



# **Intel® In-Band Manageability Framework – Telit DeviceWISE\***

**User Guide**

---

***June 2021***

***Revision 2.8***

**Intel Confidential**



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: <http://www.intel.com/design/literature.htm>

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at <http://www.intel.com/> or from the OEM or retailer.

No computer system can be absolutely secure.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

\*Other names and brands may be claimed as the property of others.

© Intel Corporation.

# Contents

---

<b>1.0</b>	<b>Introduction.....</b>	<b>7</b>
1.1	Purpose.....	8
1.2	Audience .....	9
1.3	Terminology .....	9
<b>2.0</b>	<b>Telit DeviceWISE* Overview .....</b>	<b>10</b>
2.1	Telit deviceWise* .....	10
2.2	Create Telit* IoT Portal Account.....	10
2.3	Telit* Login Page .....	11
2.4	Getting Familiar with Telit* Home Page Tabs .....	12
2.5	Import a <i>Thing</i> Definition.....	13
2.6	Obtaining Application Token .....	15
2.7	Provisioning with Telit DeviceWISE* Token.....	16
2.8	Assign Correct <i>Thing</i> Definition to the Provisioned Device .....	18
2.9	Familiarize with OTA Widgets .....	19
<b>3.0</b>	<b>OTA Updates .....</b>	<b>23</b>
3.1	Trusted Repositories.....	23
3.1	Preparing OTA Update Packages .....	24
3.2	How to Generate Signature.....	26
3.3	OTA Commands.....	26
3.4	AOTA Updates.....	28
3.5	FOTA Updates .....	43
3.6	SOTA Updates.....	47
3.7	Configuration Update.....	51
3.8	Power Management .....	60
3.9	Decommission.....	62
<b>4.0</b>	<b>Telemetry Data.....</b>	<b>63</b>
4.1	Static Telemetry .....	63
4.2	Dynamic Telemetry .....	63
4.3	Viewing Telemetry Data.....	64
<b>5.0</b>	<b>Issues and Troubleshooting.....</b>	<b>66</b>
5.1	OTA Error Status.....	66
5.2	Provisioning Unsuccessful or Device not connected to the Cloud .....	67
5.3	Acquiring Debug Messages from Agents.....	67

## Figures

Figure 1. Framework.....	7
Figure 2. Live Updates Settings .....	10
Figure 3. Telit* IoT Portal.....	11
Figure 4. Telit* Homepage .....	12
Figure 5. Developer Page.....	13
Figure 6. Select Thing Definitions.....	14
Figure 7. Thing Definitions Import .....	14
Figure 8. Applications.....	15
Figure 9. Check Connected Device.....	18
Figure 10. Set <i>Things</i> Definition .....	19
Figure 11. Things Tab.....	20
Figure 12. Methods Tab .....	21
Figure 13. Actions .....	21
Figure 14. Logs .....	22
Figure 15. Trigger AOTA.....	29
Figure 16. Fields Indicator .....	31
Figure 17. Docker-Compose Up .....	32
Figure 18. Docker-Compose Down .....	33
Figure 19. Docker-Compose Pull .....	34
Figure 20. Docker-Compose List .....	35
Figure 21. Docker-Compose Remove .....	36
Figure 22. Docker Import .....	37
Figure 23. Docker Load .....	38
Figure 24. Docker Pull.....	39
Figure 25. Docker Remove .....	40
Figure 26. Docker Stats .....	41
Figure 27. Application Update .....	42
Figure 28. Trigger FOTA .....	44
Figure 29. Trigger Fota.....	45
Figure 30. Trigger SOTA.....	47
Figure 31. Trigger SOTA.....	48
Figure 32. Trigger SOTA.....	49
Figure 33. Trigger SOTA.....	50
Figure 34. Trigger Config Update.....	52
Figure 35. Execute Config Update .....	53
Figure 36. Configuration Get.....	55
Figure 37. Events Tab .....	56
Figure 38. Configuration Load.....	56
Figure 39. Configuration Append.....	57
Figure 40. Configuration Remove.....	59
Figure 41. Reboot and Shutdown.....	60
Figure 42. Shutdown.....	61
Figure 43. Reboot.....	61
Figure 44. Decommission .....	62

Figure 45. Decommission .....	62
Figure 46. Things.....	64
Figure 47. Events .....	65
Figure 48. Results .....	65

## Tables

Table 1. Use Cases.....	8
Table 2. Features.....	8
Table 3. Terminology.....	9
Table 4. Creating AOTA Package.....	25
Table 5. Commands - Definitions and Usage.....	27
Table 6. 'docker-compose' commands.....	28
Table 7. 'docker' commands.....	28
Table 8. List of AOTA commands NOT supported.....	29
Table 9. AOTA Field Description.....	29
Table 10. FOTA Update Info .....	43
Table 11. Parameters .....	46
Table 12. Default Configuration Parameters.....	51
Table 13. Configuration Update Commands and Input Field Description .....	52
Table 14. OTA Error Status.....	66

## Revision History

---

Date	Revision	Description
June 2021	2.8	Added provision-tc parameters info Added clarity on AOTA package generation
April 2021	2.x	Add customer NOTE on Trusted repositories
September 2020	2.7	Changes on AOTA command
August 2020	2.6	EIS 2.3, ECS 1.5 and Platform releases
May 2020	2.0	Final release
April 2020	1.0	Initial release

## 1.0 Introduction

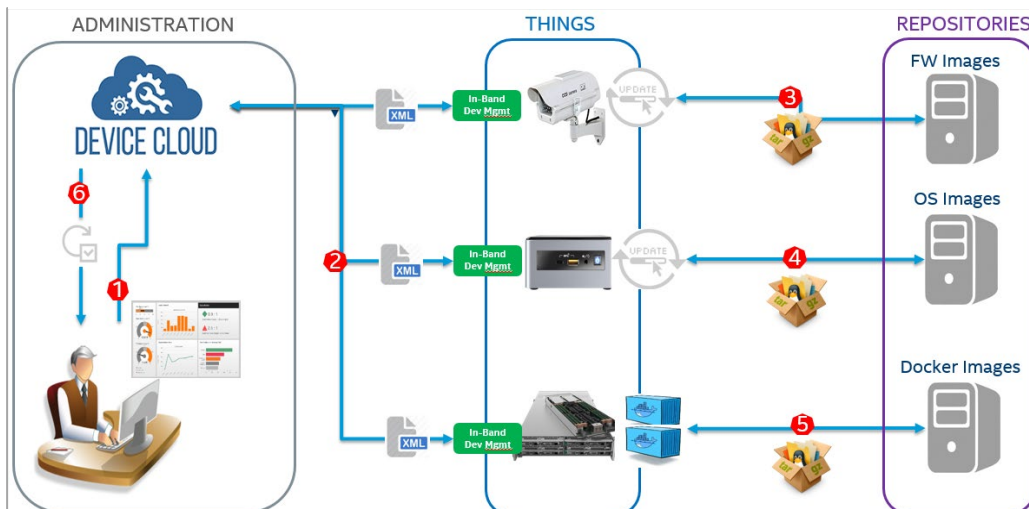
Intel® In-Band Manageability Framework is a software running on Edge IoT Device, enables an administrator to perform critical Device Management operations remotely in clouds. It also facilitates in publishing of telemetry and critical events and logs from the Edge IoT device to the cloud, and enabling administrator to take corrective actions necessarily.

The framework is designed to be modular and flexible, in order to ensure scalability of the solution across preferred Cloud Service Providers (for example, Azure\* IoT Central, Telit DeviceWISE\*, ThingBoard.io\*, etc.).

Some of the key advantages of the Intel® In-Band Manageability solutions are:

1. Out-of-box cloud support: Azure\* IoT Central, Telit DeviseWise\*, ThingsBoard.io\*.
2. Single interface to handle operating system, firmware and application (Docker container) updates.
3. Scalable across Intel® x86 (Intel Atom® processor and Intel® Core™ processor) architectures SoCs and on Vision platforms from Intel.

Figure 1. Framework



This document provides detailed instructions about provisioning a device with **Telit DeviseWISE\***.

- Refer to RDC Document Number: **626762, Intel® In-Band Manageability – Azure\*** for provisioning with **Azure\* IOT Central**.

- Refer to RDC Document Number: **626763, Intel® In-Band Manageability – ThingsBoard\*** for provisioning with **Thingsboard\***.

The Device Management use-cases covered by the Intel® In-Band Manageability Framework are listed in the Table 1:

**Table 1. Use Cases**

Use-cases	Notes
Update	<ul style="list-style-type: none"> <li>- System (OS), Software-over-the-air (SOTA)</li> <li>- Firmware-over-the-air (FOTA)</li> <li>- Application-over-the-air (AOTA)</li> </ul>
Telemetry	<ul style="list-style-type: none"> <li>- System attributes</li> <li>- Events</li> <li>- Devices States</li> <li>- Usage data</li> </ul>
Recovery	<ul style="list-style-type: none"> <li>- Rollback post updates</li> <li>- System Reboot/Shutdown</li> </ul>

Embedded within the Intel® In-Band Manageability Framework are features which ensure the Security and Diagnostics aspects:

**Table 2. Features**

Feature	Notes
Security	<ul style="list-style-type: none"> <li>- ACL for trusted repositories</li> <li>- Mutual TLS authentication between services</li> <li>- TPM to store framework secrets</li> </ul>
Diagnostics	<ul style="list-style-type: none"> <li>- Pre and Post OTA update checks</li> <li>- Periodic system checks</li> </ul>

## 1.1 Purpose

This User Guide serves the user on how to:

- Login and setup Telit DeviseWISE\*.
- Provision the Edge IoT device running the Intel® In-Band Manageability Framework.
- Perform OTA updates through the Telit DeviseWISE\* portal.



It also provides examples of Web-UI configuration, the reported Telemetry from the device and commands for performing OTA updates.

## 1.2 Audience

This User Guide is intended for:

- Independent BIOS Vendors providing Firmware Update packages to ensure FW update binary packaging.
- Independent Software Vendors (ISV) providing OS and Application update packages.
- System Integrators administrating devices running Intel® In-Band Manageability framework.

## 1.3 Terminology

**Table 3. Terminology**

Term	Description
AOTA	Application Over the Air (Docked)
BIOS	Basic Input Output System
FOTA	Firmware Over the Air
FW	Firmware
IoT	Internet of Things
ISV	Independent Software Vendors
OS	Operating System
OTA	Over-the-air
SMBIOS	System Management BIOS
SOTA	Software Over the Air (OS update)

## 2.0 Telit DeviceWISE\* Overview

---

### 2.1 Telit deviceWise\*

To provision a device with Telit deviceWise\*, user must have an account in Telit\* portal. The connection can only be made if user has a group or organization on the service. Visit the deviceWise\* domain to create an account and get an "org" or "application." Refer to <https://docs.devicewise.com/Content/GettingStarted/Getting-Started-with-IoT-Platform.htm> to get started.

### 2.2 Create Telit\* IoT Portal Account

Refer to <https://docs.devicewise.com/Content/GettingStarted/IoT-Portal-Part-1---Creating-your-account.htm> to create an account.

After you set up your password and accept Telit's\* terms and conditions, the first time you click on **Things**, you will be asked to set up live updates. This is optional, however Intel recommends putting 5 seconds for getting updates. Refer to Figure 2.

Figure 2. Live Updates Settings

**Live updates for Things**

Things support live updating. You can have the list of Things, or an individual Thing, automatically update live while leaving the page open.

This feature can be enabled/disabled and timing adjusted from your User profile.

To set this up now, just select Enable and a frequency below.

*Note: Use of this feature will result in increased API usage.*

☒ Enable live update for things

Frequency

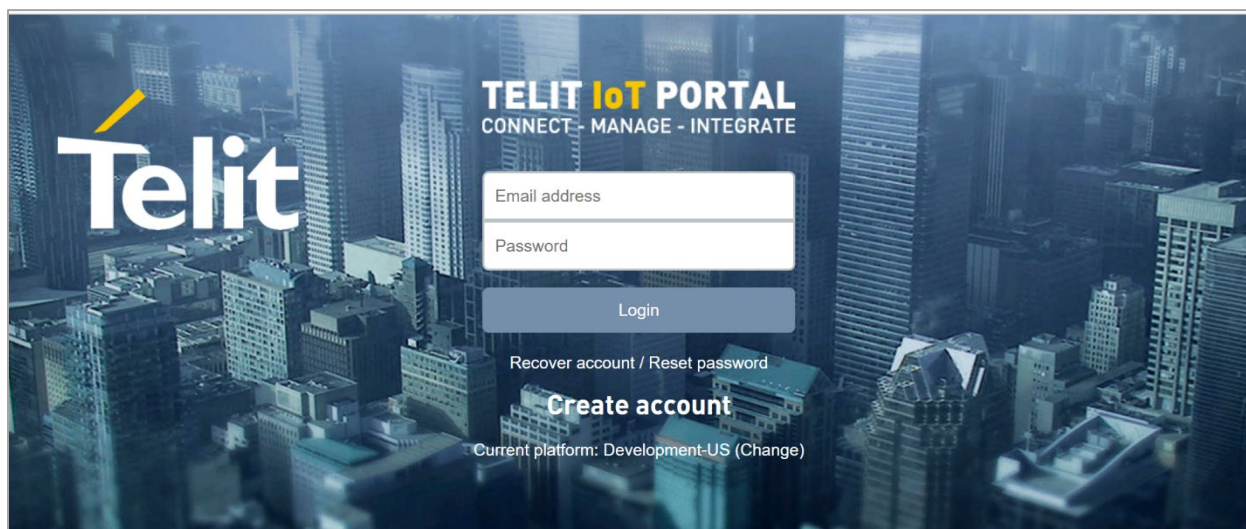
5 seconds

Update

## 2.3 Telit\* Login Page

Once user has successfully created an account, the Telit\* IoT Portal can now be accessed at <https://portal.telit.com/app/login>

Figure 3. Telit\* IoT Portal



**Note:** Enter user ID and password to log in. Login and password are case sensitive.

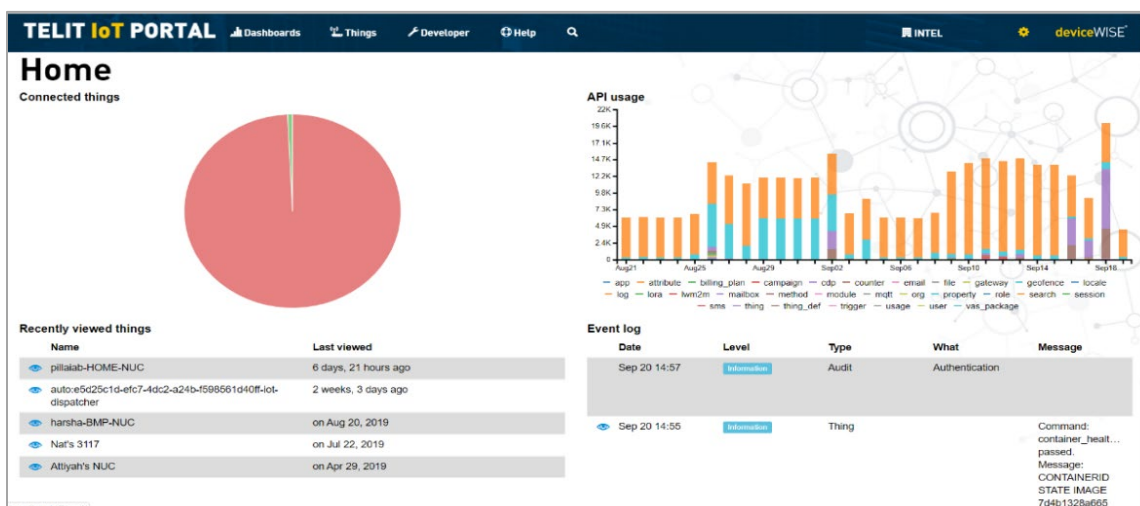
## 2.4 Getting Familiar with Telit\* Home Page Tabs

### 2.4.1 Homepage

Telits\* homepage shows a dashboard displaying recently viewed [things](#) and [connections](#), 30-days API usage, recent event logs and pie charts (showing a quick overview of connected *Things*) and status information.

The dashboard is customized as per your organization. To see dashboard data, you need to have *Things* connected.

Figure 4. Telit\* Homepage



### 2.4.2 Dashboards

**Dashboard** tab provides tools for user to customize their Dashboard view using available widgets so that user can get easy access to their data.

### 2.4.3 Things

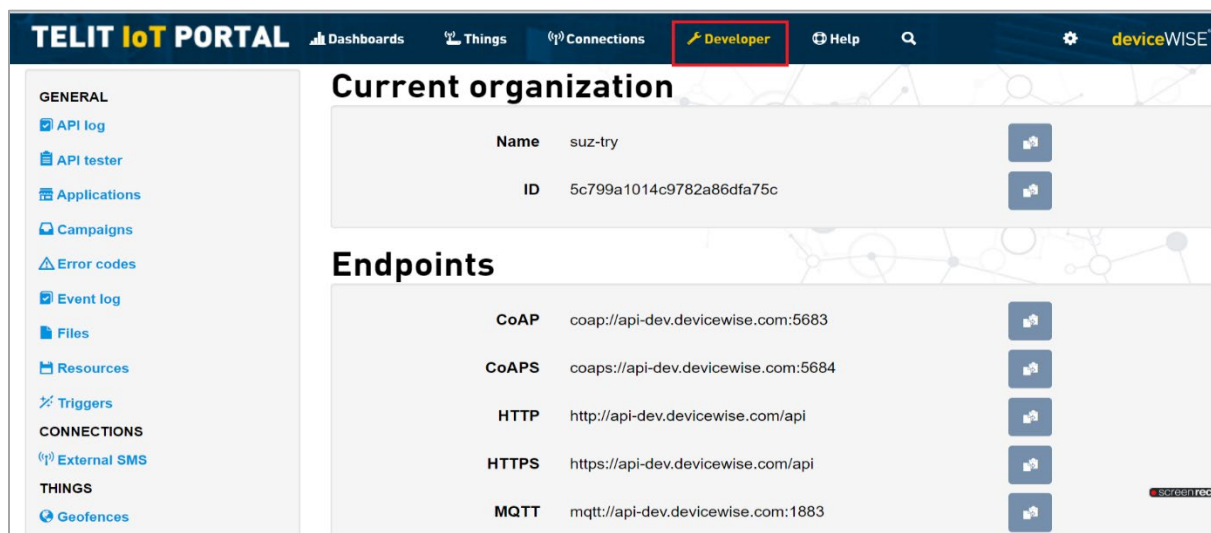
A **Thing** is a digital representation of a physical device, an application, a logical entity, or a vehicle. Each endpoint is categorized based on [Thing definitions](#). A [Thing definitions](#) is composed of properties and characteristics required to define a **Thing**.

The [Things](#) page in Telit\* portal allows user to see which endpoints have recently been added and an event log of activity within your organization.

## 2.4.4 Developer

The **Developer** page provides you an access to the tools needed in order to build applications using the Management Portal.

**Figure 5. Developer Page**



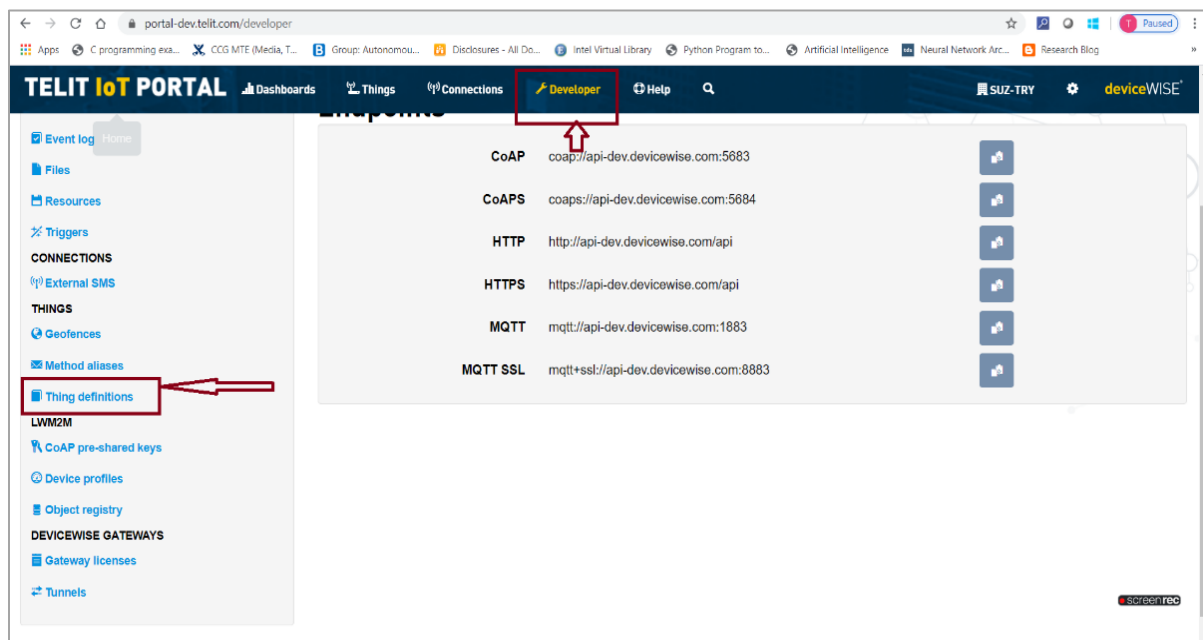
## 2.5 Import a *Thing* Definition

To use Device Management features enabled by the Intel® In-Band Manageability Framework, user needs to first import a "Thing Definition."

**Note:** Users can use the **thing\_defs.json** file located at `/usr/share/cloudadapter-agent/thing_defs.json` provided in the framework package as a reference and import it to their Telit\* account by following the steps below.

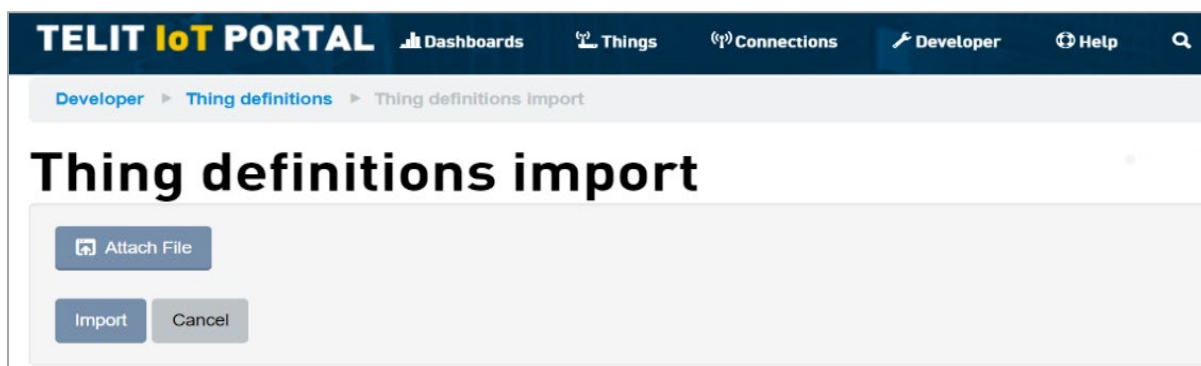
1. Log in to Telit DeviceWISE\* at <https://portal.telit.com/app/login> and click **Developer** tab.
2. Select **Thing Definitions** located at the left bar drop-down menu as shown in Figure 6.

Figure 6. Select Thing Definitions



- When a sub-screen pops-up, attach "**thing\_defs.json**" file and then click **Import** button.

Figure 7. Thing Definitions Import



- The new *Thing* definition should now be visible in the Telit\* list of definitions under "*Thing Definitions*" on the Developer tab.
- Once the *Thing* definition is installed, the Methods tab should be enabled, and you should be able to see the widgets that Intel® In-Band Manageability supports.

**Note:** The Telit\* website provides instructions for navigating the user interface to complete the steps of importing a *Thing* definition and performing required configuration tasks. The steps discussed in this document are generalized, as the name and placement of specific buttons and menus within the Telit\* website can be changed without a notice.

## 2.6 Obtaining Application Token

In the Telit\* portal, view the Developer tab and click **Applications** from the menu on the left. Copy and retain the application token, it is a required parameter of the provisioning script.

**Note:** Please be careful with your application token and DO NOT share it with anyone as it is used to connect to the cloud portal.

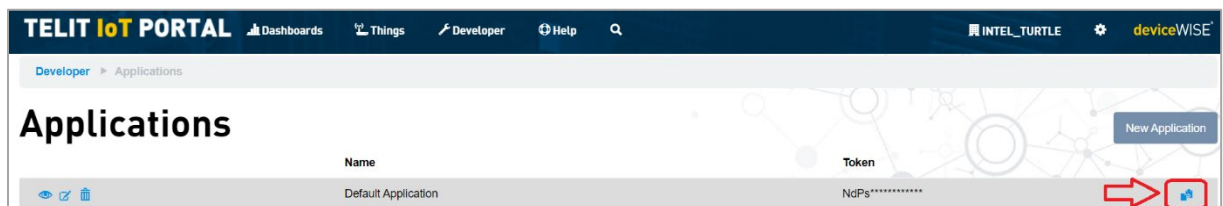
This Application token is used to link your *thing* to the Telit\* cloud and during provisioning.

1. Log in to your Telit\* IoT Portal at <https://portal.telit.com/app/login>

**Note:** The first time you log in, it will ask you to Enable Live Update for *things*. Click the box and set 5 seconds for the setting.

2. Retrieve Token
  - i. Click **Developer** tab.
  - ii. Click on **Applications**.
  - iii. Copy encrypted token.

**Figure 8. Applications**



## 2.7 Provisioning with Telit DeviceWISE\* Token

Provisioning is a Device Management phase when the Edge IoT Device is configured with the credentials to ensure that the Device Management use cases can be performed. (Example: Telemetry data reporting, OTA updating, and so on). It usually involves assigning Device ID's and Secure tokens/keys which the Device may use to identify and authenticate itself to the remote Device Management Portal.

### 2.7.1 Assumptions and Prerequisites

- Intel® In-Band Manageability Framework is installed on the device.
- The date and time on the edge device needs to be set correctly.
- Once you have copied your Application Token, you are ready to provision.

1. To provision, you need the INB provisioning script. Launch it using the command below:

```
$ sudo provision-tc
```

2. The provisioning scripts first detect the OS that it is running on and checks for TPM access, post which it starts the MQTT broker service which is the message bus used by the Intel® In-Band Manageability framework services.

```
root@ecs-intel-7004:~# provision-tc
Detected Yocto; skipping Docker configuration
Detecting TPM (can take up to 10 seconds)...
TPM detected. Enabling TPM for provisioning.
Enabling and starting mqtt (this may take some time to generate secrets)...
Created symlink /etc/systemd/system/multi-user.target.wants/mqtt.service → /etc/systemd/system/mqtt.service.
```

3. A prompt message will appear asking which cloud service to use; press **1** and **[ENTER]** for Telit\*:

```
Please choose a cloud service to use:
1) Telit Device Cloud 3) ThingsBoard
2) Azure IoT Central 4) Custom
```

4. Please select the Telit\* host to us. Enter **1** or **2**.

```
Please select Telit host to use:
1. Production (api.devicewise.com)
2. Development (api-dev.devicewise.com)
```



5. When prompted, paste the token from the text editor:

```
Provide Telit token:
```

**IMPORTANT NOTE:** Please note that Telit Token is common for every device in the Tenant Group making it critical to ensure its security and handling. Its should be kept a secret and should not be provided to unauthorized users to prevent any possible Security issues. Please consult Telit for further recommendations.

6. Either enter a desired *Thing* Key (a unique identifier), or press enter to generate one:

```
Provide Telit Thing Key (leave blank to autogenerate):
```

7. The provisioned *Thing* Key will then appear, in case the key was generated.

**Note:** Save this *Thing* Key somewhere to identify the device on the Telit\* portal.

```
Thing Key: my-thing-123
```

8. Then, you will see if you could successfully configure a cloud service.

```
Successfully configured cloud service!
```

9. The Intel® In-Band Manageability Framework services are enabled and started as seen below.

```
Enabling and starting agents...

Created symlink /etc/systemd/system/multi-user.target.wants/configuration.service →
/etc/systemd/system/configuration.service.

Created symlink /etc/systemd/system/multi-user.target.wants/dispatcher.service →
/etc/systemd/system/dispatcher.service.

Created symlink /etc/systemd/system/multi-user.target.wants/diagnostic.service →
/etc/systemd/system/diagnostic.service.

Created symlink /etc/systemd/system/multi-user.target.wants/cloudadapter.service →
/etc/systemd/system/cloudadapter.service.

Created symlink /etc/systemd/system/multi-user.target.wants/telemetry.service →
/etc/systemd/system/telemetry.service.

Turtle Creek Provisioning Complete
```

**Note:** If you failed to provision, refer to [Section 5.0](#) for Troubleshooting.

## 2.7.2 Provisioning command parameters

Provisioning can be done with or without TPM security by setting 'PROVISION\_TPM'. 'PROVISION\_TPM' can be set to:

- auto: use TPM if present; disable if not present; do not prompt.
- disable: do not use TPM.
- enable: use TPM; return error code if TPM not detected.
- (unset): default behavior; use TPM if present, prompt if not.

To run provisioning with detecting automatically TPM is present or not:

```
$sudo PROVISION_TPM=auto provision-tc
```

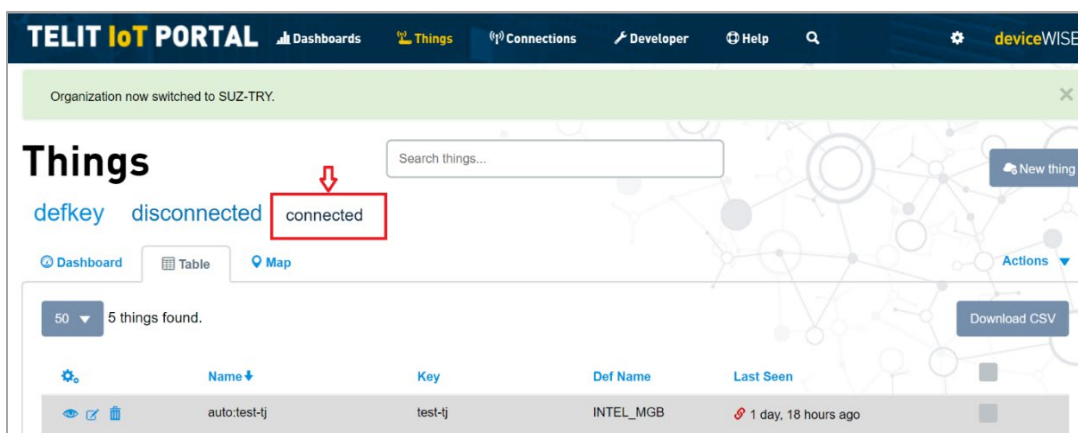
To run without TPM security:

```
$sudo PROVISION_TPM=disable provision-tc
```

## 2.8 Assign Correct *Thing* Definition to the Provisioned Device

To see whether your device is connected to Telit\* cloud, click the **connected** tab as shown in Figure 9.

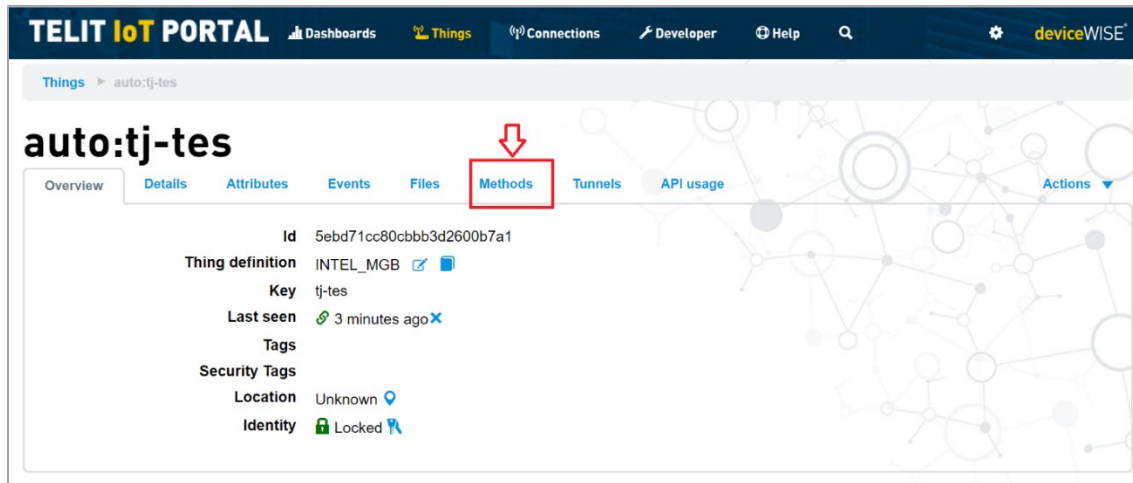
Figure 9. Check Connected Device



### 2.8.1 Set *Things* Definition

1. Click **Things** tab
2. Click the (eye icon) on your *thing* to see its details
3. To set *Thing* Definition, click (**book icon**)
4. Choose Definition **INTEL\_MGB**
5. Submit, Okay

**Figure 10. Set *Things* Definition**



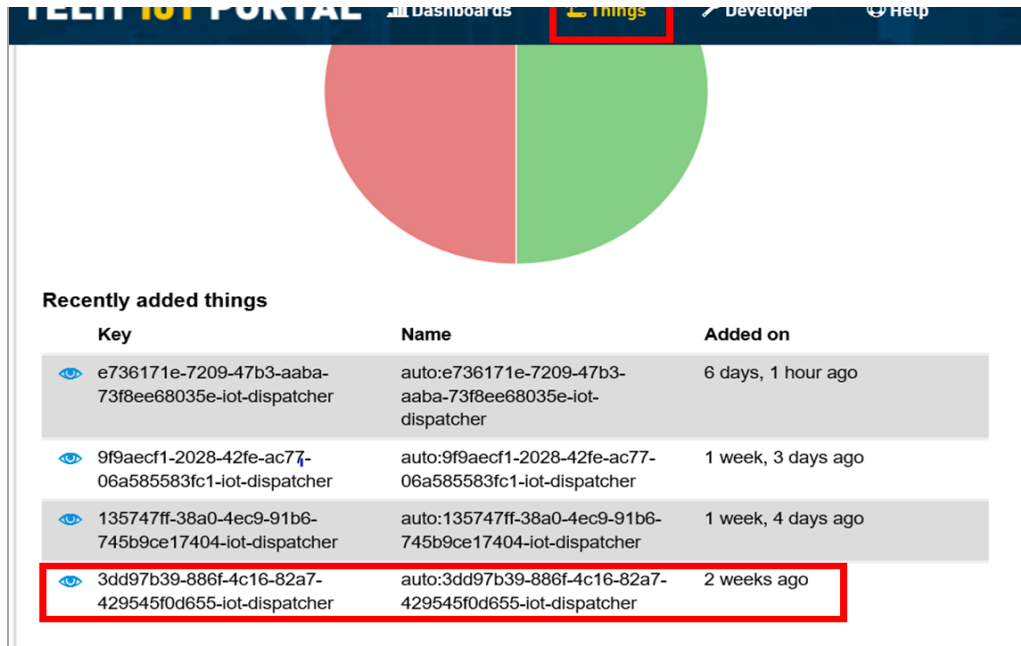
You can now click on the **Methods** tab to trigger any OTA functionality.

## 2.9 Familiarize with OTA Widgets

Method widgets would show up for a *Thing* once the *Thing*-definition is successfully imported by following instructions in [Import a Thing Definition](#). To access the Method widgets, users would need to follow the steps below:

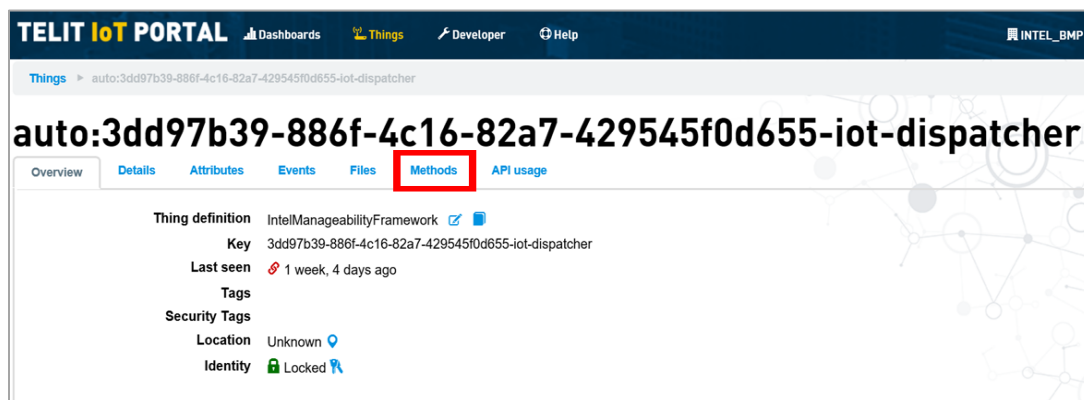
1. Click the (eye icon) of the device which need to be updated.

Figure 11. Things Tab



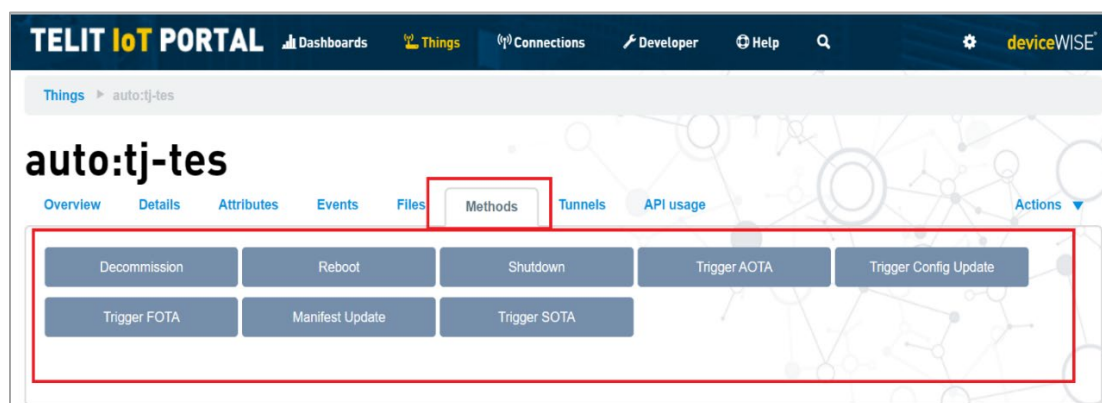
2. Click **Methods** tab as shown in Figure 12.

Figure 12. Methods Tab



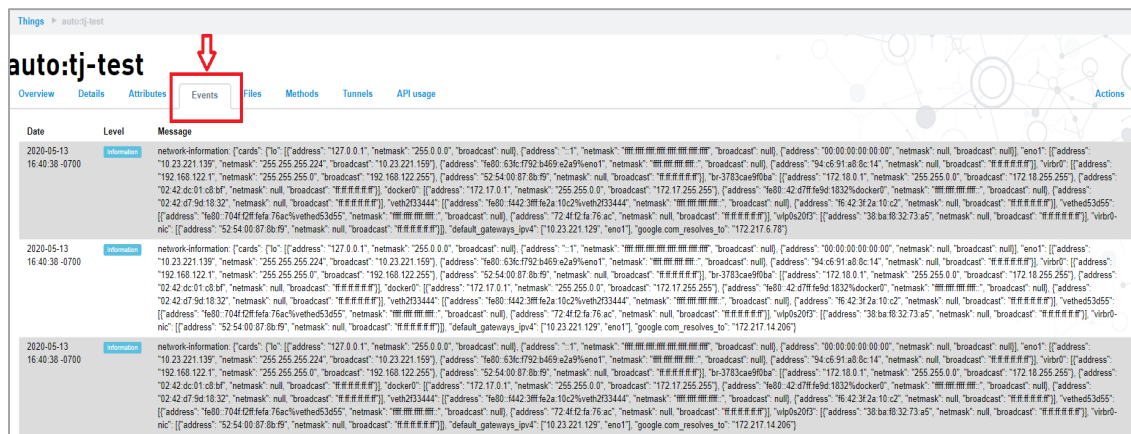
3. Now a set of menus of actions that can be performed on your *Things* definition will be displayed.

Figure 13. Actions



4. The logs are shown under the Events tab under "Thing" as shown in Figure 14.

Figure 14. Logs



Users can monitor the status of each trigger of FOTA or SOTA at a given time for one or more managed devices. More details are covered in [Telemetry Data](#).

**Note:** For all OTA updates via Manifest button, refer to **Developer Guide** documentation.

## 3.0 OTA Updates

---

After the Intel® In-Band Manageability Framework running on the Edge IoT Device is provisioned, it will establish a secure session with the Telit\* portal and the Device shall be visible as “Connected” Thing. Refer to [Section 2.8](#).

Users shall be able to perform the updates listed below on the device that is provisioned:

- AOTA (Application Over the Air update)
- FOTA (Firmware-over-the-Air update)
- SOTA (Software/OS-over-the-Air update)
- Config Update (configuration parameter update)
- Power Management (Remote Shutdown and Restart)

### 3.1 Trusted Repositories

As part of a security measure, the Intel® In-Band Manageability requires the Server URL (location) of the OTA update repository to be included in a “trusted repository list” which is maintained internally. Hence, it is mandatory that the OTA URL is included in the “trusted repository list” to initiate an OTA command. This can be achieved via OTA configuration Append command to add a new Server URL the existing Trusted Repository list.

**IMPORTANT NOTE:** Ensuring the OTA packages are hosted in secure repositories is critical task that customers need to ensure and is outside the scope of INBM.

OTA Configuration Update: Refer to [Configuration Append](#) for adding the Server URL in the trustedRepositories via 'Trigger Config Update'.

For manual update of configuration parameters, refer to **Developer Guide**.

## 3.1 Preparing OTA Update Packages

Before updates can be dispatched to the endpoint, some preparation needs to be done at the repository server to facilitate the updates.

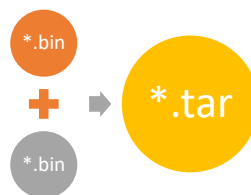
### 3.1.1 Creating FOTA Package

The FOTA package structure changes when signature is used. For a more secure FOTA update, users can include a PEM file containing the signing key to validate the downloaded file against a signature provided as part of the OTA command, refer to [How to generate Signature](#). Users may create a PEM file using the OpenSSL and Cryptography libraries.

1. **With Signature:** FOTA package structure with signature only accepts a *tar* (archive) file as a FW update package. The *tar* file should consist of the firmware update binary (e.g., \*.bin, \*.cap, and so on) file as a capsule. In addition, a signature file should also be included as additional security.

Archiving the \*.bin file with a *tar* archive tool can be performed with the command below:

```
$ tar cvf ifwi_update.tar ifwi_update.bin signing_cert.pem
```



2. **Without Signature:** FOTA package structure without signature only accepts a single firmware update binary (e.g., \*.bin, \*.cap, and so on) file as a capsule.





### 3.1.2 Creating SOTA Package

SOTA on the Ubuntu\* operating system does not require any SOTA package.

SOTA on Yocto\* is handled by INB-based on OS implementation:

1. Debian package manager: It does not require any SOTA package creation but instead requires the APT repositories set correctly and path included in the apt resources.
2. Mender.io: These involve OS update images, also known as **mender artifacts**, generated by the build infrastructure. More information on mender integration can be found at <https://docs.mender.io>.

### 3.1.3 Creating AOTA Package

AOTA Package structure for the commands below should use the following format.

**Table 4. Creating AOTA Package**

AOTA Command	AOTA Package structure
AOTA Docker-Compose package (Same format for up/pull)	<p>Container Tag == Container Image Name  <b>Example:</b> The container Image name and the tar file name should be the same  <b>Container Tag</b> =CPU  <b>Tar file</b> = CPU.tar.gz  <b>Note:</b> The Tar file should contain a folder with the same name CPU. This folder CPU, needs to have the <i>docker-compose.yml</i> file.</p> <p><b>Steps:</b></p> <ol style="list-style-type: none"> <li>1.Make a folder</li> </ol> <pre>\$ mkdir CPU</pre> <ol style="list-style-type: none"> <li>2.Copy the <i>docker-compose.yml</i> file into the folder</li> </ol> <pre>\$ cp docker-compose.yml CPU/.</pre> <ol style="list-style-type: none"> <li>3.Tar the folder</li> </ol> <pre>\$ tar -cvf CPU.tar.gz CPU</pre>

AOTA Docker Load/Import	<p>Package needs to be in <i>tar.gz</i> format</p> <p>The package needs to have a folder within with the same name as the package.</p>
-------------------------	--

### 3.1.4 Creating Configuration Load Package

The Configuration load package structure changes when signature field is used. For a more secure OTA update, users can include a PEM file containing the signing key to validate the downloaded file against a signature provided as part of the OTA command, refer to [How to Generate Signature](#). Users may create a PEM file using the OpenSSL and Cryptography libraries.

1. Configuration Load package structure **with signature** contains a tar file with the *intel\_manageability.conf* file and the *signature\_cert.pem* file. Archiving the *intel\_manageability.conf* file with a *tar* archive tool can be performed with the command below:

```
$ tar cvf conf_update.tar intel_manageability.conf signing_cert.pem
```

2. Configuration Load package structure with **no signature** only contains *intel\_manageability.conf* file.

## 3.2 How to Generate Signature

We can use signature field, by having the *succeed\_rpm\_cert.pem* file within the tar package for FOTA/Config Load. A signature needs to be given in the form before clicking 'Execute' to trigger OTA.

The signature can be generated using OpenSSL, or Cryptography libraries along with the *key.pem* file. Below is the process to generate this signature.

Please find the *create\_signature.py* and *succeed\_rpm\_cert.pem* files in the Intel® In-Band Manageability Framework release package.

```
$python3.6 create_signature.py succeed_rpm_cert.pem firmware_file.bin 1234
```

The output from the above command is used as the input to the signature field while triggering OTA FOTA/Config Load.

## 3.3 OTA Commands

To trigger OTA commands on the device provisioned with Telit\*, navigate to the **Commands** tab of the device on the portal as stated in [Section 2.5.2](#).

**Table 5. Commands - Definitions and Usage**

Command	Definition
<a href="#">Trigger AOTA</a>	Remotely launch/update docker containers on the Edge IoT Device
<a href="#">Trigger FOTA</a>	Update the BIOS firmware on the system
<a href="#">Trigger SOTA</a>	User-friendly, parameter driven updates to OS software packages on the system
<a href="#">Trigger Config Update</a>	Update the Intel® In-Band Manageability configurations
<a href="#">Reboot</a>	Remotely reboot the Endpoint
<a href="#">Shutdown</a>	Remotely shutdown the Endpoint
<a href="#">Decommission</a>	Decommission a device from cloud
<a href="#">Manifest Update</a>	Any OTA update type can be done via the Manifest Update, by entering XML text to update the Endpoint. <b>(Refer to <a href="#">Developer Guide</a>)</b>

## 3.4 AOTA Updates

Supported AOTA commands and their functionality:

**'docker-compose'** commands currently supported:

**Table 6. 'docker-compose' commands**

<b>'docker-compose' Command</b>	<b>Definition</b>
<a href="#">Up</a>	Deploying a service stack on the device
<a href="#">Down</a>	Stopping a service stack on the device
<a href="#">Pull</a>	Pulls an image or a repository from a registry
<a href="#">List</a>	Lists containers
<a href="#">Remove</a>	Removes docker images from the system

**'docker'** commands currently supported:

**Table 7. 'docker' commands**

<b>'docker' Command</b>	<b>Definition</b>
<a href="#">Import</a>	Importing an image to the device
<a href="#">Load</a>	Loading an image from the device
<a href="#">Pull</a>	Pulls an image or a repository from a registry
<a href="#">Remove</a>	Removes docker images from the system
<a href="#">Stats</a>	Returns a live data stream for all the running containers

**'application'** command currently supported:

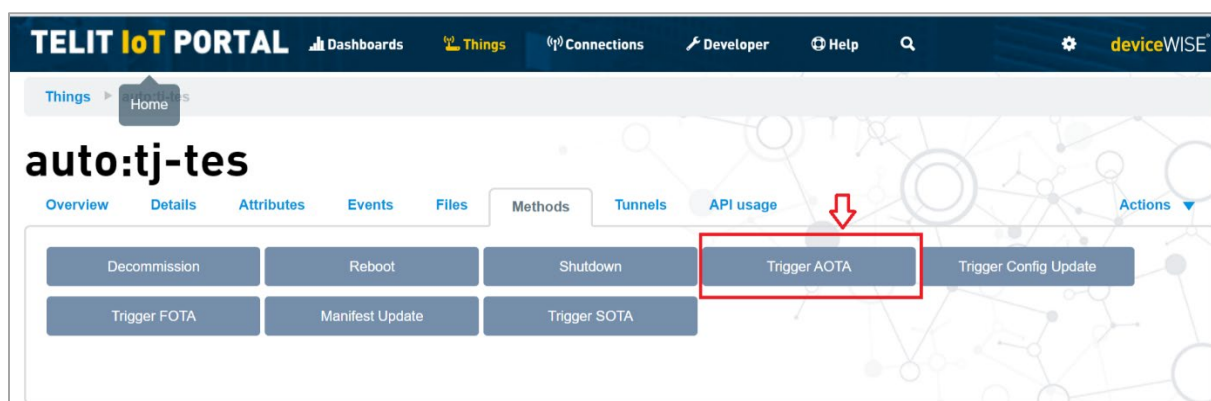
<b>'application' Command</b>	<b>Definition</b>
<a href="#">Update</a>	Updating an application package

**Table 8. List of AOTA commands NOT supported**

Docker-Compose	Import
	Load
	Update
	Stats
Docker	Up
	Down
	Update
	List
Application	Up
	Down
	List
	Remove
	Pull
	Load
	Stats
	Import

In order to trigger Application-over the Air updates, select your Edge Device and click on the **Methods** tab, followed by clicking the **Trigger AOTA** button as shown in Figure 15.

**Figure 15. Trigger AOTA**



Now, populate the AOTA pop-up window with the parameters as shown in the Table 9 and then click **Execute** to trigger the AOTA update.

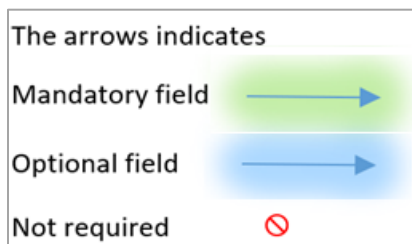
The AOTA form has fields below:

**Table 9. AOTA Field Description**

Field	Input description
App and Its command	Docker-Compose supports: Up, Down, Pull, List and Remove. Docker supports: Load, Import, Pull, Remove and Stats.
Container Tag	Name tag for image/container.
Docker Compose File	Field to specify the name of custom yaml file for docker-compose command. Example: custom.yml
Fetch	Server URL to download the AOTA container tar.gz file.  If the server requires username/password to download the file, you can provide the server username/ server password.  <b>NOTE:</b> Follow <a href="#">Creating AOTA Package</a> to build the package.
Server Username/ Server Password	If server where we host the package to download AOTA file needs credentials, we need to specify the username and password.
Docker Registry Docker Registry Username/Password	Specify Docker Registry if accessing any registry other than the default 'index.docker.io'. Example for docker Registry: amr-registry-pre.caas.intel.com  Optional fields Docker Registry Username/Password can be used to when using private images in AOTA through docker pull and docker-compose up, pull commands.

**Note:** Following sections demonstrate what fields to fill for respective AOTA operations with required and optional fields.

**Figure 16. Fields Indicator**



For each of the AOTA functions, insert the correct parameters as described and click on **Execute** button. To view the results of an AOTA command, navigate to the **Events** tab.

### 3.4.1 AOTA Docker-Compose Operations

#### 3.4.1.1 Docker-Compose Up

##### NOTES:

1. The Container Tag name should be same as the file name in the fetch field.  
Example: Container Tag: CPU Downloaded fetch file: CPU.tar.gz.
2. Docker-Compose yml file should have the correct docker version.

Figure 17. Docker-Compose Up

**Trigger AOTA**

App:

Command:

Container Tag:

Fetch:

Docker Compose File:

Signature:

Version:

Server username:

Server password:

Docker Registry:

Docker Registry Username:

Docker Registry Password:



### 3.4.1.2 Docker-Compose Down

Figure 18. Docker-Compose Down

**Trigger AOTA**

App: docker-compose

Command: down

Container Tag:

Fetch:

Docker Compose File:

Signature:

Version:

Server username:

Server password:

Docker Registry:

Docker Registry Username:

Docker Registry Password:

Output:

### 3.4.1.3 Docker-Compose Pull

**NOTE:** The Container Tag name should be same as the file name in the fetch field. Example:  
Container Tag: CPU Downloaded fetch file: CPU.targ.gz

Figure 19. Docker-Compose Pull

**Trigger AOTA**

App: docker-compose

Command: pull

Container Tag: mysql

Fetch: https://ubit-artifactory-or.intel.com/artifactory/mysql.tar.gz

Docker Compose File:

Signature:

Version:

Server username: xxxxx

Server password: xxxxx

Docker Registry: amr-registry-pre.caas.intel.com

Docker Registry Username: xxx

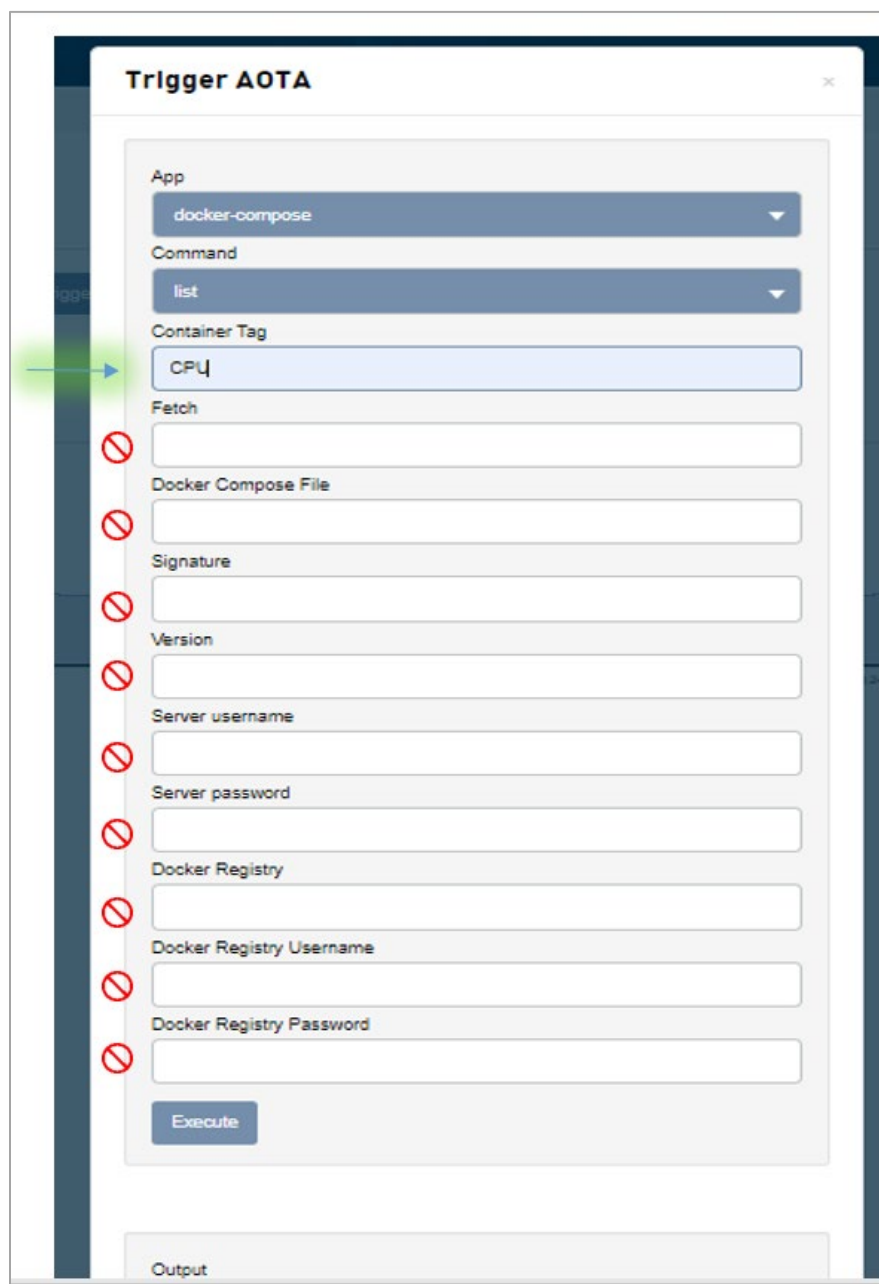
Docker Registry Password: xxx

Execute

Output

### 3.4.1.4 Docker-Compose List

Figure 20. Docker-Compose List



**Trigger AOTA**

App: docker-compose

Command: list

Container Tag: CPU

Fetch: ⊘

Docker Compose File: ⊘

Signature: ⊘

Version: ⊘

Server username: ⊘

Server password: ⊘

Docker Registry: ⊘

Docker Registry Username: ⊘

Docker Registry Password: ⊘

Execute

Output

### 3.4.1.5 Docker-Compose Remove

Figure 21. Docker-Compose Remove

**Trigger AOTA**

App: docker-compose

Command: remove

Container Tag: CPU

Fetch:

Docker Compose File:

Signature:

Version:

Server username:

Server password:

Docker Registry:

Docker Registry Username:

Docker Registry Password:

Execute

Output

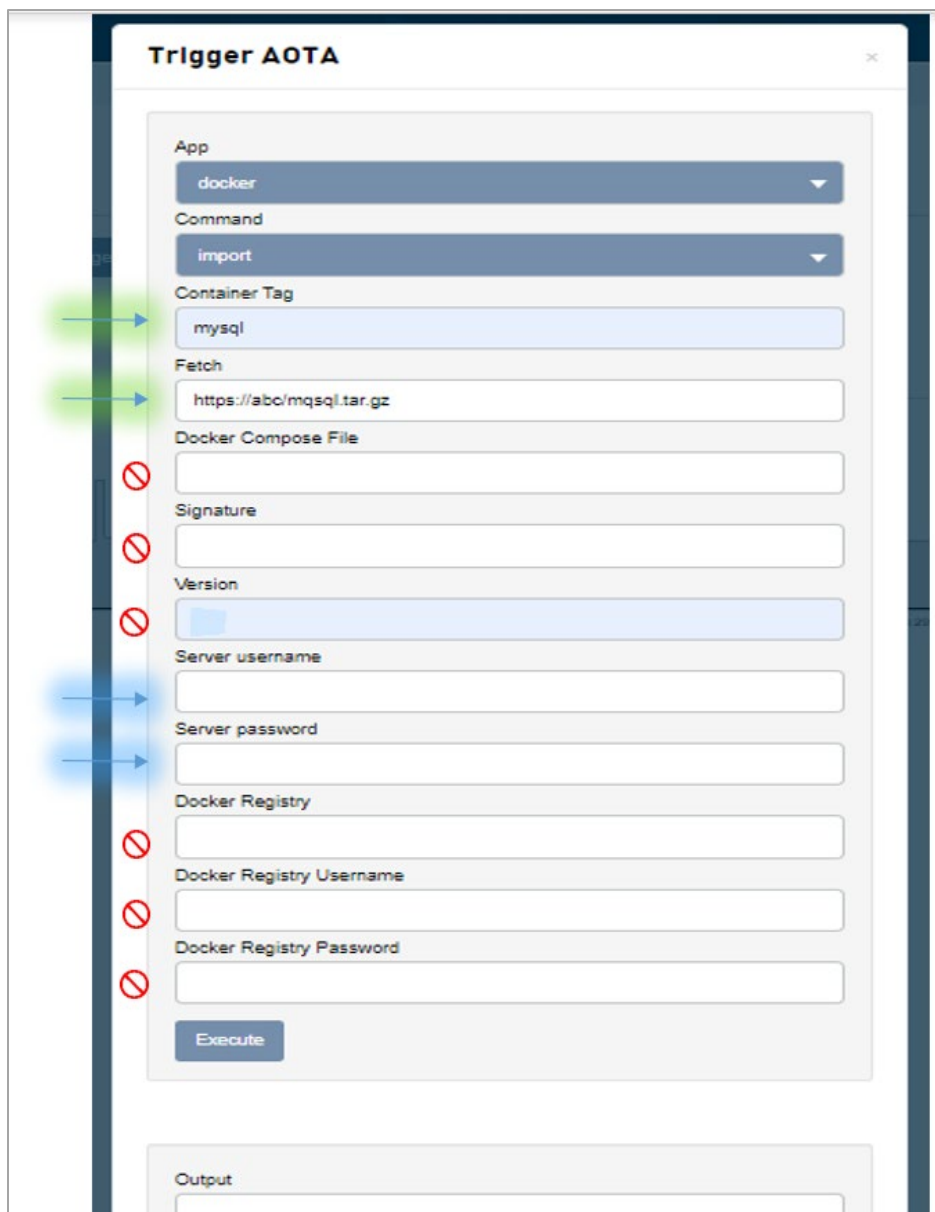
## 3.4.2 AOTA Docker Operations

### 3.4.2.1 Docker Import

**NOTE:** The Container Tag name should be same as the file name in the fetch field.

Example: Container Tag: CPU, Downloaded fetch file: CPU.targ.gz

Figure 22. Docker Import



**Trigger AOTA**

App:

Command:

Container Tag:

Fetch:

Docker Compose File:

Signature:

Version:

Server username:

Server password:

Docker Registry:

Docker Registry Username:

Docker Registry Password:

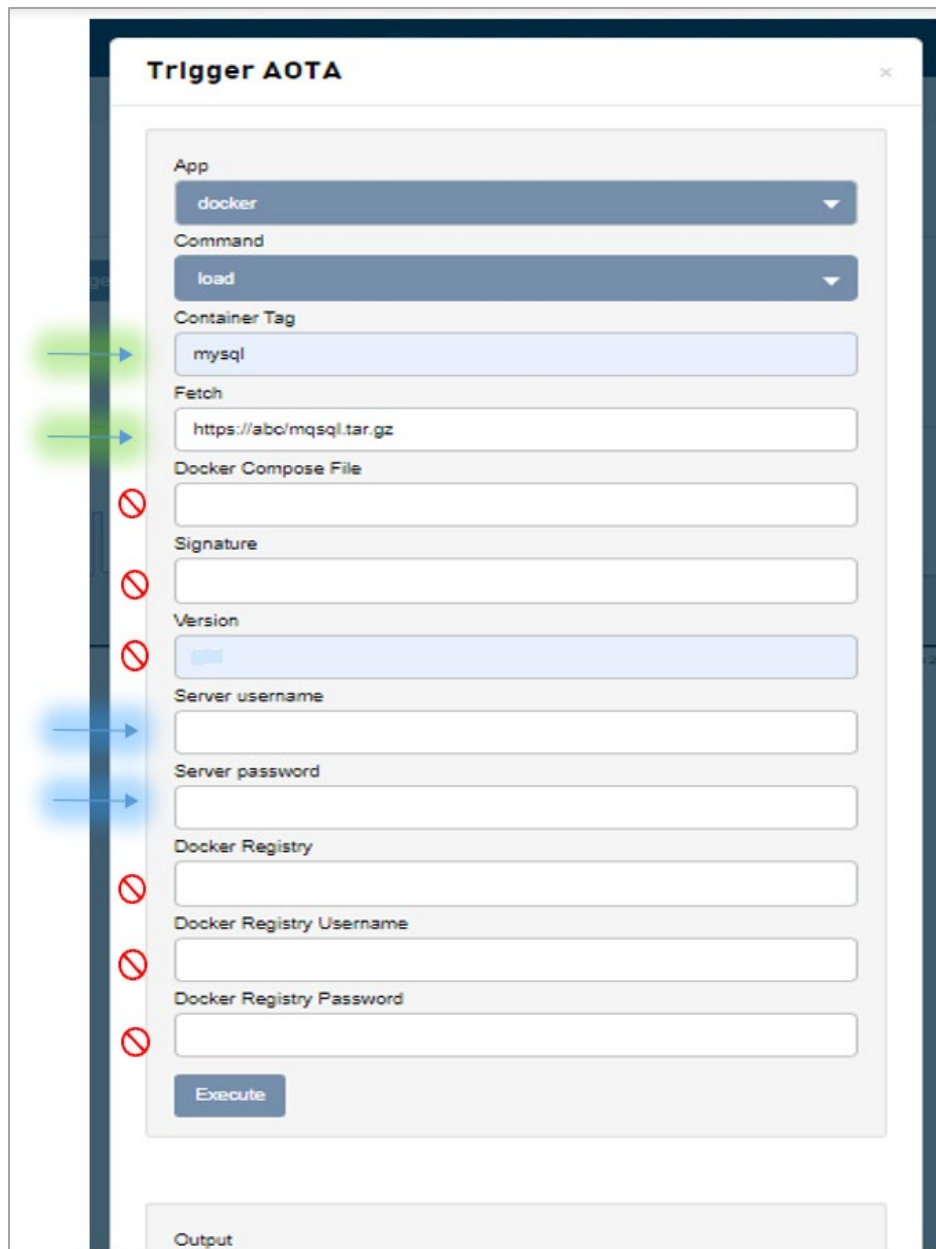
Output:

### 3.4.2.2 Docker Load

**NOTE:** The Container Tag name should be same as the file name in the fetch field.

Example: Container Tag: CPU, Downloaded fetch file: CPU.targ.gz

Figure 23. Docker Load



**Trigger AOTA**

App: docker

Command: load

Container Tag: mysql

Fetch: https://abo/mqsql.tar.gz

Docker Compose File:

Signature:

Version:

Server username:

Server password:

Docker Registry:

Docker Registry Username:

Docker Registry Password:

Execute

Output

### 3.4.2.3 Docker Pull

Figure 24. Docker Pull

**Trigger AOTA**

App:

Command:

Container Tag:

Fetch:

Docker Compose File:

Signature:

Version:

Server username:

Server password:

Docker Registry:

Docker Registry Username:

Docker Registry Password:

### 3.4.2.4 Docker Remove

Figure 25. Docker Remove

**Trigger AOTA**

App: docker-compose

Command: remove

Container Tag: CPU

Fetch:

Docker Compose File:

Signature:

Version:

Server username:

Server password:

Docker Registry:

Docker Registry Username:

Docker Registry Password:

Execute

Output



### 3.4.2.5 Docker Stats

Figure 26. Docker Stats

Trigger AOTA

App

docker

Command

stats

Container Tag

Fetch

Docker Compose File

Signature

Version

Server username

Server password

Docker Registry

Docker Registry Username

Docker Registry Password

Execute

Output

### 3.4.3 AOTA Application Operations

#### 3.4.3.1 Application Update

**NOTE:** The Device Reboot command is an optional field.

For any Xlink driver update, it is mandatory to reboot the device.

Input **yes** for Device Reboot as shown in Figure 27.

You can only use signed packages to update Xlink Driver application.

**Figure 27. Application Update**

**Trigger AOTA**

App  
application

Command  
update

Container Tag

Device Reboot  
yes

Fetch  
<https://af01p-png.devtools.intel.com/artifactory/bit-creek-local-public-png-lor>

Docker Compose File

Signature

Version

Server username

Server password

Docker Registry

Docker Registry Username

Docker Registry Password

Execute

## 3.5 FOTA Updates

To perform FOTA updates, IBVs must supply the SMBIOS or Device Tree info that is unique to each platform SKU and fulfill the vendor, version, release date, manufacturer and product name that matches the endpoint as shown in Table 10.

**Note:** The following information must match the data sent in the FOTA update command for Intel® In-Band Manageability Framework to initiate a Firmware update process.

**Table 10. FOTA Update Info**

Information	Field	Checks
FW	Vendor	Checks for string match between the user input and platform vendor
	Version	
	Release Date	Checks if the current firmware file release date is newer than release date on the platform
System	Manufacturer	Checks for a string match between the user input and platform manufacturer
	Product Name	Checks for string match between the user input and platform product name

To find the FW and System fields at the endpoint, run the commands below:

### Intel® x86 UEFI-based products

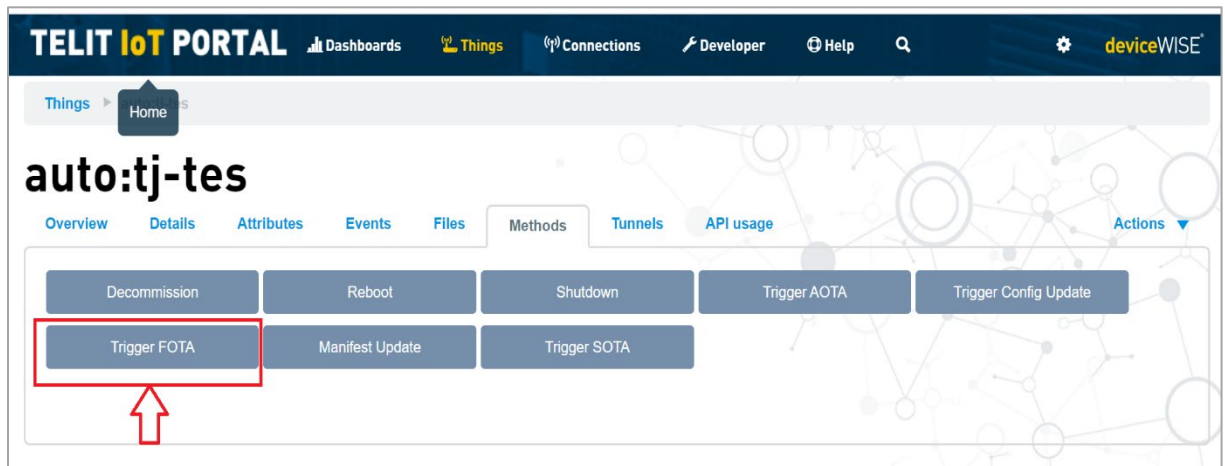
For UEFI based platforms the Firmware and system information can be found by running the following command.

```
$ sudo dmidecode -t bios -t system
```

### 3.5.1 FOTA Update via Button Click:

1. Go to the **Things** tab and select the device on which FOTA needs to be triggered.
2. Select the **Methods** tab and then click the **Trigger FOTA** button.

Figure 28. Trigger FOTA



3. Populate the FOTA pop-up window with the parameters in the table below.

**Note:** If triggering a secure FOTA update with a \*.pem file within the *tar*, a signature needs to be given in the respective field. The signature can be generated using OpenSSL, or Cryptography libraries along with the key.pem file.

4. Click **Execute** to trigger the FOTA update.

**Figure 29. Trigger Fota**

The screenshot shows a web form titled "Trigger FOTA". It contains the following fields from top to bottom: "Bios Version", "Fetch", "Manufacturer", "Path", "Product", "Release Date", "Signature", "ToolOptions", "Vendor", "Server username", and "Server password". Each field has a corresponding indicator on the left side of the form. A vertical bar on the left contains arrows pointing to each field. The arrows for "Bios Version", "Fetch", "Manufacturer", "Product", "Release Date", "Vendor", and "Server username" are green, indicating they are mandatory fields. The arrows for "Signature", "ToolOptions", "Server username", and "Server password" are blue, indicating they are optional fields. The arrow for "Path" is a red circle with a diagonal line through it, indicating it is not required. At the bottom of the form is an "Execute" button.

The arrows indicates	
Mandatory field	
Optional field	
Not required	

Table 11. Parameters

Parameter	Description
BIOSVersion	Verify with BIOS Vendor (IBV)
Fetch	Repository URL <b>NOTE:</b> Follow <a href="#">Creating FOTA Package</a> to build package.
Manufacturer	Endpoint Manufacturer Name
Path	FOTA path created in repository
Product	Product name set by Manufacturer
Release Date	Specify the release date of the BIOS file you are applying
Release Date	Verify with BIOS Vendor (IBV)
Signature	Digital signature
ToolOptions	Any Tool options to be given for the Firmware Tool
Server Username/Password	If server where we host the package to download FOTA file needs credentials, we need to specify the username and password

## 3.6 SOTA Updates

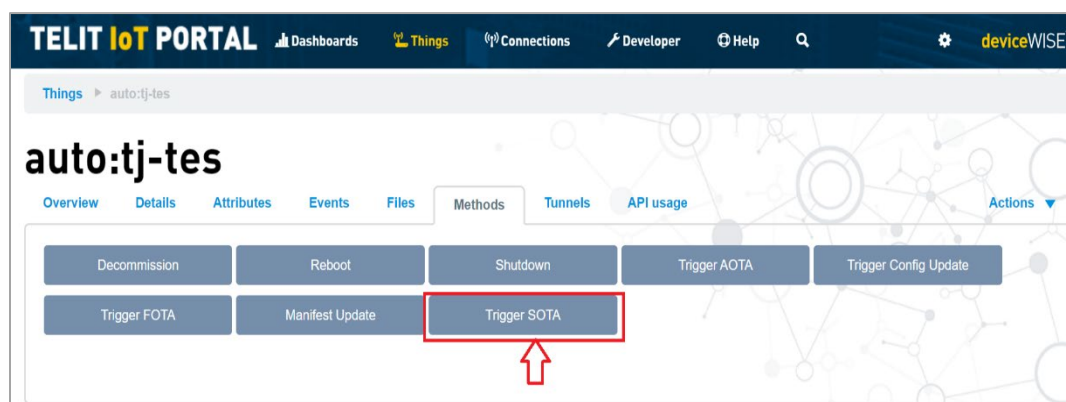
SOTA commands vary based on OS type and update mechanisms supported by it.

The Ubuntu\* OS or Yocto\*-based OS, which include the Debian package manager do not require any package preparation, while a Yocto\*-based OS with Mender.io based solution does. This changes the interface slightly as explained below.

### 3.6.1 SOTA Update Via Button Click (Debian Package Manager and Ubuntu\*)

1. Go to the **Things** tab on the upper top in the Telit DeviceWISE\* portal.
2. Click on the **connected** tab and select the device which the SOTA needs to be triggered.
3. Select the **Methods** tab and then click on the **Trigger SOTA** button.

Figure 30. Trigger SOTA



4. Populate the SOTA pop-up screen with **Log to File** as **No** to have logs written to the cloud. Otherwise, select **Yes** to have logs to be written to a file. SOTA log files can be located at the endpoint `/var/cache/manageability/repository-tool/sota/`.

- Click **Execute** to trigger the SOTA update.

**Figure 31. Trigger SOTA**

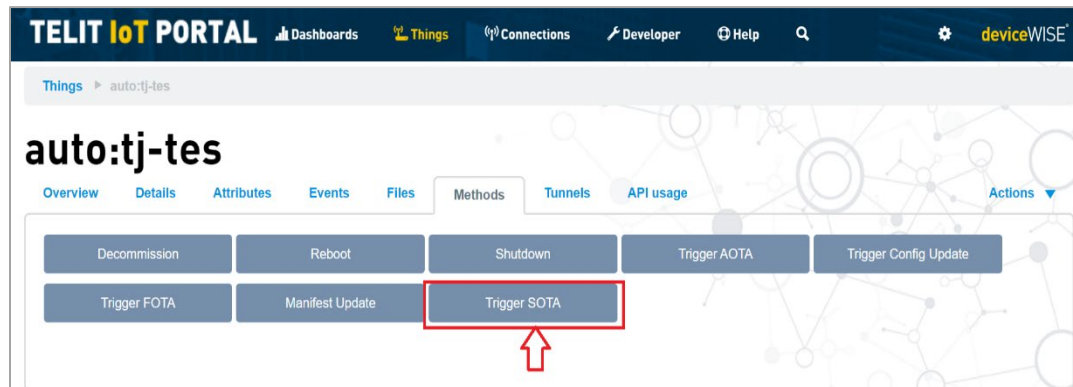
The arrows indicates	
Mandatory field	
Optional field	
Not required	



### 3.6.2 SOTA Update Via Button Click (Mender)

1. Go to **Things** tab located at the top-left of the Telit DeviceWISE\* portal.
2. Click on the connected tab and select the device which SOTA needs to be triggered.
3. Select the **Methods** tab and then click on the **Trigger SOTA** button.

**Figure 32. Trigger SOTA**



4. Populate the SOTA pop-up screen with **Log to File** as **No** to have logs will be written to cloud, otherwise **Yes** to have logs to be written to a file. SOTA log files can be located at the endpoint `/var/cache/manageability/repository-tool/sota/`
5. Click **Execute** to trigger the SOTA update.

Figure 33. Trigger SOTA

The arrows indicates

Mandatory field	
Optional field	
Not required	

## 3.7 Configuration Update

Configuration update is used to update, retrieve, append and remove configuration parameter values from the Configuration file located at `/etc/intel_manageability.conf`. Refer to Table 12 to understand the configuration tags, its values and the description.

**Table 12. Default Configuration Parameters**

Telemetry		
Collection Interval Seconds	60 seconds	Time interval after which telemetry is collected from the system.
Publish interval seconds	300 seconds	Time interval after which collected telemetry is published to dispatcher and the cloud.
Max Cache Size	100	Maximum cache set to store the telemetry data. This is the count of messages that telemetry agent caches before sending out to the cloud.
Container Health Interval Seconds	600 seconds	Interval after which container health check is run and results are returned.
Diagnostic Values		
Min Storage	100 MB	Value of minimum storage that the system should have before or after an update.
Min Memory	200 MB	Value of minimum memory that the system should have before or after an update.
Min Power Percent	20%	Value of minimum battery percent that the system should have before or after an update.
Mandatory SW	docker, trtl, telemetry	List of software that should be present and are checked for.
Docker Bench Security Interval Seconds	900 seconds	Time interval after which DBS will run and report back to the cloud.
Network Check	True	This configures network check on the platforms based on their Ethernet capability.
Dispatcher Values		
DBS Remove Image on Failed Container	False	Specifies if the image should be removed in the event of a failed container as flagged by DBS.
Trusted Repositories		List of repositories that are trusted and packages can be fetched from them.
SOTA Values		
Ubuntu Apt Source	Repository link	Location used to update Debian packages.

Proceed Without Rollback	True	Whether SOTA update should go through even when rollback is not supported on the system.
--------------------------	------	--

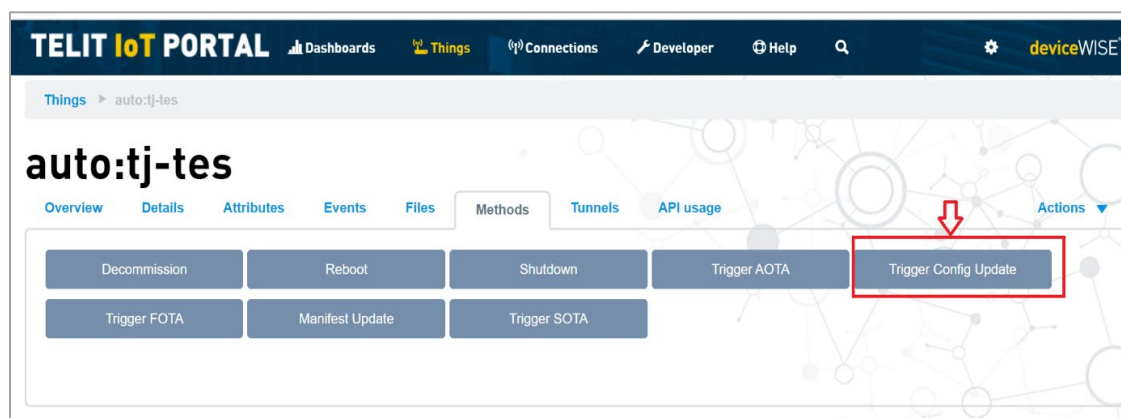
**Table 13. Configuration Update Commands and Input Field Description**

Trigger Configs	Description of field
Command	<b>Set:</b> Command used to update the configuration value using <b>key:value</b> pair. <b>Get:</b> Command used to retrieve a specific configuration value using <b>key:value</b> pair. <b>Load:</b> Command used to replace an entire configuration file. <b>Append:</b> Command used to append values to a configuration parameter. <b>Remove:</b> Command used to remove a specific value from the configuration parameter.
Fetch	The URL to fetch config file from in the case of a load.
Path	Specifies the path of element to get, set, append or remove in <b>key:value</b> format.
Signature	Digital signature refers to [TBD]

**To trigger a configuration update, follow the steps below:**

1. Go to the **Things** tab and select the device on which Config update needs to be triggered.
2. Select the **Methods** tab and then click the **Trigger Config Update** button as shown in Figure 34.

**Figure 34. Trigger Config Update**

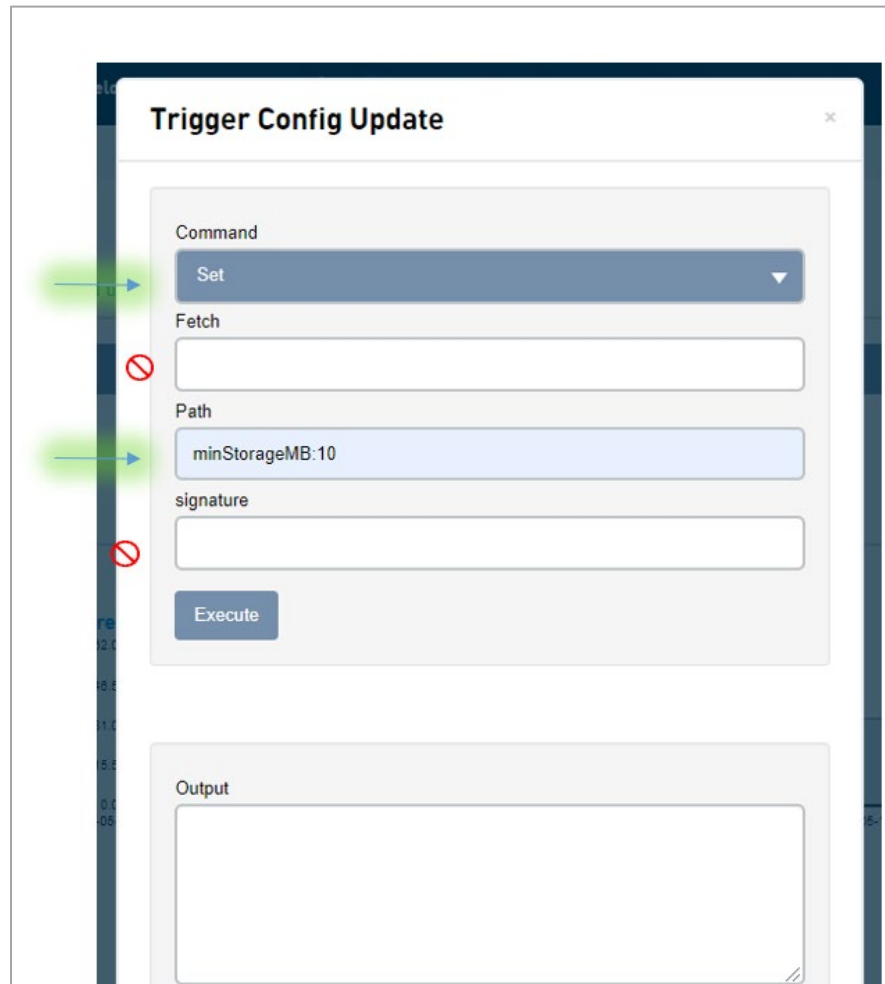


3. Populate the Config Update pop-up window with required parameters.

**Note:** If triggering a secure Config update load with a \*.pem file within the *tar*, a signature needs to be given in the respective field. The signature can be generated using OpenSSL, or Cryptography libraries along with the key.pem file.

4. Click **Execute** to trigger the Config update.

**Figure 35. Execute Config Update**



**Trigger Config Update**

Command: Set

Fetch:

Path: minStorageMB:10

signature:

Execute

Output

## 3.7.1 Configuration Update Via Button Click

### 3.7.1.1 Configuration Set

#### Examples:

To set a configuration value, input for path field in key:value format-> **minStorageMB:10**

To set multiple values at once use ;

Example to separate key:value pairs-> **minStorageMB:10;minMemoryMB:250**

**NOTE:** Field path takes in key:value pairs as an input with key as the configuration parameter tag and value as the value to be updated.

#### Results:

The configuration contents inside the file before and after the update are shown below.

Before update	After update
<pre>&lt;minStorageMB&gt;100&lt;/minStorageMB&gt; &lt;minMemoryMB&gt;200&lt;/minMemoryMB&gt;</pre>	<pre>&lt;minStorageMB&gt;10&lt;/minStorageMB&gt; &lt;minMemoryMB&gt;250&lt;/minMemoryMB&gt;</pre>

### 3.7.1.2 Configuration Get

#### Examples:

To get one configuration value, use configuration tag as input for path-> **minStorageMB**

To get multiple values at once use ; to separate tags-> **minStorageMB;minMemoryMB**

**NOTE:** Field path takes keys as an input with key as the configuration parameter tag whose value to be retrieved. Also, to retrieve multiple values at once use, to separate one tag from another as shown above in the example.

Figure 36. Configuration Get

The screenshot shows a 'Trigger Config Update' window. It has a 'Command' dropdown menu with 'Get' selected. Below it is a 'Fetch' field, which is empty and has a red 'X' icon next to it. Then there is a 'Path' field containing the text 'minStorageMB'. Below that is a 'signature' field, also empty, with a red 'X' icon next to it. At the bottom of the form is an 'Execute' button. Below the form is an 'Output' section with a large, empty text area for displaying results.

Results of Configuration Get update can be seen in the Telit\* cloud under events tab:

Figure 37. Events Tab

auto:tj-tes

Overview

Details

Attributes

Events

Files

Methods

Tunnels

API usage

Date	Level	Message
2020-05-14 09:36:31 -0700	Information	{ "status": 200, "message": "Configuration update: successful" }
2020-05-14 09:36:31 -0700	Information	Got response back for command: get_element response: (diagnostic/minMemoryMB:200).
2020-05-14 09:36:30 -0700	Information	Got response back for command: get_element response: (diagnostic/minStorageMB:100).
2020-05-14 09:36:30 -0700	Information	Command: check_network passed. Message: At least one network interface is healthy (has a default route).
2020-05-14 09:36:29 -0700	Information	Configuration Method Triggered

More events

### 3.7.1.3 Configuration Load

**NOTE:** Follow [Creating Configuration Load Package](#) to build package.

Figure 38. Configuration Load

Trigger Config Update

Command

Load

Fetch

http://134.134.155.57:8000/conf.tar

Path

signature

Execute

Output



### 3.7.1.4 Configuration Append

#### NOTES:

1. Append is only applicable to three configuration tags i.e *trustedRepositories*, *sotaSW* and *ubuntuAptSource* from the configuration file.
2. Field path takes in key:value pair format.

**Example:** If you need to add a new Server URL to download OTA package, path is *trustedRepositories:https://af01p-igk.devtools.intel.com/artifactory/turtle-creek/test/*

**Figure 39. Configuration Append**

The screenshot shows a 'Trigger Config Update' dialog box. It has a 'Command' dropdown menu set to 'Append'. Below it are four input fields: 'Fetch' (empty, with a red 'X' icon), 'Path' (containing 'trustedRepositories:http://abc.com/'), 'signature' (empty, with a red 'X' icon), and an 'Execute' button. At the bottom right, there is an 'Output' section with a large empty text area.

**End Results of Configuration Append:**

Before configuration append update.

```
<trustedRepositories>
  https://af01p-igk.devtools.intel.com/artifactory/SID-Docker-local/bmp/test
  https://ubit-artifactory-or.intel.com/artifactory/iotg-bmp-internal-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-test-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-igk-local/
  http://ci_nginx:80
</trustedRepositories>
```

After configuration append update, you can see the value appended below.

```
<trustedRepositories>
  https://af01p-igk.devtools.intel.com/artifactory/SID-Docker-local/bmp/test
  https://ubit-artifactory-or.intel.com/artifactory/iotg-bmp-internal-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-test-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-igk-local/
  http://ci_nginx:80
  ➡ https://af01p-igk.devtools.intel.com/artifactory/turtle-creek/test/
</trustedRepositories>
```

### 3.7.1.5 Configuration Remove

**NOTE:** Remove is only applicable to three configuration tags i.e *trustedRepositories*, *sotaSW* and *ubuntuAptSource*

Field path takes in key value pair format, example: *trustedRepositories:https://af01p-igk.devtools.intel.com/artifactory/turtle-creek/test/*

**Figure 40. Configuration Remove**

Trigger Config Update

Command

Remove

Fetch

Path

trustedRepositories:https://af01p-igk.devtools.intel.com/artifactory/turtle-cre

signature

Execute

Output

## End Results of Configuration Remove:

Before configuration remove trigger (Removing the highlighted value)

```
<trustedRepositories>
  https://af01p-igk.devtools.intel.com/artifactory/SID-Docker-local/bmp/test
  https://ubit-artifactory-or.intel.com/artifactory/iotg-bmp-internal-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-test-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-igk-local/
  http://ci_nginx:80
  https://af01p-igk.devtools.intel.com/artifactory/turtle-creek/test/
</trustedRepositories>
```

After configuration remove update, you can see the value removed:

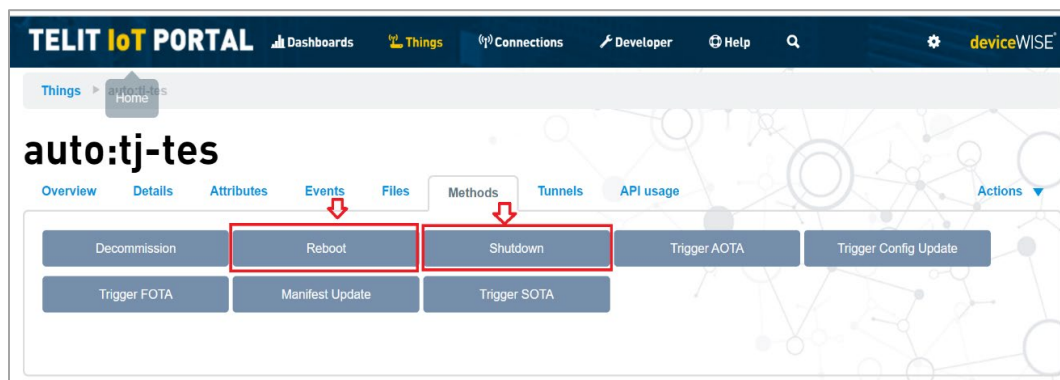
```
<trustedRepositories>
  https://af01p-igk.devtools.intel.com/artifactory/SID-Docker-local/bmp/test
  https://ubit-artifactory-or.intel.com/artifactory/iotg-bmp-internal-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-test-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-igk-local/
  http://ci_nginx:80
</trustedRepositories>
```

## 3.8 Power Management

Shutdown and Restart capabilities are supported via button click or through manifest.

1. Select the device within the *Things* tab and you will see Reboot and Shutdown buttons as shown in Figure 41.

**Figure 41. Reboot and Shutdown**

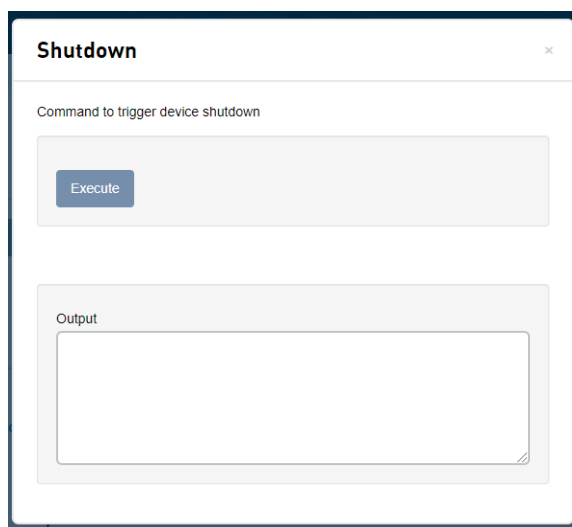


1. Clicking the reboot or shutdown buttons shows screen pop-up as shown below and then clicking the **Execute** button will trigger the respective command on the device.

### 3.8.1 Shutdown

Clicking on the **Execute** button will trigger the respective command on the device.

**Figure 42. Shutdown**

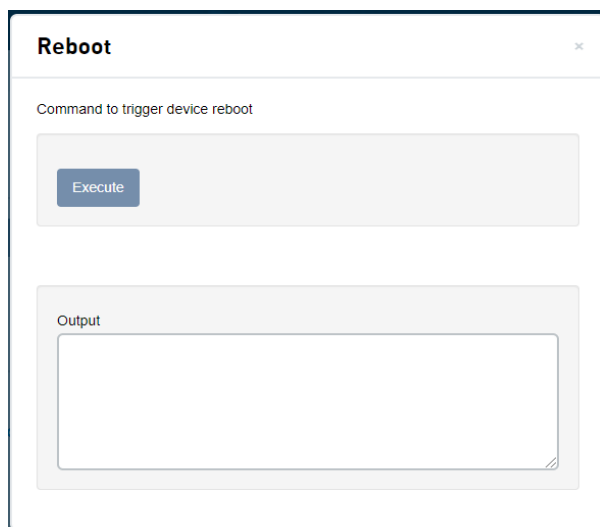


The screenshot shows a window titled "Shutdown" with a close button (X) in the top right corner. Inside the window, there is a label "Command to trigger device shutdown" above a light gray rectangular area. Within this area is a blue button labeled "Execute". Below this area is another light gray rectangular area labeled "Output", which contains a white rectangular box with a small icon in the bottom right corner, representing a text output field.

### 3.8.2 System Reboot

Clicking on the **Execute** button will trigger the respective command on the device.

**Figure 43. Reboot**



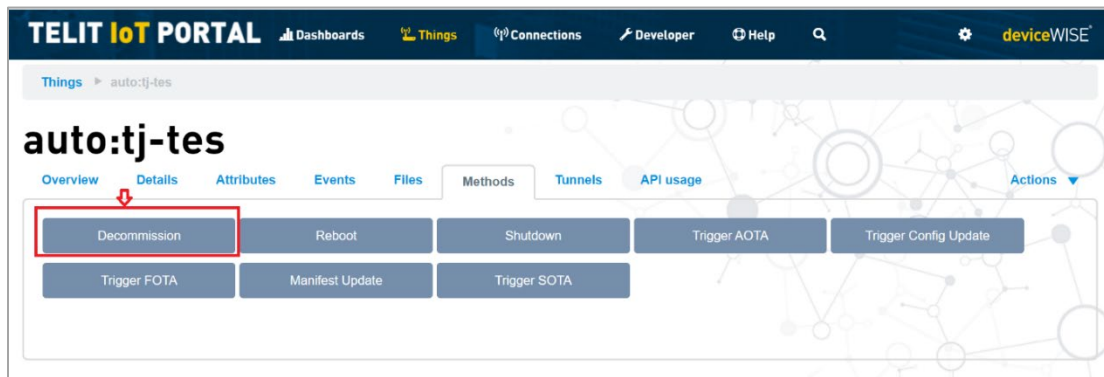
The screenshot shows a window titled "Reboot" with a close button (X) in the top right corner. Inside the window, there is a label "Command to trigger device reboot" above a light gray rectangular area. Within this area is a blue button labeled "Execute". Below this area is another light gray rectangular area labeled "Output", which contains a white rectangular box with a small icon in the bottom right corner, representing a text output field.

## 3.9 Decommission

The Decommission command is used to remove all the credentials and then result in a device shutdown.

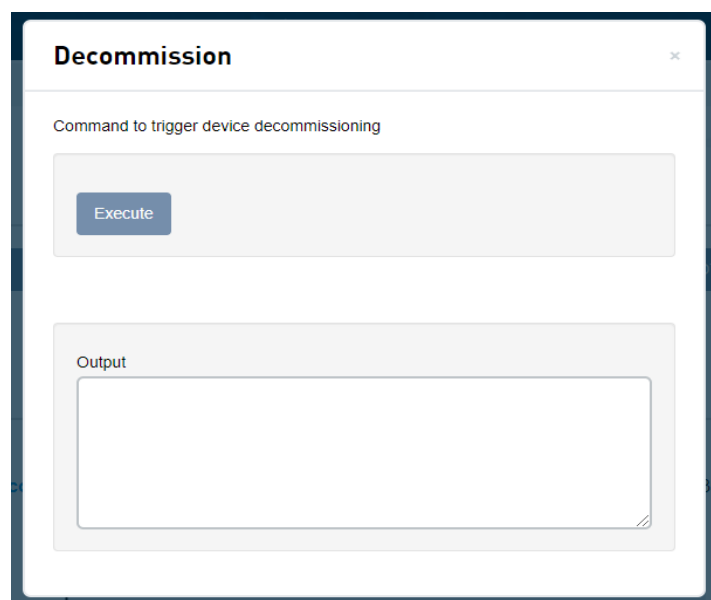
1. To trigger Decommission, select the *Things* tab and you will see **Decommission** button as shown in Figure 44.

Figure 44. Decommission



2. Clicking on the **Execute** button will trigger the respective command on the device.

Figure 45. Decommission



§

## 4.0 Telemetry Data

---

The Intel® In-Band Manageability provides two types of telemetry data, static telemetry and dynamic telemetry. The telemetry data will indicate the health of each endpoint.

### 4.1 Static Telemetry

This contains the following information and can be viewed under the **Attributes** tab for a selected *Thing*.

- BIOS-release-date
- BIOS-vendor
- BIOS-version
- CPU-ID
- OS-information
- System-Manufacturer
- System-Product-Name
- Total-physical-memory
- System-Product-Name

### 4.2 Dynamic Telemetry

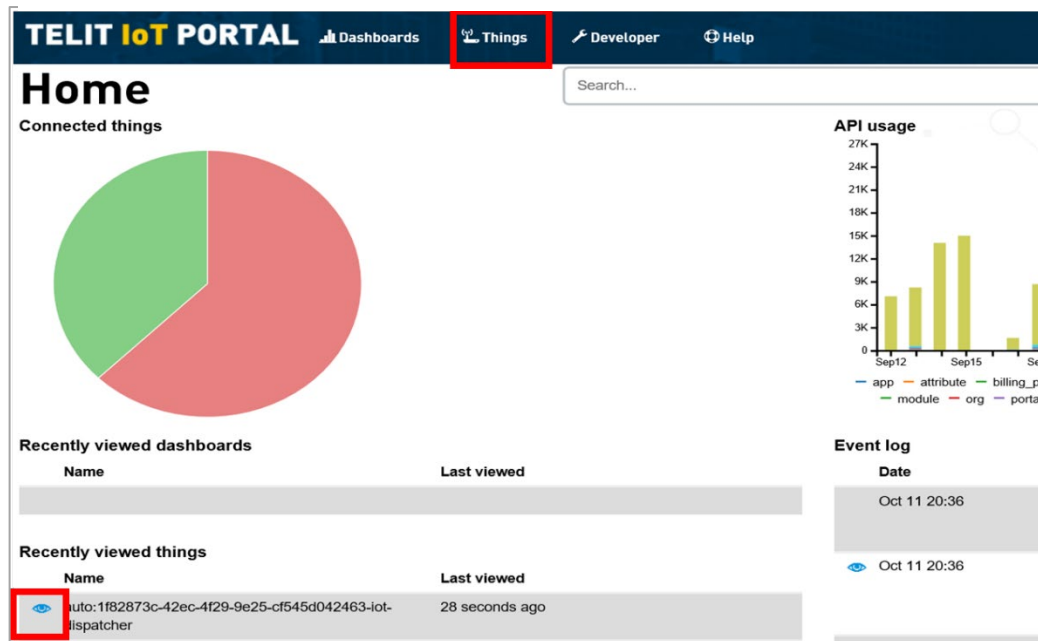
Each endpoint publishes the following Dynamic Telemetry Data in 5-minute intervals.

- Available-memory
- Core-temp-Celsius
- Percent-disk-used
- System-cpu-percent
- Container-stats(cpu-usage)
- Network Information

## 4.3 Viewing Telemetry Data

1. To view telemetry data, click the **Things** tab and then click on the (eye icon) next to respective *Thing* (endpoint).

Figure 46. Things



2. Telemetry data can be viewed at the **Properties** section in the respective *Thing* page. The graphs that you see under **Properties** on your *Thing* page, is the telemetry data for your *Thing*.





## 5.0 Issues and Troubleshooting

### 5.1 OTA Error Status

Table 14. OTA Error Status

Error Message	Description
COMMAND_FAILURE	Diagnostic agent checks fail to run properly or if diagnostic agent/ config agent is not up when contacted. {'status': 301, 'message': 'COMMAND FAILURE'}
COMMAND_SUCCESS	Post and pre-install check go through. {'status': 200, 'message': 'COMMAND SUCCESS'}
FILE_NOT_FOUND	File to be fetched is not found. {'status': 404, 'message': 'FILE NOT FOUND'}
IMAGE_IMPORT_FAILURE	Image is already present when Image Import is triggered. {'status': 401, 'message': 'FAILED IMAGE IMPORT, IMAGE ALREADY PRESENT'}
INSTALL_FAILURE	Installation was not successful due to invalid package or one of the source file, signature or version checks failed. {'status': 400, 'message': 'FAILED TO INSTALL'}
OTA_FAILURE	Another OTA is in progress when OTA is triggered. {'status': 303, 'message': 'OTA IN PROGRESS, TRY LATER'}
UNABLE_TO_START_DOCKER_COMPOSE	Docker compose container is not able to be started or spawned etc. {'status': 400, 'message': "Unable to start docker-compose container."}
UNABLE_TO_STOP_DOCKER_COMPOSE	Docker compose down command was not successful. {'status': 400, 'message': "Unable to stop docker-compose container."}
UNABLE_TO_DOWNLOAD_DOCKER_COMPOSE	Docker compose downloaded command failed. {'status': 400, 'message': "Unable to download docker-compose container."}

XML_FAILURE	Result of bad formatting, missing mandatory tag. {'status': 300, 'message': 'FAILED TO PARSE/VALIDATE MANIFEST'}
-------------	--

## 5.2 Provisioning Unsuccessful or Device not connected to the Cloud

In case if the provisioning script is struck while creating symlinks at the end of provisioning or Device is not connected to the cloud there is a chance that other system services that are waiting might possibly blocked the INB services from starting. In order to fix this issue, follow steps below:

Check if bootup is complete or not using the command:

```
$ sudo system-analyze critical-chain
```

If the boot-up is not complete, list all the jobs:

```
$ sudo systemctl list-jobs
```

Stop all the jobs that are under 'waiting' state:

```
$ sudo systemctl stop <job_unit_name>
```

And try provisioning the device again by following the steps in [Provisioning with Telit DeviceWISE\\* Token.](#)

## 5.3 Acquiring Debug Messages from Agents

Refer to the **Developer Guide Documentation.**

§