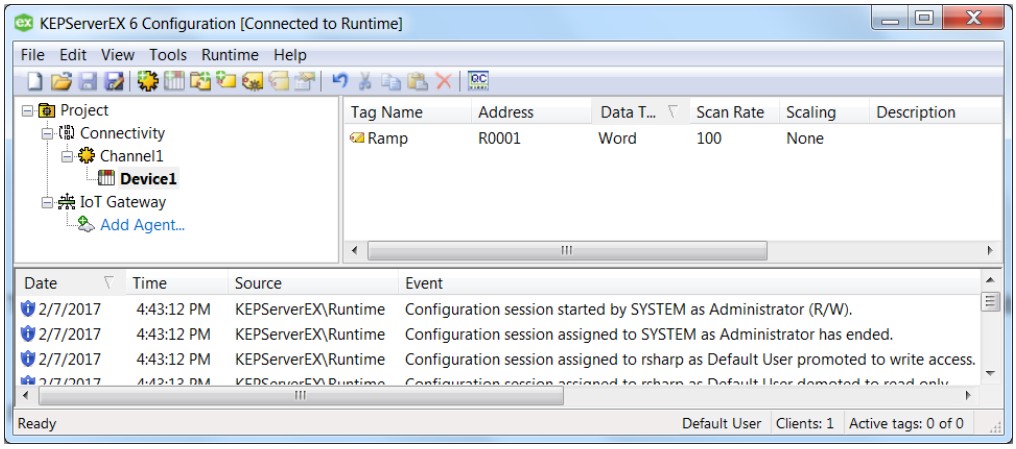
- In KEPServerEX, import the "root ca" certificate that was used to create the IoT Edge gateway (according to the Windows instructions at https://docs.microsoft.com/en-us/azure/iot-edge/how-to-create-transparent-gateway-windows#installation-on-the-downstream-device). This ensures that KEPServerEX trusts the certificate used by the IoT Edge server to create a TLS-protected MQTT connection to IoT Edge. If KEPServerEX doesn't trust the Edge certificate, it refuses the connection.

3. Ensure KEPServerEx Can Resolve Edge Device IP Address

If there is not a real DNS entry for the IoT Edge device's hostname (e.g. iotedgegw.local), add an entry to the Kepware server's \windows\system32\drivers\etc\hosts file so that the name can resolve. Test this by running 'ping <hostname>' from a command prompt.

4. Create IoT Device and Get SAS Token

1. Open a KEPServerEX® instance with the IoT Gateway advanced plug-in. In this example, one channel and device are configured with the Simulator driver, and there is one tag that ramps up on scan.

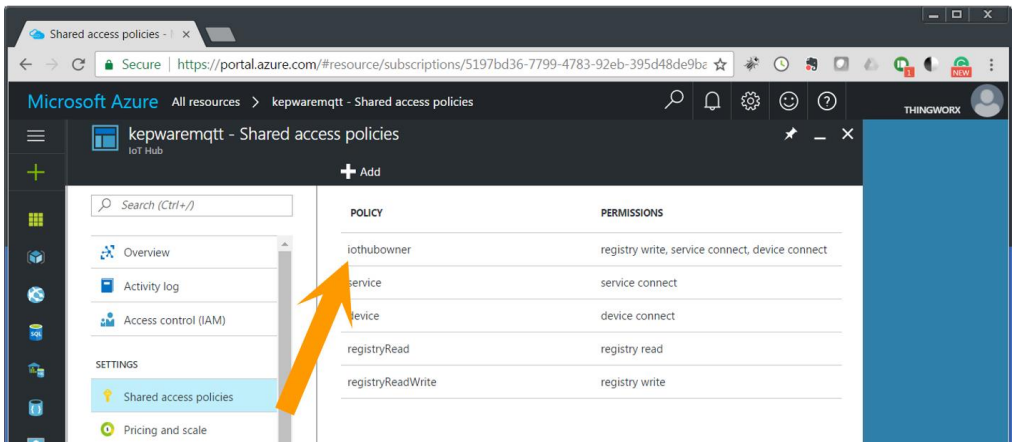


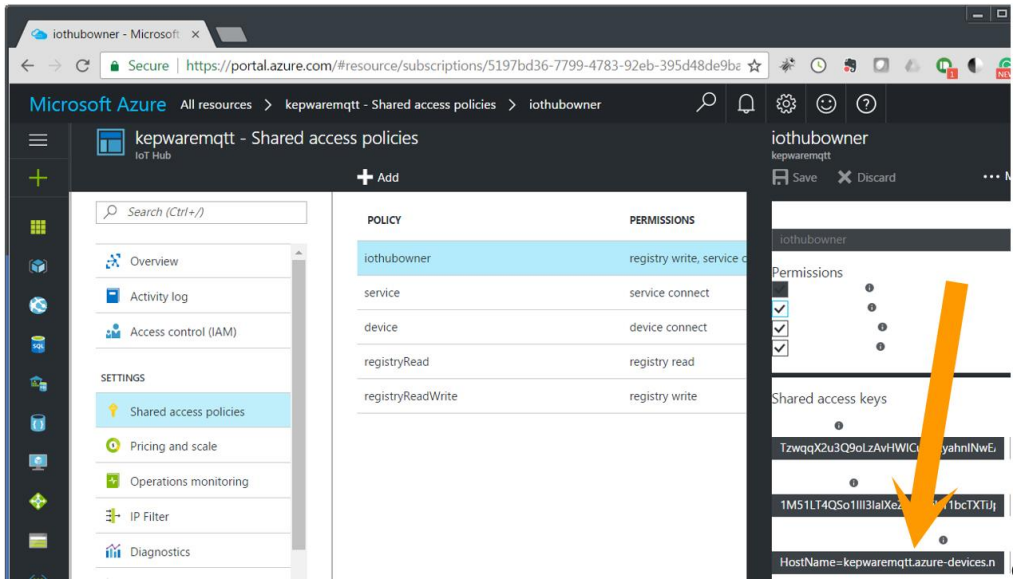
2. Download and install the Device Explorer tool at: https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer.

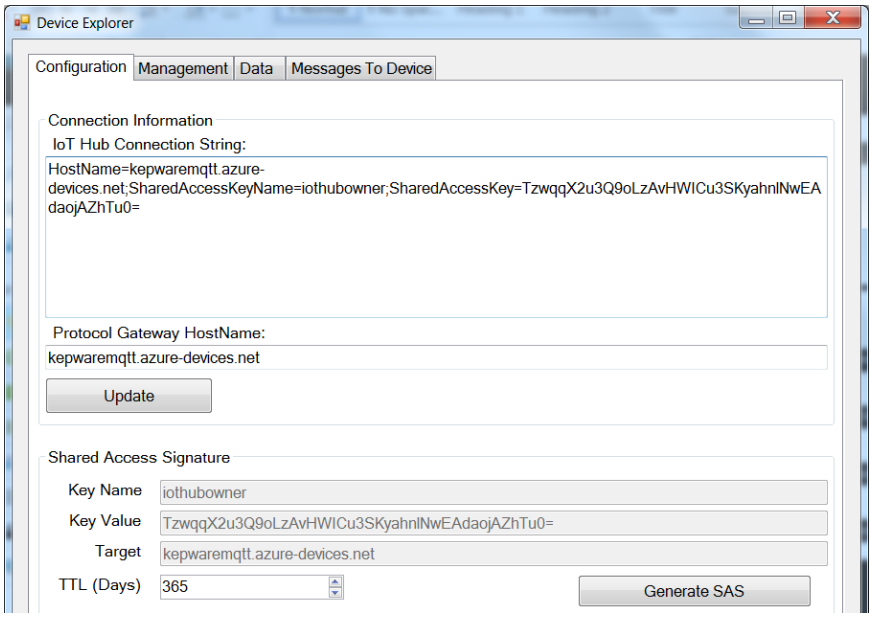
3. Navigate to the IoT Hub in the Azure portal.

4. Click **Shared access policies** under Settings in the IoT Hub and then select the appropriate policy. In this example, using the "iothubowner" policy, a shared access key is generated as well as a **Connection string**. The **Connection string** for 'iothubowner' user includes both the Shared Access Key and the Hostname URL.

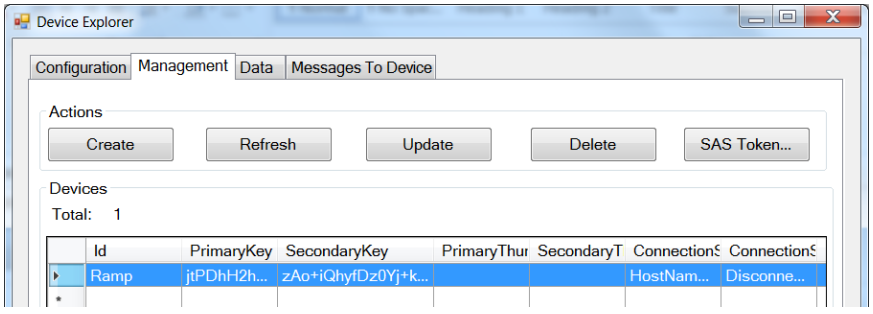
5. Copy the **Connection string** and paste it into the **IoT Hub Connection String** field within the Device Explorer application downloaded in a previous step.



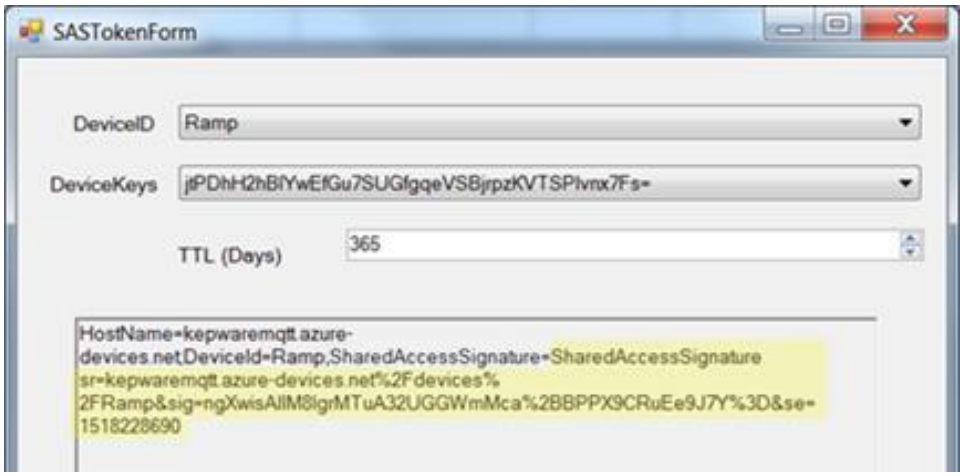




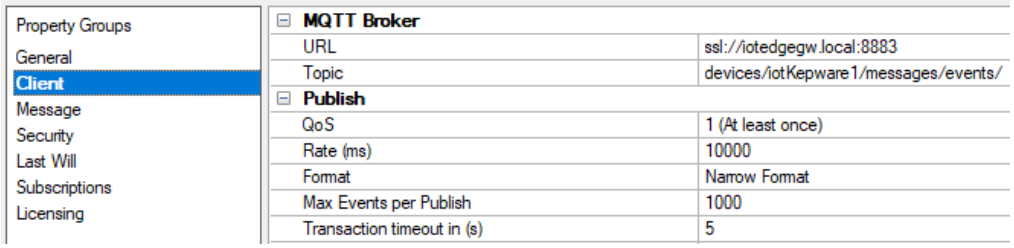
6. In Device Explorer, click **Update**, then create a device by accessing the **Management** tab.
7. Click **Create** and give the device a unique name.



8. Click **SAS Token…** to generate the SAS token in Device Explorer. Part of the string from this dialog needs to be copied (starting with everything AFTER "SharedAccessSignature=") and will be used as the **SAS Key** later. This example will copy the highlighted text.



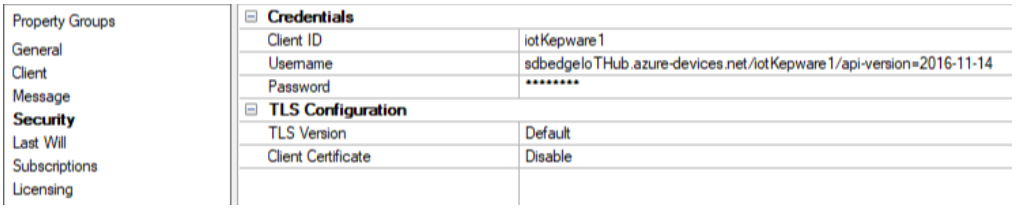
9. Record the following pieces of information:
   - **<IoT hub name>** (Example: sdbedgeIoTHub.azure-devices.net)
   - **<deviceID>** (Example: iotKepware1)
   - **<SAS token>** (Example: SharedAccessSignature sr=sdbedgeIoTHub.azure-devices.net%2Fdevices%2FiotKepware1&sig=HS%2FfyEVuCFxem5JYZJ%2BzKKIQZyp1SqfcSVQDzSlgtCg%3D&se=1562872697)
   - **<IoT Edge Gateway host name>** (Example: iotedgegw.local)
10. In KEPServerEX, add an IOT agent. Use the following formats for the indicated properties:



    a. URL format: ssl://**< IoT Edge Gateway host name >**:8883 (Example: ssl://iotedgegw.local:8883). This should point to IoT Edge as the endpoint, not the IoT Hub directly.

    b. Topic format: devices/**<deviceID>**/messages/events/ (Example: devices/iotKepware1/messages/events/). This must include the forward slashes, including the ending /.

11. Create security credentials. The expected formats are as follows:

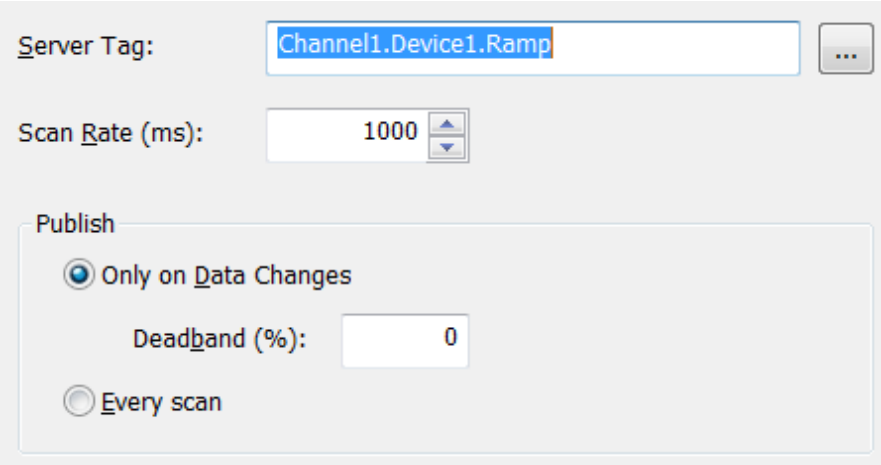


    a. Client ID: **<deviceID>**

b. Username: **<IoT Hub name>/<deviceID>**/api-version=2016-11-14. (Example: sdbedgeIoTHub.azure-devices.net/iotKepware1/api-version=2016-11-14). This must include the version.

c. Password: **<SAS Key>**

12. Next, add an IoT item.



13. Verify that the MQTT Client agent has connected to the Azure IoT Hub by observing the event log has an entry similar to the following:

Date	Time	Level	Source	Event
2/10/2017	2:41:35 PM	Information	KEPServerEX\Runtime	MQTT agent 'Ramp_2_Cloud' is connected to broker 'ssl://kepwaremqtt.azure-devices.net:8883'

5. Start KEPServerEX IoT Agent

1. Verify a successful connection to the local IoT edge device instead of IoT Hub.
2. On the edge device, run the “docker logs -f edgeHub” command.
3. Once KEPServerEX IoT Gateway starts, verify a successful connection in the Edge Hub logs, similar to the one below (CTRL-C exits):

```
2018-07-12 17:05:00.939 +00:00 [INF] - Attempting to connect to IoT Hub for client iotKepware1 via AMQP...
2018-07-12 17:05:00.945 +00:00 [INF] - Connected to IoT Hub for client iotKepware1 via AMQP, with client operation timeout 60000.
2018-07-12 17:05:00.945 +00:00 [INF] - Closing receiver for device iotKepware1
2018-07-12 17:05:00.945 +00:00 [INF] - Closed cloud proxy for device iotKepware1
2018-07-12 17:05:00.945 +00:00 [INF] - New cloud connection created for device iotKepware1
2018-07-12 17:05:00.953 +00:00 [INF] - New token requested by client iotKepware1, but using existing token as it is usable.
2018-07-12 17:05:01.863 +00:00 [INF] - Successfully generated identity for clientId iotKepware1 and username sdbedgeIoTHub.azure-devices.net/iotKepware1/api-version=2016-11-14
2018-07-12 17:05:01.867 +00:00 [INF] - ClientAuthenticated, iotKepware1, 280edb2d
2018-07-12 17:05:01.868 +00:00 [INF] - New device connection for device iotKepware1
2018-07-12 17:05:01.870 +00:00 [INF] - Bind device proxy for device iotKepware1
2018-07-12 17:05:01.870 +00:00 [INF] - Binding message channel for device Id iotKepware1
```

4. After a few minutes, verify data starts flowing up to IoT Hub by monitoring the device in the Data tab of the Device Explorer application.

