



Azure IoT Hub Client SL

With the “Azure IoT Hub Client SL” library, CODESYS controllers can exchange messages with the “Azure IoT Hub” cloud service from Microsoft.

Product description

The “Azure IOT Hub” cloud service from Microsoft directly links IoT devices. (For more detailed information, refer to <https://azure.microsoft.com/en-us/services/iot-hub/>). The “Azure IoT Hub Client SL” library provides function blocks for sending and receiving messages. A sample project demonstrates how to use the library.

The library contains separate function blocks for communication via HTTPS and MQTT. The library supports the following functions:

- Send “Device to Cloud (D2C)” messages (telemetry data)
- Receive “Cloud to Device (C2D)” messages
- Read the device twin
- Update the device twin (desired properties only)
- Subscribe Device Twin (desired properties, MQTT only)
- Direct method call (Cloud -> Device, MQTT only)
- Creation of a new azure SaS token (Shared Access Signature)

The sample project “Azure IoT Client SL Example.project” demonstrates how to use the corresponding function blocks.

Getting started

1. Set up Azure IoT Hub and create devices in IoT Hub

see <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal>
or <https://docs.microsoft.com/en-us/azure/iot-hub/tutorial-connectivity>

2. Generate shared access signatures (SaS tokens)

The authentication of a device is done via ‘Shared Access Signatures’ (SaS-Token). For access via MQTT, a SaS token based on the device policy ‘device’ is required. For access via HTTPS, an SaS token based on the device policy ‘device’ and an SaS token based on the policy ‘iothubowner’ are required. The SaS token can be generated e.g. via the Azure Cloud Shell (Azure Portal), via the tool ‘Azure IoT Explorer’, or via function in library `GenerateSasToken()`. The function `GenerateSasToken` requires 4 input parameters (url-adress, primary key, policy name, expiry) and returns SaS-Token as WSTRING(255).



Start cloud shell via Azure Portal

Command to generate a SaS token 'device':

```
az iot hub generate-sas-token -d YOUR_DEVICE_ID -n YOUR_IOT_HUB --du
DURATION_IN_SECONDS --policy device
```

Command to generate a SaS token 'iothubowner':

```
az iot hub generate-sas-token -n YOUR_IOT_HUB --du DURATION_IN_SECONDS
```

The screenshot shows the Azure IoT Explorer interface. On the left is a sidebar with navigation options like 'Pricing and scale', 'Networking', 'Certificates', 'Built-in endpoints', 'Failover', 'Properties', 'Locks', 'Explorers', 'Query explorer', 'IoT devices', 'Automatic Device Management', 'IoT Edge', and 'IoT device configuration'. The main area displays a table titled 'Query devices' with columns: Device ID, Status, Last Status Update (UTC), Authentication Type, and Cloud to Device Message Count. One device is listed with a redacted ID, Status 'Enabled', Last Status Update '--', Authentication Type 'Sas', and Message Count '0'. Above the table is a query builder with fields, operators, and values, and a 'Query devices' button. Below the table is a 'Switch to query editor' link.

Generate SaS-Token

For more information on 'Cloud Shell' and 'Azure CLI', click here:

<https://docs.microsoft.com/en-us/cli/azure/iot/hub?view=azure-cli-latest>

Alternatively, the 'Azure IoT Explorer' tool can be used to generate a SaS token (only for policy 'device').

see <https://github.com/Azure/azure-iot-explorer>

Download: <https://github.com/Azure/azure-iot-explorer/releases>

3. Setting the name of the IoT hub, device ID, and SaS token in the sample project

sSubDomainName: Name of the Azure IoT Hubs without 'azure-devices.net'(see 1.)

sDeviceId: Device ID (see 1.)

wsDeviceSaS: SaS token of the device (MQTT, HTTP) (see 2.)

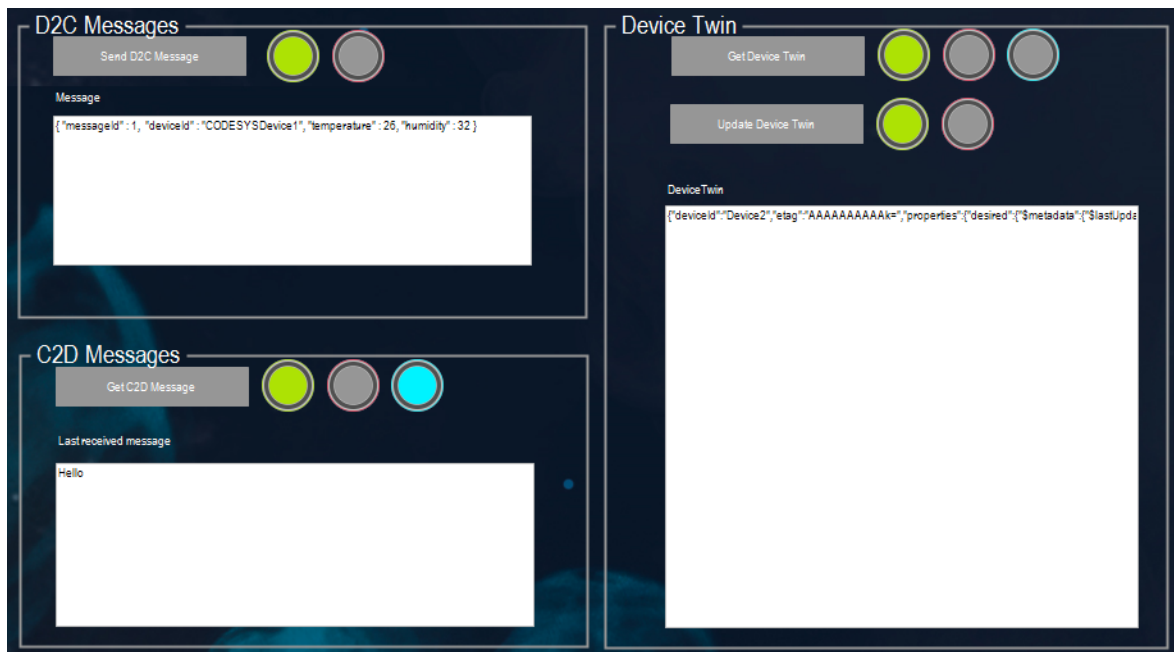
wsIoTHubOwnerSaS: SaS token of the IoT Hub owner (HTTP) (see 2.)

4. Downloading the project to the controller and starting it

Application AzureHTTPTDemo

The following functions can be executed from the visualization:

- Send D2C message
- Get C2D message
- Get device twin
- Update device twin



Visualization of the HTTPS sample project

Applikation AzureMQTTDemo

The following functions can be executed from the visualization:

- Connect to IoT Hub via MQTT
- Send D2C message
- Get C2D message
- Get device twin
- Update device twin
- Direct method call
- Subscribe device twin (desired properties)

The screenshot displays the Azure IoT Explorer interface with a dark blue background. At the top, the 'Connection Settings' section includes input fields for 'Domain name' (IoT Hub URL), 'Device Id' (Device2), and 'SaS Token' (IoT Hub URL), along with two circular status icons. Below this, the interface is divided into six panels:

- D2C Messages:** Features a 'Send D2C Message' button and a text area containing a JSON message: `{ "messageId": 1, "deviceId": "CODESYSDevice1", "temperature": 25, "humidity": 32 }`.
- Get Device Twin:** Includes a 'Get Device Twin' button and a text area showing a JSON object: `{ "desired": { "Version": 9 }, "reported": { "temperature": 24, "Version": 82 } }`.
- C2D Messages:** Has a 'Last received message' label and a text area displaying 'Hello..'. A small blue dot is visible in the top right corner of this panel.
- Update Device Twin:** Contains an 'Update Device Twin' button and a text area for 'DeviceTwin Properties' with the JSON `{ "temperature": 24 }`.
- Direct method call:** Displays a large yellow circle with a grey border.
- Subscribe Device Twin (desired):** Includes a 'Last received message' label and an empty text area.

Visualization of the MQTT sample project

5. Sending and receiving messages with the tool 'Azure IoT Explorer'

With the tool 'Azure IoT Explorer' (download see 2.) messages can be displayed and sent.

The screenshot displays the Azure IoT Explorer (preview) application. The top navigation bar includes the title 'Azure IoT Explorer (preview)', a menu (File, Edit, View, Window, Help), and buttons for Notifications and Settings. The breadcrumb path is 'Home > rliothub > Devices > rldevice > Telemetry'.

The left sidebar contains a menu with the following items: Device identity, Device twin, Telemetry (selected), Direct method, Cloud-to-device ..., Module identities, and IoT Plug and Play....

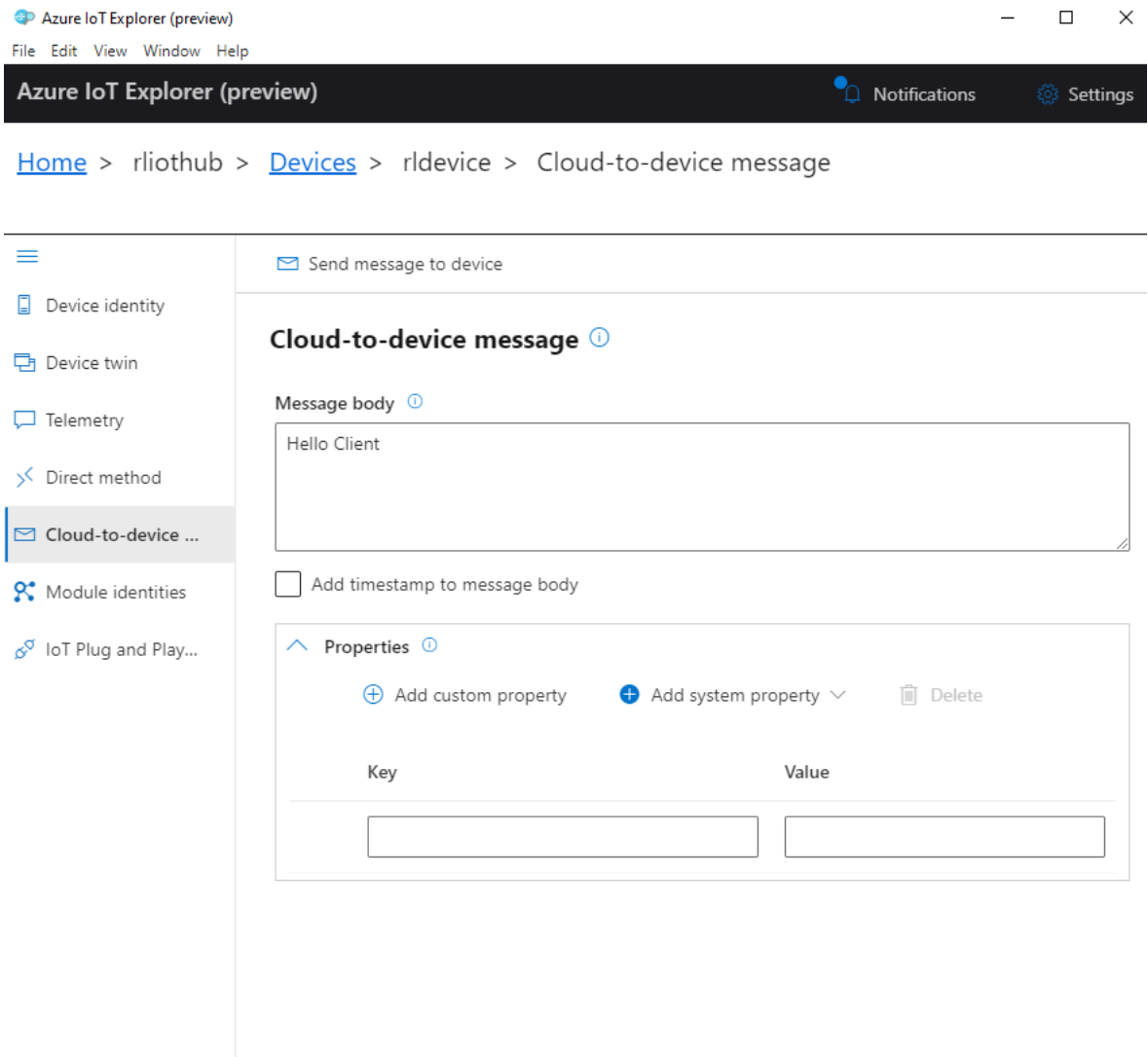
The main content area is titled 'Telemetry' and includes the following controls:

- Buttons: Stop, Show system properties, Clear events, and Simulate a device.
- Consumer group: \$Default
- Specify enqueue time: No (toggle)
- Use built-in event hub: Yes (toggle)
- Status: Receiving events...

The Telemetry section displays a message received on 'Fri Jun 04 2021 09:45:30 GMT+0200 (Mittleuropäische Sommerzeit)'. The message body is a JSON object:

```
{  "body": {    "messageId": 1,    "deviceId": "CODESYSDevice1",    "temperature": 26,    "humidity": 32  },  "enqueuedTime": "Fri Jun 04 2021 09:45:30 GMT+0200 (Mittleuropäische Sommerzeit)",  "properties": {}}
```

Azure IoT Explorer: Monitoring of D2C messages



Azure IoT Explorer: Sending C2D messages

Azure IoT Explorer (preview)

File Edit View Window Help

Azure IoT Explorer (preview) Notifications Settings

[Home](#) > [rliothub](#) > [Devices](#) > [rldevice](#) > Direct method

Device identity

Device twin

Telemetry

Direct method

Cloud-to-device ...

Module identities

IoT Plug and Play...

Invoke method

Direct method

Method name *

Switch

Payload

"1"

Connection timeout in seconds

10

Response timeout in seconds

10

Azure IoT Explorer: Remote method call

Options

In the Azure environment, messages are usually sent in JSON format. The CODESYS library "JSON Utilities SL" is ideal for parsing and generating JSON data.

General information

Supplier:

CODESYS GmbH
Memminger Strasse 151
87439 Kempten
Germany

Support:

Technical support is not included with this product. To receive technical support, please purchase a CODESYS Support Ticket.

<https://support.codesys.com>

Item:

Azure IoT Hub Client SL

Item number:

Sales/Source of supply:

CODESYS Store
<https://store.codesys.com>

Included in delivery:

CODESYS package

System requirements and restrictions

Programming system	CODESYS Development System V3.5.15.0 or later
Runtime system	CODESYS Control V3.5.15.0 or later
Supported platforms and devices	Note: Use the "Device Reader" project for locating the functions supported by the PLC. The "Device Reader" project is available in the CODESYS Store free of charge.
Additional requirements	Microsoft Azure account with Azure IoT Hub service; Device Explorer
Restrictions	•
Licensing	License activation optional on CODESYS Key or Soft Key (free of charge component of CODESYS Controls). Licensing via Soft Key is strictly linked to hardware. Note: Without a license the software runs for 30 minutes in demo mode.
Required accessories	-

Note: Technical specifications are subject to change. Errors and omissions excepted. The content of the current online version of this document applies.

Creation date: 2023-08-21