Microsoft Azure IoT Hub API

V2.10

Table of contents

Introduction

This application note describes how to use the functions in the `az_iot.inc` include file to communicate with the Azure IoT Hub.
To generate SAS tokens locally on the device there are two options. Either runtime V1.95.00 or later should be used, or the included extension module `mod_azi.rmx` must be loaded. LX devices must always use runtime version 1.95.00 or later.

The IoT Hub API has support for two kinds of credentials:

Client certificates

Client certificates are normal X.509 certificates that must be installed on the device. The IoT Hub must either be configured with the CA certificate or with the thumbprint for the specific certificate.

SAS tokens

The SAS tokens are time limited tokens that are generated based on a Shared Access Key.
The SAS tokens can either be generated remotely and then transferred to the device or they can be generated locally on the device using the function aziCreateSAS.

Azure IoT Hub

IoT Hub is a fully managed service that enables reliable and secure bidirectional communications between millions of IoT devices and a solution back end. [1]
The `az_iot.inc` include file provides functions to monitor and manage the device, using the following features:

Device-to-cloud communications

The API provides two different ways of sending data from the device to the cloud; device-to-cloud messages and by reporting updated device twin properties.

See https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-d2c-guidance for more information about device-to-cloud communication.

Device-to-cloud messages

Device-to-cloud messages can be sent using the `aziPublish` function, and can be used to send a payload and some properties to the IoT hub for further processing.
See https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messages-d2c for more information about device-to-cloud messages.

1 https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-azure-iot

Reported Device twin properties

The function aziTwinReportProperty can be used to send updated device twin properties to the cloud.
This feature is only available with the standard tier of IoT Hub.

Cloud-to-device communications

The API provides three different ways of performing cloud-to-device communication; cloud-to-device messages, device twins and direct method calls.

Note that these features are only available with the standard tier of IoT Hub.

See https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-c2d-guidance for more information about cloud-to-device communications.

Cloud-to-device messages

Device-to-cloud messages can be received with `aziReceiveMessage` and up to 10 properties can be read using `aziGetProperty`.

Device Twins

Device twins are JSON documents that store device state information (metadata, configurations, and conditions).[2]

The device twin properties can be managed using the aziTwinRequest and aziTwinReportProperty functions as well as handling the Update Device Twin Property message from `aziReceiveMessage`.

The payload for these functions are JSON objects that contain parts of the device twin, which can then be updated when the properties change or should be changed.

Direct methods

Direct methods are a way of invoking code on the device from the IoT hub. This can e.g. be used to request the device to reset or to perform a specific task. When a direct method is invoked, the Direct method request message is received and `aziGetMethodRequest` can be called to get the name of the method, while `aziMethodReply` should be used to send the reply.

2 https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-device-twins

Application structure

This section describes how the VPL application for communicating with the IoT Hub could be structured. The `az_iot.inc` include file uses the MQTT functions for communicating with the IoT Hub, so it can not be used at the same time.

The device must have been created on the IoT Hub and the credentials must be available on the device.

A connection to a network that can access the IoT Hub must be established and the connection can then be created using `aziOpenSAS` or `aziOpenCert`.

It is then necessary to wait for the connection to be established to the IoT Hub, using `aziConnected` and `aziStatus` to monitor the connection.

Once the connection has been established, `aziSubscribe` must be called to make the device receive the needed events.

The main loop for handling the communication must call `aziWaitEvent` to check for new messages.
If a messages has been received, the `aziReceiveMessage` function block must be called to read the message and based on the type of message, different functions can be called to held handle the message.

Functions

The following functions are used to communicate with the IoT Hub and can be found in the `az_iot.inc` include file.

aziOpenSAS()

Open the connection to the IoT hub using SAS Tokens.
The tokens have the format `"SharedAccessSignature sig={signature-string}&se={expiry}&sr={URL-encoded-resourceURI}"`
The tokens can either be generated remotely and stored on the device or they can be created locally using the aziCreateSAS function.

Input:

- deviceId - The device identifier
- password - The security token.
- server - The hostname of the IoT hub server.
- iface - The network interface to use.

Returns:

- 0 - Success.
- -1 - No available connections.
- -2 - Invalid parameter
- -3 - Connection is already in use.

aziOpenCert()

Open the connection to the IoT hub, using X.509 certificates

Input:

- deviceId - The device identifier
- cert - The name of the client certificate
- cert_pass - The password for the private key in the client certificate
- server - The hostname of the IoT hub server.
- iface - The network interface to use.

Returns:

- 0 - Success.
- -1 - No available connections.
- -2 - Invalid parameter
- -3 - Connection is already in use.

aziClose()

Close the connection to the IoT hub

Input:
- None.

Returns:
- None.

aziSubscribe()

Subscribe to the events on the connected IoT hub.

Input:
- None.

Returns:
- 0 - Success.
- 1 - Invalid connection.
- 2 - Not connected to server.
- 3 - Invalid parameter

aziConnected()

Tells if the connection to the IoT hub is established.

Input:
- None.

Returns:
- TRUE - Device is connected to IoT hub.
- FALSE - Device is not connected to IoT hub.

aziStatus()

Returns the status of the connection.

Input:

- None.

Returns:

- 0 - Connected to the IoT Hub.
- 1 - Invalid connection.
- 2 - Not connected to network.
- 3 - IoT hub server not found.
- 4 - No reply from IoT hub
- 5 - Connection rejected, unacceptable protocol version.
- 6 - Connection rejected, client ID rejected.
- 7 - Connection rejected, server unavailable.
- 8 - Connection rejected, bad user-name or password.
- 9 - Connection rejected, not authorized.
- 20 - Secure connection failed.
- 21 - Secure connection require client certificate.
- 22 - Certificate verifications failed.
- 23 - Client certificate is incomplete.

aziWaitEvent()

Waits for a new message, blocking the thread.

Input:

- timeout - The number if seconds to wait before timing out. Use 0 to return immediately, use -1 to wait forever.

Returns:

- 1 - A messages is received. Use aziReceiveMessage to read it.
- 0 - Timeout.
- -1 - No connection.

aziReceiveMessage()

Function block that receives messages from the IoT Hub.

Input:
- data - Address of a buffer to store the data of the message in.
- maxsize - The size of the buffer. Any message data after this limit is lost.

Output:
- ready - True if a message has been received.
- size - The size of the read data.
- type - The type of the message:
  - -1 - Unknown message.
  - 1 - Cloud to device message. Use aziGetProperty to read the properties of the message. The data contains the contents of the message.
  - 2 - Direct method request. Use aziGetMethodRequest to read the name of the requested method and use aziMethodReply to reply to the request. The data contains the parameters for the method as JSON.
  - 3 - Device twin response. The data contains the entire device twin. Use aziTwinRequestStatus to get the status.
  - 4 - Device twin report property reply. Use aziTwinReportPropertyStatus to get the status.
  - 5 - Update Device Twin property. Use aziTwinDesiredVersion to get the desired version number. The data contains JSON with the desired properties.

aziGetProperty()

Retrieves a property that was received with a cloud-to-device message.
Note that if the names and values are too long, it is possible that not all the sent properties are available.

Input:
- index - The index of the property to read(1..10)

Output:
- key - The name of the property
- value - The value of the property

Returns:
- 0 - Success.
- -1 - Invalid index, there is no property at the given index.

aziPublish()

Sends a message to the connected IoT hub. The message contains two payloads; a data buffer and optionally a number of properties. The data buffer will typically consist of JSON formatted data, but can use other formats.
The properties can both be predefined system properties and custom application properties.
See https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messages-construct for more information about the messages and the different property types.

Input:

- properties - The optional properties to send with the message.
  URL-encoded string, e.g. `"temperature=30&inputs=62"`
- data - Address of a buffer with the data to send.
- size - Size of the data to send.

Returns:

0 - Success.

1 - Invalid connection.

2 - Not connected to server.

3 - Invalid parameter

4 - A publish is already in progress

5 - Message transmission interrupted due to connection loss. The message will be transmitted again when connection is re-established.

aziGetMethodRequest()

Retrieves the name of the requested method.

Output:

- method - The name of the called method.

Returns:

- 0 - Success.
- -1 - No unread method request available.

aziMethodReply()

Sends a reply to a method request.

Input:

- status - The status code for the reply.
- data - Address of buffer with the payload for the reply.
- size - The size of the payload.

Returns:

0 - Success.

1 - Invalid connection.

2 - Not connected to server.

3 - Invalid parameter

4 - A publish is already in progress

5 - Message transmission interrupted due to connection loss. The message will be transmitted again when connection is re-established.

6 - No method available to reply to.

aziTwinRequest()

Request the device twin properties from the HUB.
The properties will be available as the data in a Twin response from aziReceiveMessage while the status can be read with aziTwinRequestStatus.

Returns:

0 - Success.

1 - Invalid connection.

2 - Not connected to server.

3 - Invalid parameter

4 - A publish is already in progress

5 - Message transmission interrupted due to connection loss. The message will be transmitted again when connection is re-established.

aziTwinRequestStatus()

Read the status of a device twin properties request.

Returns:

- - The status
- -1 - No status available

aziTwinReportProperty()

Changes reported twin properties. The status of the request can be read with aziTwinReportPropertyStatus.

Input:

- data - Address of buffer with the JSON describing the change.
- size - The size of the data.

Returns:

- 0 - Success.
- 1 - Invalid connection.
- 2 - Not connected to server.
- 3 - Invalid parameter
- 4 - A publish is already in progress
- 5 - Message transmission interrupted due to connection loss. The message will be transmitted again when connection is re-established.

aziTwinReportPropertyStatus()

Read the status of a device twin report property request.

Returns:

- - The status
- -1 - No status available

aziTwinDesiredVersion()

Read the desired version number of the twin property change request received from the IoT Hub.

Returns:

- The desired version or an empty string if none is available.

aziLoadModule()

Load the IoT Hub Extension module.
This is required to use the aziCreateSAS function on NX devices with a runtime version earlier than V1.95.00.

Returns:

0	- Success.
1	- Failed to load the module. Make sure that the `mod_azi.rmx` extension module is included in the project.

aziCreateSAS()

Create a SAS token to use for login to server.
The aziLoadModule function must be called before this function when using a runtime earlier than V1.95.00.

Input:

- lifetime - The number of seconds the token will be valid. The default is 24 hours
- tz_offset - The time zone offset from UTC in seconds.
- host - The hostname of the IoT hub server.
- DeviceId - The device identifier
- sharedAccessKey - The base64 encoded shared access key to sign the token with.

Returns:

- The generated SAS token or an empty string if an error has happened.