Getting Started with Azure IoT Hub on Talaria TWO Platform

Getting Started Guide

InnoPhase, Inc.
6815 Flanders Drive
San Diego, CA 92121
innophaseinc.com

Platform	InnoPhase Talaria TWO SDK
Device	Talaria TWO Evaluation Kit EVB-A (INP3010, INP3011, INP3012, INP3013)
Language	C

Contents

1 Document Information

1.1 Naming Conventions

Talaria TWO	Family of devices using the InnoPhase ultra-low power wireless technology, including the Talaria TWO SoC and Talaria TWO Modules
Talaria TWO SoC	InnoPhase custom wireless platform (INP2045)
Talaria TWO Modules	Modules integrating the Talaria TWO SoC (INP1010, INP1011, INP1012, INP1013)
Talaria TWO EVB-A	Evaluation boards containing the Talaria TWO Modules (INP3010, INP3011, INP3012, INP3013)

1.2 Glossary

API	Application Programming Interface
ELF	Executable and Linkable Format
EVB	Evaluation Board
GPIO	General Purpose Input/Output
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
SDK	Software Development Kit

1.3 Revision History (Version, Date, Description of change)

Version	Date	Description of Change
1.1	30/07/2021	First Version

Table 1: Revision History

2 Introduction

This document describes how to connect 'InnoPhase Talaria TWO INP301x EVB-A' devices with Azure IoT Hub using Talaria TWO SDK. This multi-step process includes:

- Configuring Azure IoT Hub
- Registering your IoT device
- Provisioning your devices on Device Provisioning service
- Build and deploy Azure IoT SDK on device

2.1 Introduction and features of Talaria TWO Family of Devices

The INP3010/3011/3012/3013 EVB-A's are available for evaluating the performance and capability of the Talaria TWO INP1010/1011/1012/1013 modules.

The kits use InnoPhase's award-winning Talaria TWO Multi-Protocol Platform with ultra-low power Wi-Fi + BLE5 for wireless data transfer, an embedded Arm Cortex-M3 for system control and user applications plus advanced security features for device safeguards.

The kits include an Arduino UNO format baseboard with a Talaria TWO module attached and a different antenna option per kit.

The EVB-A can be used in stand-alone mode or attached to an Arduino UNO compatible host or shield board. The baseboard has all module GPIOs accessible through either an internal 20-pin header or the Arduino connectors. Power is supplied from USB, host Arduino board or battery connector.

Also mounted on the baseboard are environmental sensors for capturing temperature, humidity, pressure and light. It is ideal platform for developing exciting new battery-based, cloud connected products such as smart locks, smart sensors, or security and health monitoring devices.

Product brief can be found in below link

https://innophaseinc.com/wp-content/uploads/modules/INP3010_INP3011-EVB-A-Product-Brief.pdf

For more details, please visit https://innophaseinc.com/talaria-two-modules/

3 Step 1: Prerequisites

You should have the following items ready before beginning the process:

3.1 Prepare the Development Environment (IDE, Toolchain, SDK)

'Talaria TWO Software Development Kit' is used for developing applications on Talaria TWO Platform.

Talaria TWO SDK is available through 'InnoPhase Customer Portal Access' and is available after portal registration, Mutual Non-Disclosure Agreement (MNDA) and Development Tools License Agreement (DTLA).

For detailed information on registering the getting the SDK access, please use the following link:

https://innophaseinc.com/portal/customer-registration/

The Talaria TWO SDK comes packaged with -

- User Guides for Development Environment Setup on a Host PC
- SDK API Reference Manual
- Application Notes
- Reference Applications and Solution Ready Applications
- Several Example Applications
- Documents for the user to start the development targeting different use-cases

A comprehensive User Guide *'T2-UG001-Talaria TWO SDK Environment Set-up.pdf'* is available covering how to setup the development environment for using Talaria TWO SDK on an Ubuntu VirtualBox based environment with a Windows 10 host.

This document details about installing the toolchain and necessary software packages required for the development, CLI commands for building target executables, programming the target and debugging of the application.

Talaria TWO SDK also supports the development using an Eclipse based IDE in Windows OS based PC. The details of setting up the development environment in Windows OS using the IDE is provided in User Guide *'T2-UG002 -Eclipse Set-up in Windows.pdf'*.

'T2-RM001-Talaria TWO SDK API Reference Guide.pdf' gives details about all the SDK API's available to the user.

All these tools and documents are available through the customer portal after registration through the customer portal link provided in this section.

3.2 Setup Your IoT Hub

Refer to Azure IoT Hub user guide to create and setup Azure IoT Hub using the portal. This can be found at the link below:

https://docs.microsoft.com/azure/iot-hub/iot-hub-create-through-portal

Creating and registering a new device identity using the portal is also explained in the same link.

Device-specific 'connection string' from this step will be used in few of the Sample Applications code.

3.3 Setup Device Provisioning Service (DPS) to Provision your device to Hub

The document for creating a new 'Device Provisioning Service' (DPS) and linking it to the IoT Hub we created in previous step, can be found at the link below:

https://docs.microsoft.com/en-us/azure/iot-dps/quick-setup-auto-provision

The Device Provisioning Service (DPS) 'Scope ID' and enrollment entity's 'Registration ID' will be used to run Device Provisioning Service (DPS) Sample Application.

Device Provisioning Service (DPS) Sample Application with both 'Symmetric Key' based attestation and 'X.509 CA' based attestation, are provided.

For 'Symmetric Key' based attestation, the enrollment entity's 'Symmetric Key' is also used for running the Sample Application.

For 'X.509 CA' based attestation, further instructions to generate the certificates and setup the enrollment are provided in the details with the Device Provisioning Service (DPS) Sample Application's document. These steps include uploading the rootCA cert to IoT Hub, performing proof of possession, for X.509 cert and creating enrollment entity on Device Provisioning Service (DPS) Portal. The link to this document is provided in one of the sections below.

3.4 Acquire a Talaria TWO INP301x EVB-A Development Kit

The Talaria TWO INP301x Dev Kits can be procured from distributor's links provided below:

https://www.mouser.com/manufacturer/innophase/

https://www.richardsonrfpd.com/Products/Search?searchBox=innophase&instockonly=false

4 Step 2: Prepare your Device

4.1 Setting up Talaria TWO INP301x EVB-A Board

A User Guide for setting up Talaria TWO EVB-A can be found in the link below:

https://innophaseinc.com/wp-content/uploads/modules/User-Guide-for-Talaria-TWO-EVB-A-Evaluation-Board.pdf

This has all the details needed for a successful setup, including description of components, power supply requirements, details of jumpers and the driver needed etc.

4.2 Tool for Programming the Device

Talaria TWO Download Tool is used for programming the EVB-A and using Debug Console. This tool is available for Windows and Linux platforms.

User Guide for this tool can be found in link below:

https://innophaseinc.com/wp-content/uploads/modules/Talaria-TWO-Download-Tool-User-Guide.pdf

This tool can be downloaded from the link below:

https://innophaseinc.com/talaria-two-modules#eval-software

The Download Tool is found in the following folder in the Evaluation Software download from above link:

I-CUBE-T2-STW.zip\STM32CubeExpansion_T2-HostAPI-lib_V1.0\Utilities\PC_Software\TalariaTwo_DownloadTool\Tool_GUI

5 Step 3: Build SDK and Run Samples

Detailed instructions for cloning the repo for downloading sample device code, setting up the development environment, compiling, programming and running the Sample Applications, are provided below:

https://github.com/InnoPhaseInc/talaria_two_azure/blob/main/README.md

Sample Applications covering the Device to Cloud, Cloud to Device, Device Twin, Direct Methods and Device Provisioning Service are available in the repo.

To connect the Talaria TWO EVB-A DevKit to Azure IoT Hub, you'll modify sample apps in few places for Azure IoT settings, rebuild the image, and provide the Wi-Fi AP setting as boot-argument when flashing the Sample App image to the DevKit using the Download Tool.

Detailed instructions for building, programming and running individual Sample Applications are provided below.

5.1 Azure IoT HUB Client Sample

Sample Application 'iothub_client_sample_mqtt' demonstrates Device to Cloud (D2C) and Cloud to Device (C2D) functionality using Azure IoT Hub.

Detailed instructions are provided in the link below for setting this up and running the sample on EVB-A to achieve these functionalities:

https://github.com/InnoPhaseInc/talaria_two_azure/blob/main/examples/iothub_client_sample_mqtt/README.md

5.2 Azure IoT HUB Device Twin and Direct Method Sample

Sample Application 'iothub_client_device_twin_and_methods_sample' demonstrates Device Twin and Direct Methods functionalities of Azure IoT Hub.

Detailed instructions are provided in the link below for setting this up and running the sample on EVB-A to achieve these functionalities:

https://github.com/InnoPhaseInc/talaria_two_azure/blob/main/examples/iothub_devicetwin_and_methods_sample/README.md

5.3 Azure IoT Hub Device Provisioning Service Sample

Sample Application 'prov_dev_client_ll_sample' demonstrates the working of Talaria TWO devices with Device Provisioning Service (DPS) for Azure IoT Hub.

Detailed instructions are provided in the link below for setting this up and running the sample on EVB-A to achieve the provisioning using Device Provisioning Service (DPS):

https://github.com/InnoPhaseInc/talaria_two_azure/blob/main/examples/prov_dev_client_ll_sample/README.md

6 Step 4: Integration with Azure IoT Explorer

6.1 Installing Azure IoT Explorer and Connecting it to a Hub

Go to Azure IoT explorer releases and expand the list of assets for the most recent release.

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

Download and install the most recent version of the application.

Connect Azure IoT Explorer to IoT Hub by providing your IoT Hub's connection string.

This can be found from the Azure Portal as detailed below:

Click on your IoT Hub > Shared access polices > iothubowner > connection string-primary key > Copy to clipboard

Launch Azure IoT Explorer then click on "Add connection" > paste the Connection String > Save.

Once the IoT Hub connection has been added, go to your specific IoT Hub and click on "View devices in this hub". Then you will see your device and can choose it, as shown in below screenshot:

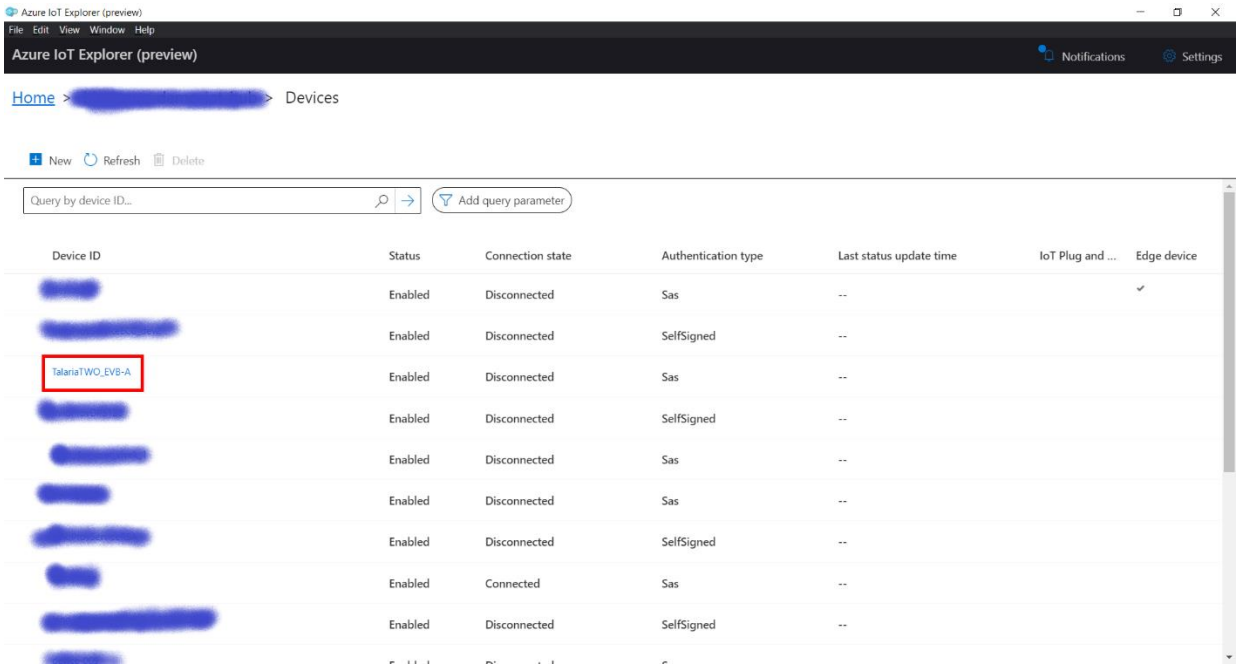

Figure 2: Connecting IoT Explorer to Hub and viewing the devices

Then, selecting the device and clicking Device identity, you will be able to see detail information of the device, as shown in below screenshot:

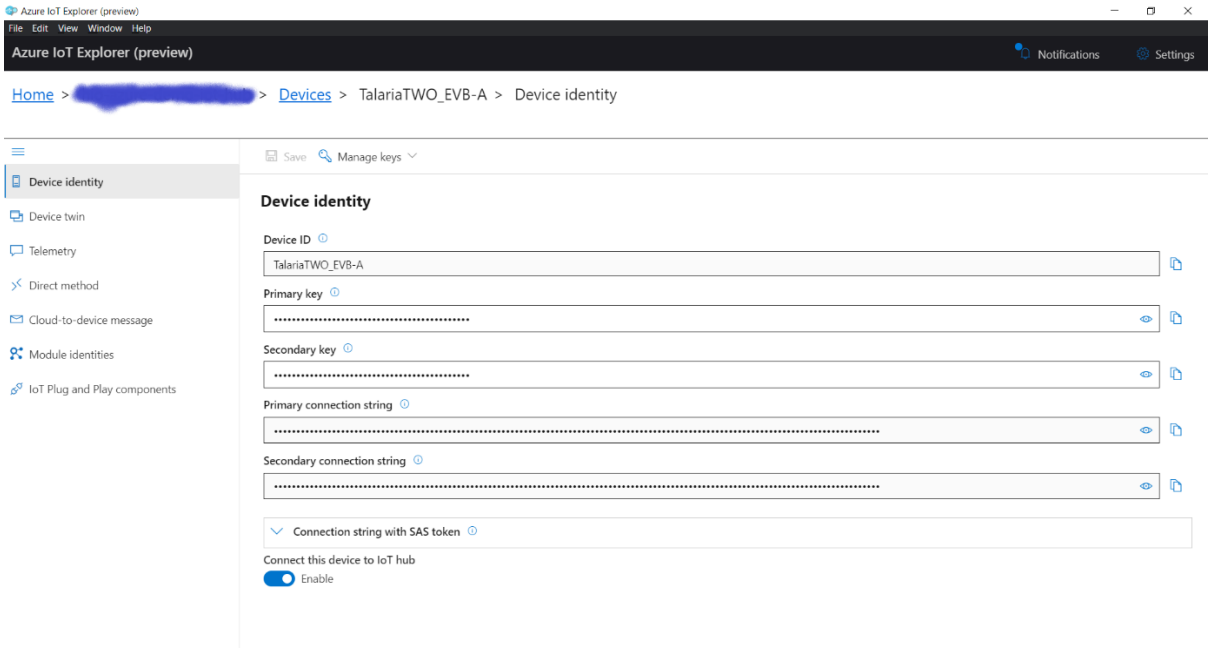

Figure 2: Viewing device identity

6.2 Interacting with the EVB-A using Azure IoT Explorer

Monitoring and interaction with the device can be performed using Azure IoT Explorer with the various options available on the left pane. All the applicable options' working with Talaria TWO EVB-A is explained in detail with the screenshots in the below sections.

6.2.1 Monitoring Device Telemetry, Device2Cloud Messages

With Azure IoT Explorer, you can view the flow of telemetry from your device to the cloud. To do this, first prepare and boot the 'Azure IoT Hub Client' Sample Application as detailed in section 5.1.

In Azure IoT Explorer select 'Telemetry' option in left panel and confirm that 'Use built-in event hub' option is set to yes. Then click 'Start' button to see telemetry as the device sends messages to the cloud.

This is shown in the screenshot below:

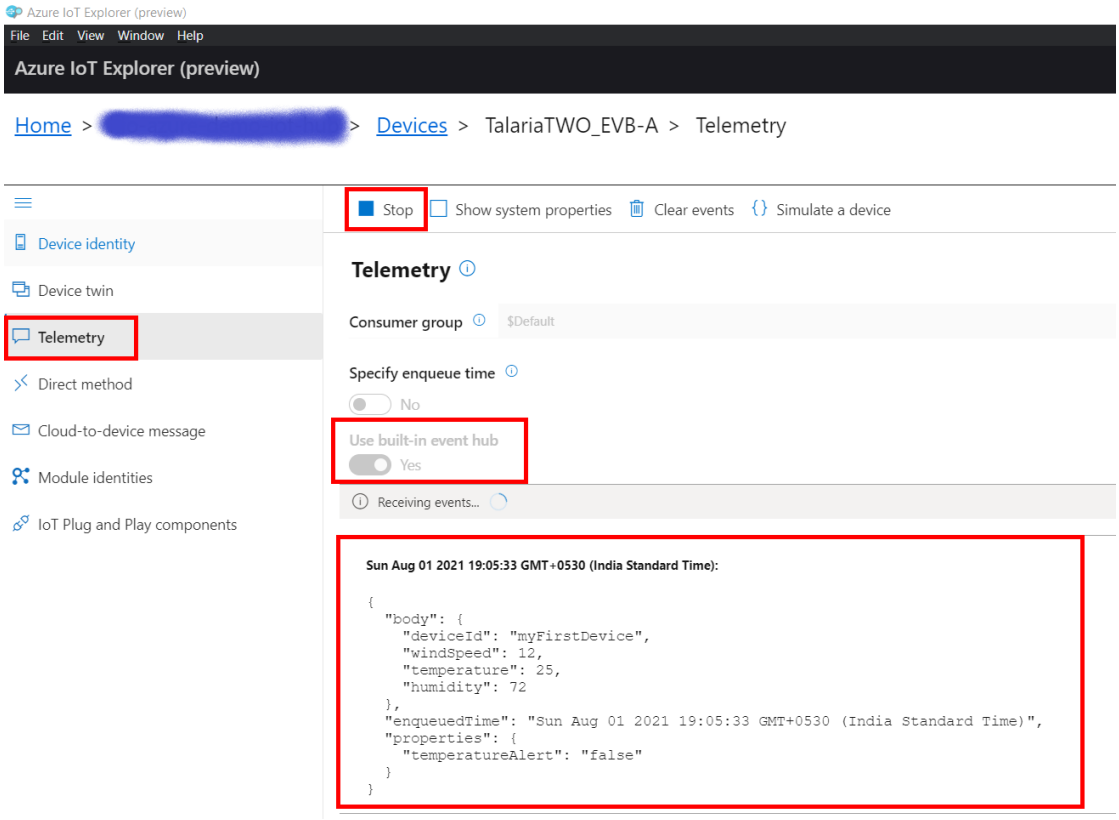

Figure 3: Monitoring device telemetry. Device2Cloud messages

6.2.2 Sending Cloud to Device Messages

Prepare and boot the 'Azure IoT Hub Client' Sample Application as detailed in section 5.1.

In Azure IoT Explorer select 'Cloud-to-device message' option in left panel and give a message body of your choice. This is shown in the screenshot shown below:

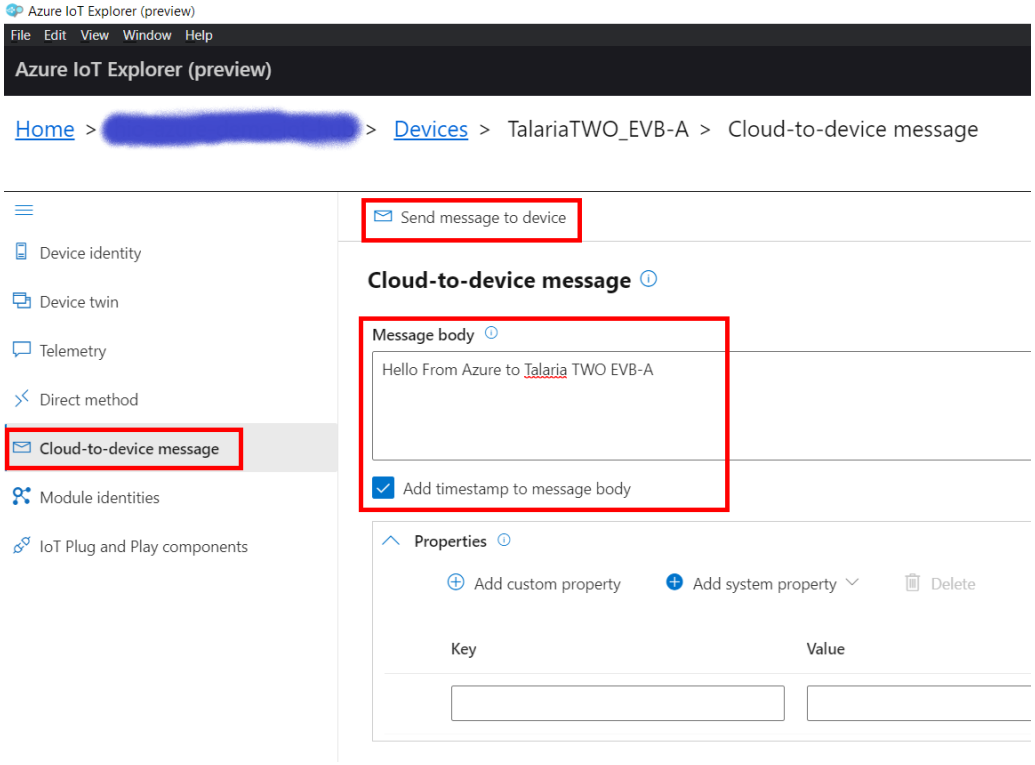

Figure 4: Sending Cloud2Device messages

Then press 'Send message to device' button shown in the screenshot. And you will see a pop-up as shown below:

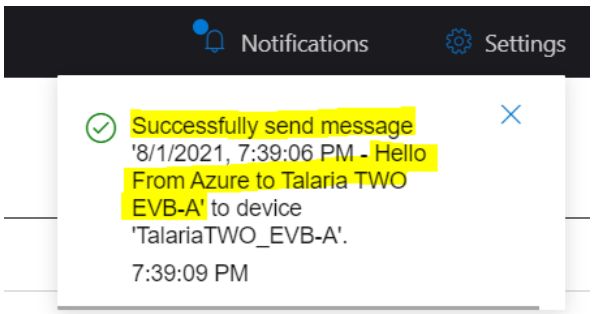

Figure 5: Sending Cloud2Device messages, success pop-up in Azure IoT Explorer

On the Talaria TWO EVB-A console, the below logs will appear -

--

Message ID: 26504ea3-9b1e-4438-8216-966390441ee2

Correlation ID: <null>

Data: <<<8/1/2021, 7:11:33 PM - Hello From Azure to Talaria TWO EVB-A>>> & Size=60

--

After sending the messages many times from the IoT Explorer, the screenshot from the device console is shown below:

Figure 6: Sending Cloud2Device messages, Talaria TWO Device Console

6.2.3 Viewing Device Twin 'reported' properties getting updated

In Azure IoT Explorer, select 'Device Twin' option in left panel.

If the device was created using the portal, by default the Device Twin JSON will look like as shown below (without any values in reported and desired properties):

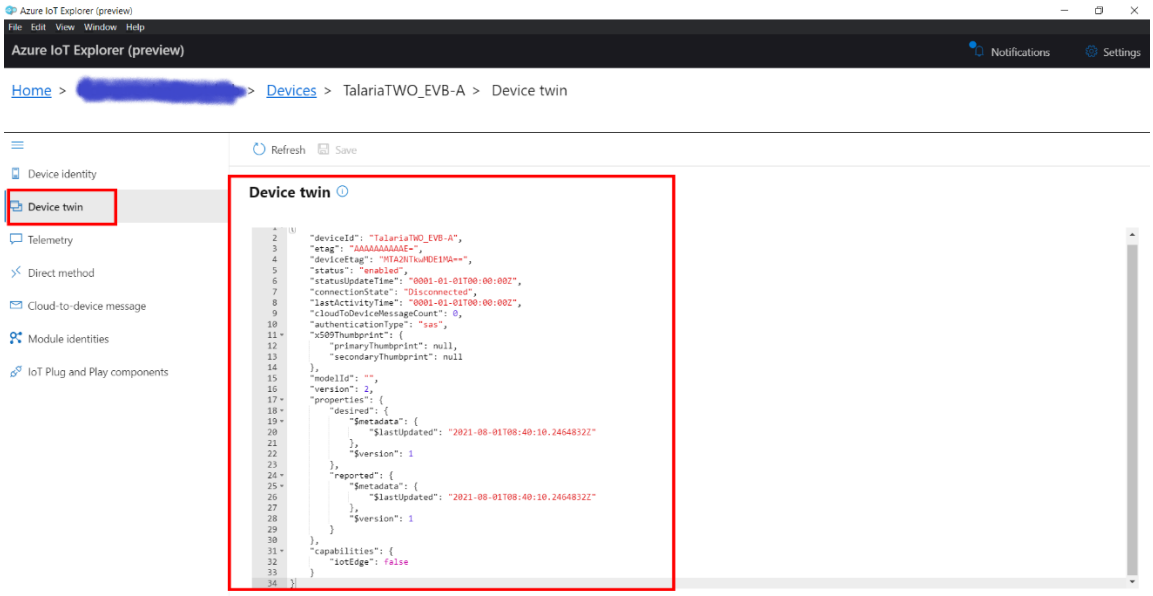

Figure 7: Default Device Twin JSON

Now, prepare and boot the 'Device Twin and Direct Method' Sample Application as detailed in section 5.2.

After running this Sample Application successfully, you will see a log:

--

'Device Twin reported properties update completed with result: 204'

--

This is shown in the screenshot below:

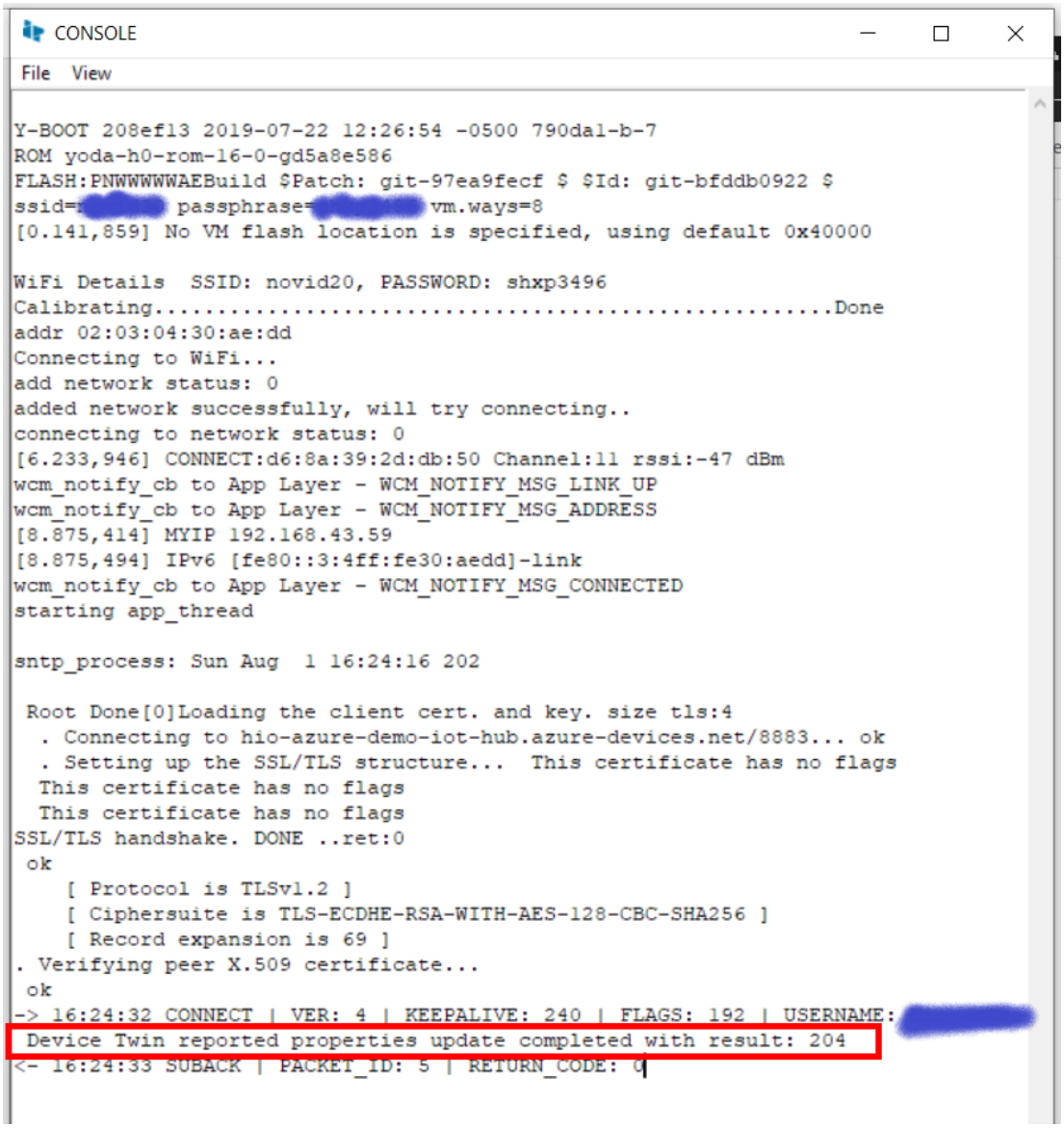

Figure 8: Device Twin 'reported' properties update success from Talaria TWO Device Console

In Azure IoT Explorer, refreshing the Device Twin, you will see the 'reported' values from running the application are reflecting now in the Device Twin JSON, as shown below:

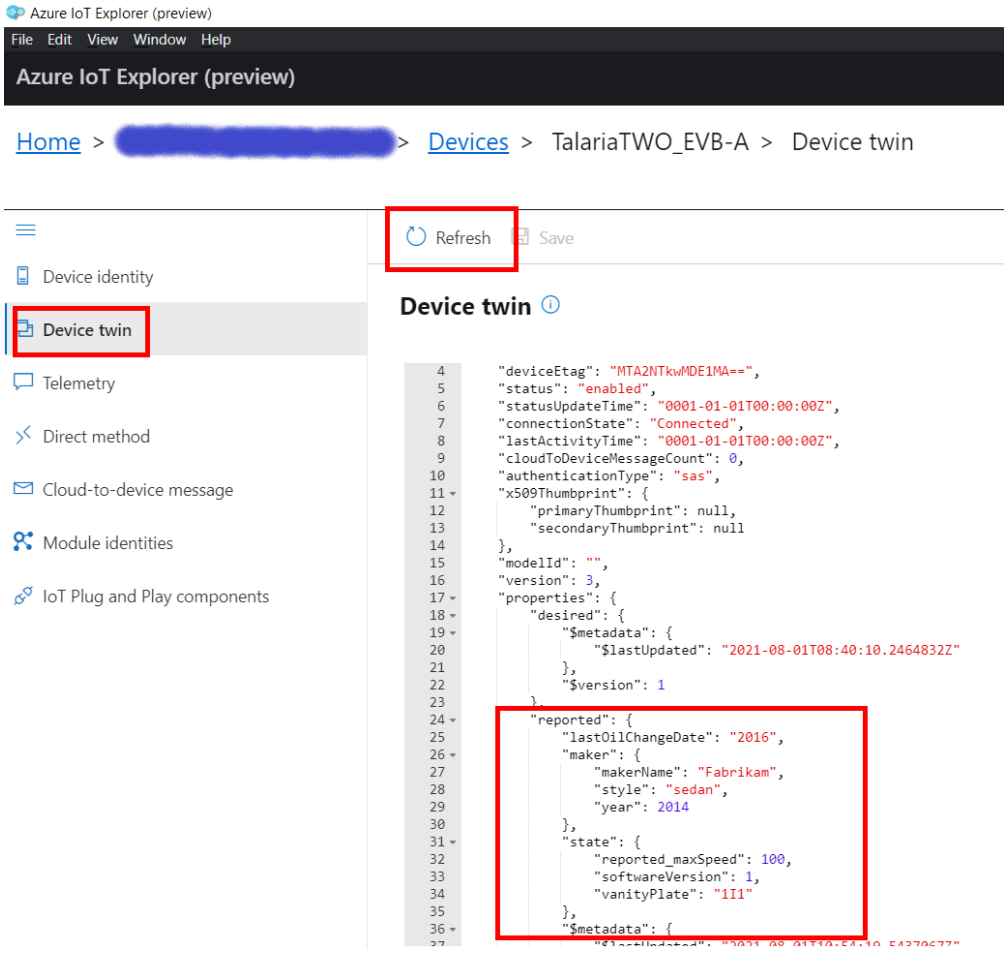

Figure 9: Device Twin 'reported' properties update success from Azure IoT Explorer

6.2.4 Updating Device Twin 'desired' properties and getting device callback

Continuing from the previous section '6.2.3', you will notice that the desired section of the Device Twin JSON is empty.

In Azure IoT Explorer, please paste the parts from the following JSON blob under `desired` property

--

```
"desired": {
  "changeOilReminder": "LOW_OIL",
  "settings": {
    "desired_maxSpeed": 126,
    "location": {
      "longitude": 72000000,
```



```
            "latitude": 26000000
          }
        },
        "$metadata": {
          "$lastUpdated": "2021-06-26T00:06:52.9307518Z"
        },
        "$version": 1
      },
```

--

Please note that the $metadata part does not need to be copied, this is just for reference as for how the final 'desired' section will look.

The final desired part should look like as shown in below screenshot:

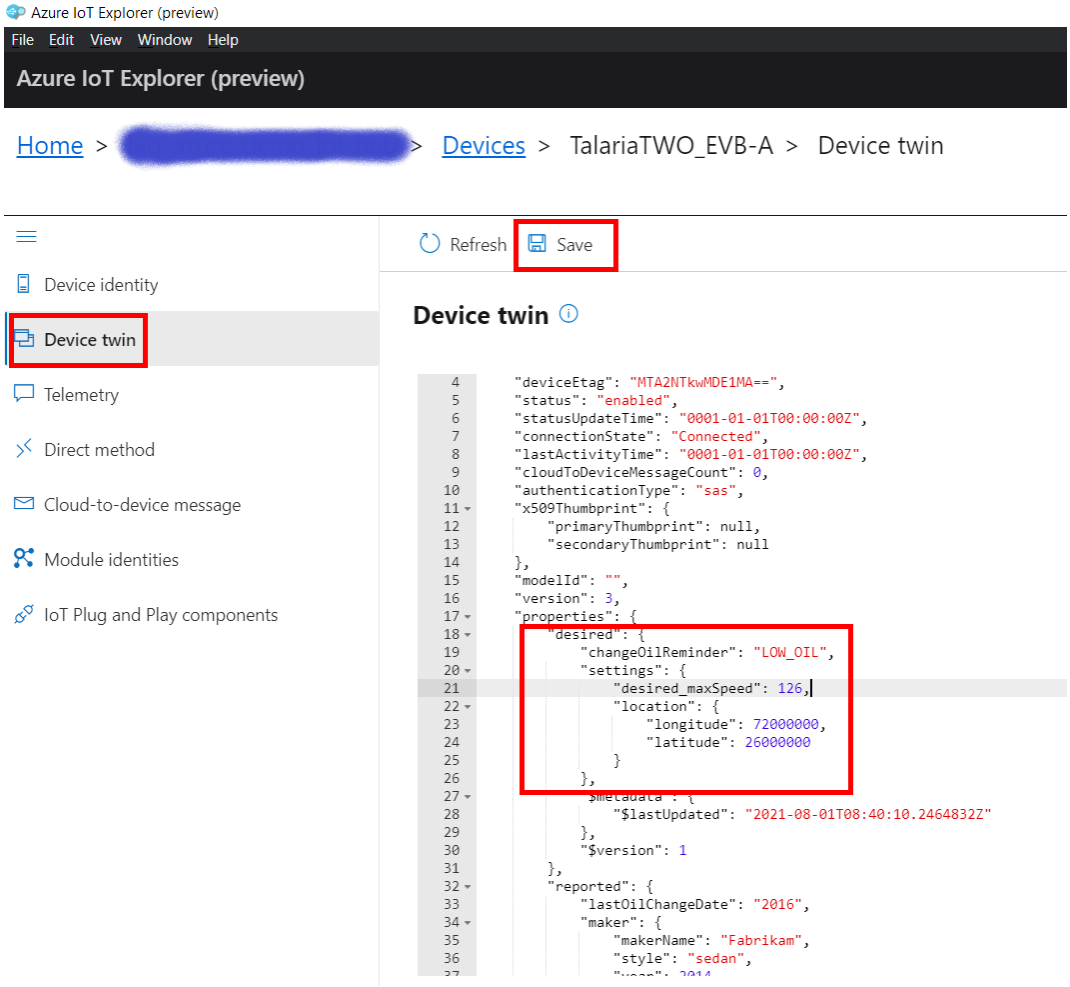

Figure 10: Updating Device Twin 'desired' properties from Azure IoT Explorer

Then press ‘Save’ button shown in the screenshot. And you will see a popup as shown below:

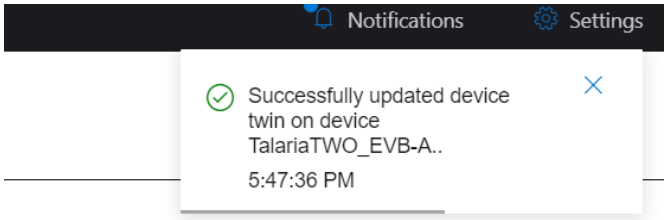

Figure 11: Updating Device Twin ‘desired’ properties, success pop-up in Azure IoT Explorer

On the Talaria TWO EVB-A console, the below logs should occur -

--

Received a new changeOilReminder = LOW_OIL

Received a new desired_maxSpeed = 126

Received a new latitude = 26000000

Received a new longitude = 72000000

--

This is shown in the screenshot below:

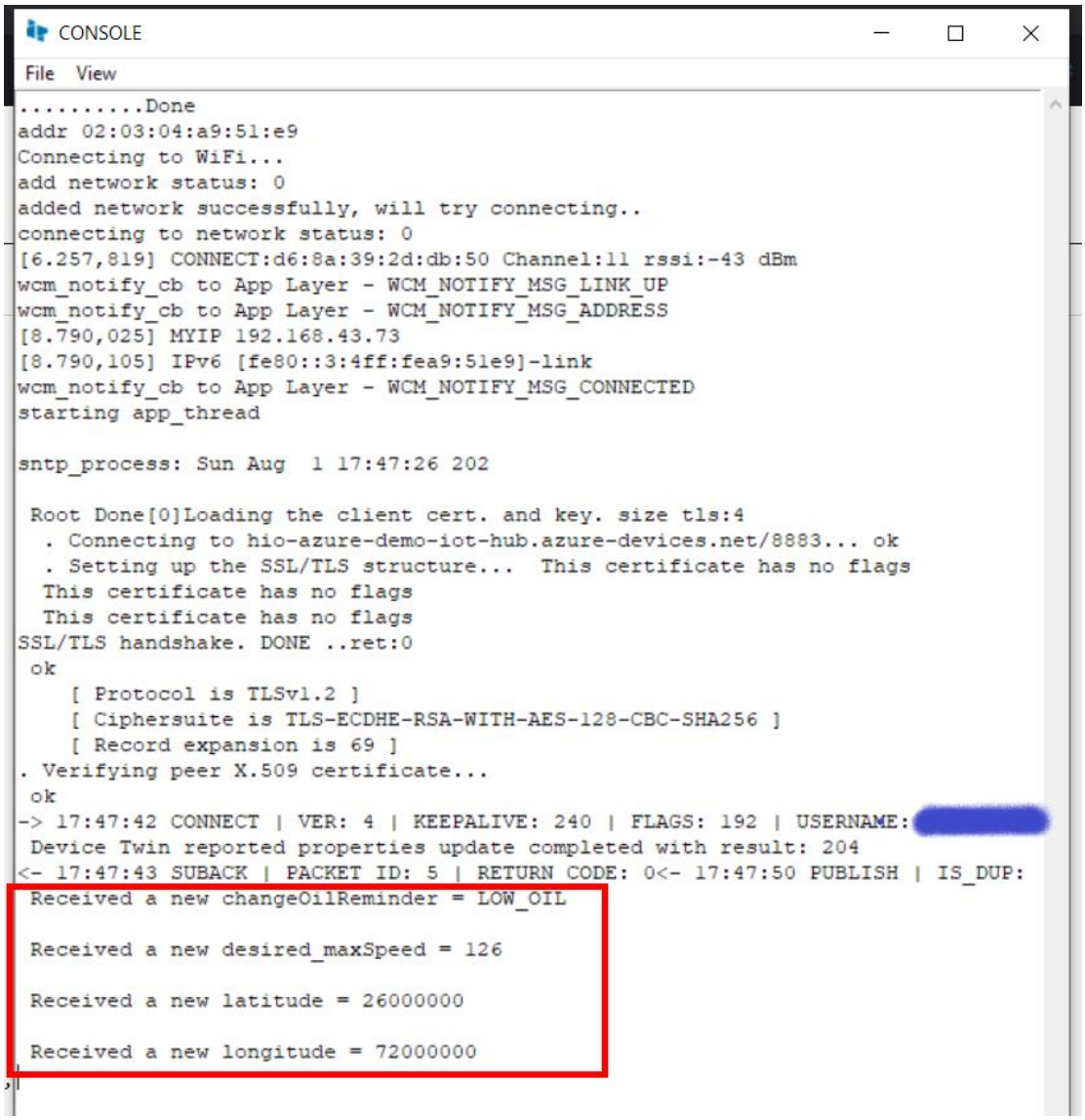

Figure 12: Updating Device Twin 'desired' properties, Talaria TWO Device Console

6.2.5 Calling a Direct Method

Prepare and boot the 'Device Twin and Direct Method' Sample Application as detailed in section 5.2.

In Azure IoT Explorer, select 'Direct Method' option in left device panel, and use method name 'getCarVIN' and choose a payload string.

This is shown in the screenshot below:

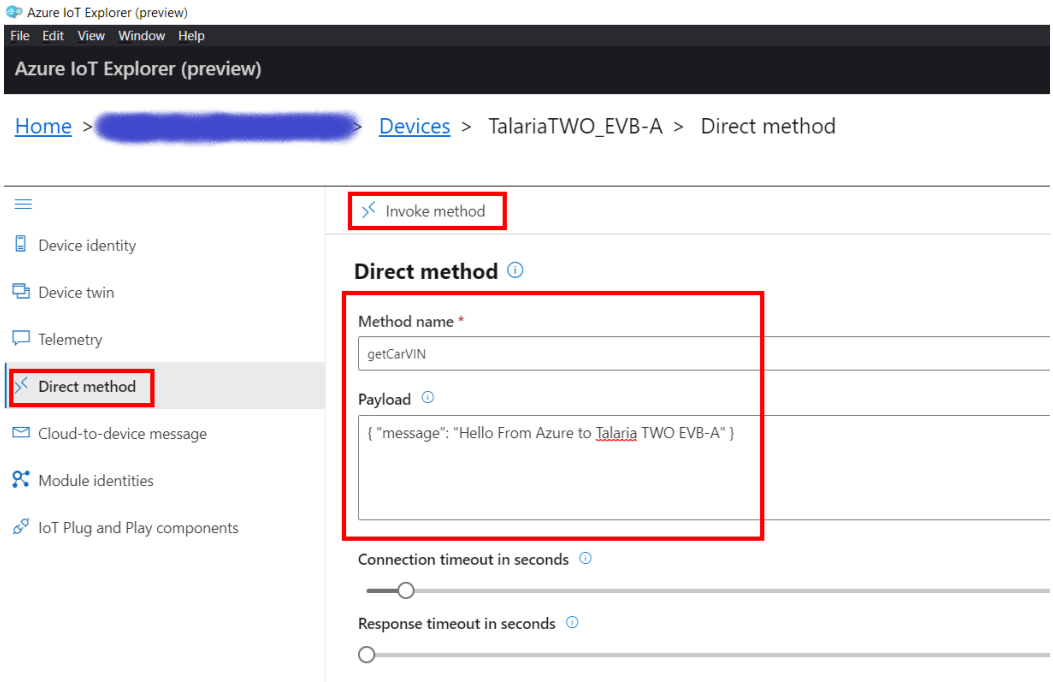

Figure 13: Invoking 'Direct Method' from Azure IoT Explorer

Then press 'Invoke method' button shown in the screenshot. And you will see a popup as shown below:

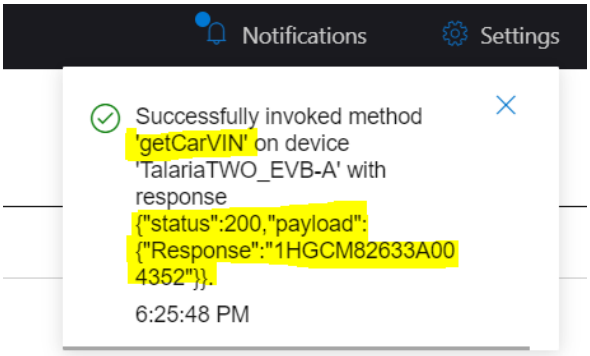

Figure 14: Invoking 'Direct Method', success pop-up in Azure IoT Explorer

On the Talaria TWO EVB-A console, the below logs will appear -

--

Method Name: getCarVIN

Payload: {"message":"Hello From Azure to Talaria TWO EVB-A"}

--

This is shown in the screenshot below:

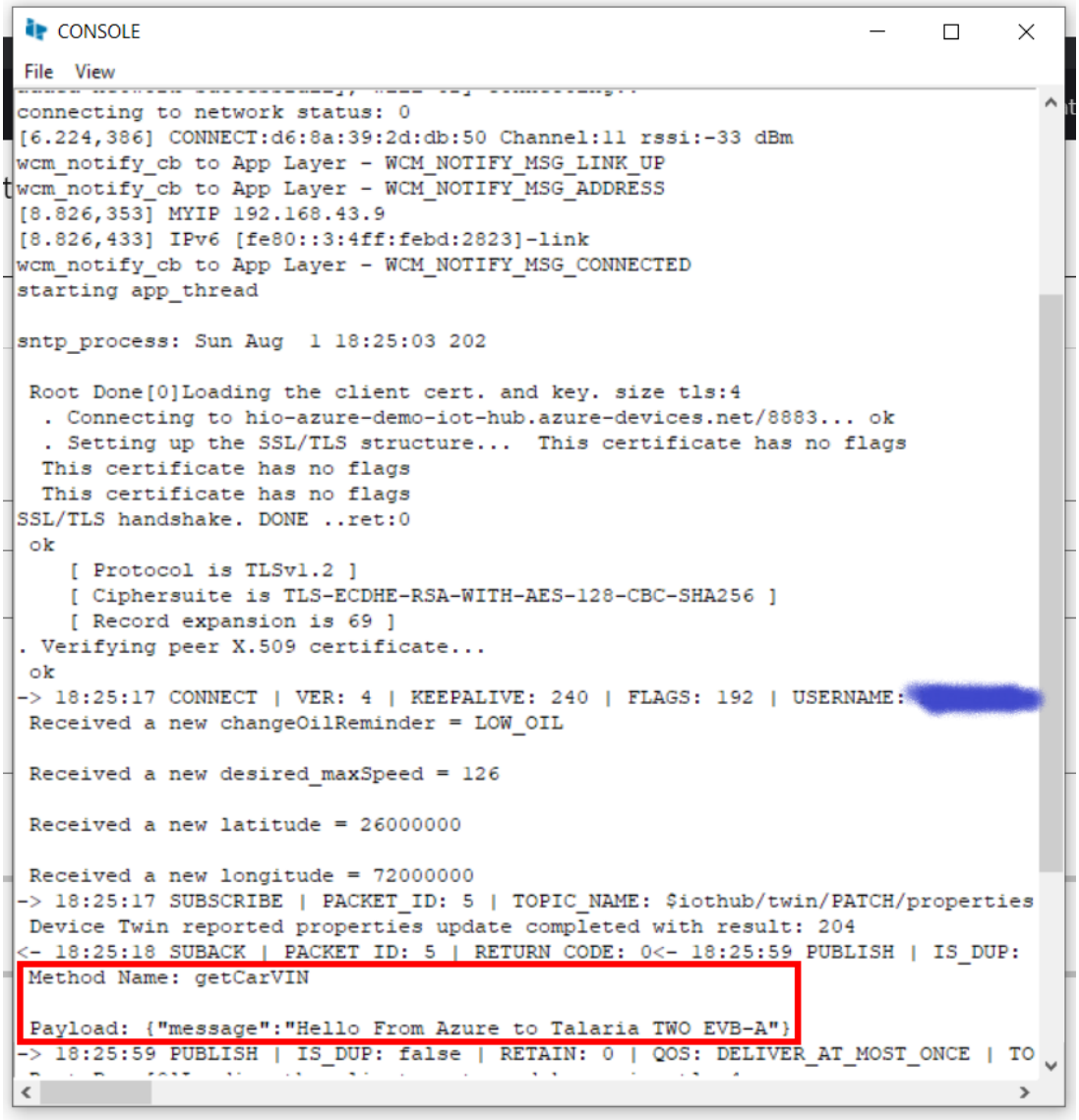

Figure 15: Invoking 'Direct Method', success from Talaria TWO Device Console

7 Step 7: Additional Links

Please refer to the below link for additional information

- Manage cloud device messaging with Azure-IoT-Explorer
  https://github.com/Azure/azure-iot-explorer/releases

- How to use IoT Explorer to interact with the device
  https://docs.microsoft.com/en-us/azure/iot-pnp/howto-use-iot-explorer#install-azure-iot-explorer

- Schematics of the kit can be downloaded from the link below:
  https://innophaseinc.com/wp-content/uploads/modules/INP3010_3011-Schematic.pdf
- Additional documentation is available on the Talaria TWO modules webpage:

  https://innophaseinc.com/talaria-two-modules#doc
- Data Sheet for the modules used in the kit can be downloaded from the link below:

  https://innophaseinc.com/wp-content/uploads/modules/INP101_INP1011-Talaria-TWO-Modules-Datasheet.pdf

8 Support

1. Sales Support: Contact an InnoPhase sales representative via email – sales@innophaseinc.com
2. Technical Support:
   a. Visit: https://innophaseinc.com/support/
   b. Also Visit: https://innophaseinc.com/talaria-two-modules
   c. Contact: support@innophaseinc.com

InnoPhase is working diligently to provide outstanding support to all customers.

9 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, InnoPhase Incorporated does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and assumes no liability associated with the use of such information. InnoPhase Incorporated takes no responsibility for the content in this document if provided by an information source outside of InnoPhase Incorporated.

InnoPhase Incorporated disclaims liability for any indirect, incidental, punitive, special or consequential damages associated with the use of this document, applications and any products associated with information in this document, whether or not such damages are based on tort (including negligence), warranty, including warranty of merchantability, warranty of fitness for a particular purpose, breach of contract or any other legal theory. Further, InnoPhase Incorporated accepts no liability and makes no warranty, express or implied, for any assistance given with respect to any applications described herein or customer product design, or the application or use by any customer's third-party customer(s).

Notwithstanding any damages that a customer might incur for any reason whatsoever, InnoPhase Incorporated' aggregate and cumulative liability for the products described herein shall be limited in accordance with the Terms and Conditions of identified in the commercial sale documentation for such InnoPhase Incorporated products.

Right to make changes — InnoPhase Incorporated reserves the right to make changes to information published in this document, including, without limitation, changes to any specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — InnoPhase Incorporated products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an InnoPhase Incorporated product can reasonably be expected to result in personal injury, death or severe property or environmental damage. InnoPhase Incorporated and its suppliers accept no liability for inclusion and/or use of InnoPhase Incorporated products in such equipment or applications and such inclusion and/or use is at the customer's own risk.'