SmartHub INFER IoT Center

User Guide

INFER IoT Center 3.0.0

SmartHub.ai

SmartHub Inc.
4332 Holt Street,
Union City, CA, 94587,
USA
www.smarthub.ai

Contents

Introduction to the SmartHub INFER IoT Center

1

SmartHub INFER IoT Center provides an IoT device management capability solution that drives operational and cost efficiency when deploying and managing IoT infrastructure.

SmartHub INFER IoT Center on-boards, configures, manages, monitors, and secures unmanned IoT devices and objects at scale. The solution enables you to pre-register and bulk onboard IoT devices, manage alerts and notifications, troubleshoot, change the configuration of devices, view audit logs, and perform compliance management operations through over the air updates. The functionality of SmartHub INFER IoT Center can be broadly classified into the following three areas:

Monitoring and Alerting

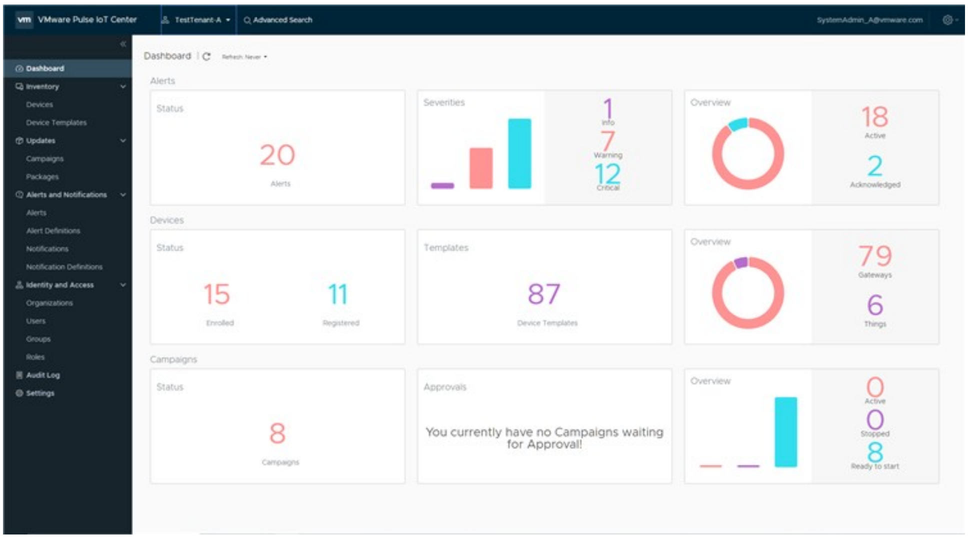
- Metric Monitoring
- Diagnostics, Logging, and Troubleshooting
- Creating Alerts on Static Thresholds
- Alert Aggregation and Clearance

Over The Air (OTA) Campaign

- Software, Firmware, Operating System, and BIOS Updates
- Package Repository and Updates

Device Management and Configuration

- Device Provisioning
- Device Enrollment
- Remote Command Execution
- File Upload to Server from Device
- Gateway Configuration



This chapter includes the following topics:

- Privacy Notice
- Browser Support
- Password Requirements
- Roles and Permissions
- Terminology
- Typical Use Cases

Privacy Notice

SmartHub INFER IoT Center has the ability to collect data from IoT and end-user devices as configured by you. When using SmartHub INFER IoT Center, you are solely responsible for complying with all applicable laws which include, but not limited to, data privacy laws.

You are responsible for providing any necessary notice, and for obtaining any necessary consents, for the data you collect and send to SmartHub INFER IoT Center. For more information about SmartHub's privacy notice, please contact support

Browser Support

The SmartHub INFER IoT Center supports Mozilla Firefox and Google Chrome web browsers.

The SmartHub INFER IoT Center console is tested with the latest versions of the following browsers:

Table 1-1. Supported Browsers

Browser	Platform	Minimum Version
Google Chrome	Microsoft Windows 10, Mac OS Mojave	70.x
Mozilla Firefox	Microsoft Windows 10, Mac OS Mojave	63.x

Note Internet Explorer, Microsoft Edge, and Safari browsers are not supported.

Password Requirements

Your password must meet the following requirements:

- Must be between 8 and 20 characters long.
- Must contain one numeral from *0* to *9*.
- Must contain one lowercase letter from *a* to *z*.
- Must contain one uppercase letter from *A* to *Z*.
- Must contain one of the following special characters: *@#$*!^*

Roles and Permissions

To perform specific operations on the SmartHub INFER IoT Center console, you must have the required roles and permissions.

Roles

SmartHub INFER IoT Center provides the following default roles:

Identity and Access Administrator
Users with the **Identity and Access Administrator** role can add or modify an organization, add or modify users, groups, roles, and notifications, and view audit logs.

Campaign Administrator
Users with the **Campaign Administrator** role can add or modify campaigns, packages for OTA updates, and view notification definitions and notification destinations.

Package Administrator
Users with the **Package Administrator** role can add or modify packages.

Device Administrator
Users with the **Device Administrator** role can add or modify devices and device templates.

Gateway Administrator
Users with the **Gateway Administrator** role can add devices, create device tokens and credentials, and view device templates.

Alert Administrator Users with the **Alert Administrator** role can acknowledge alerts and view alerts, organizations, users, roles, groups, devices, device templates, notifications, and so on. The **Alert Administrator** role is a view only role.

Monitoring Administrator Users with the **Monitoring Administrator** role can modify alerts and notifications, and view metrics.

Table 1-2. Roles and Permissions

Group	Role	Permissions
Organization Administrators	**Identity and Access Administrator**	View Organization
		Edit Organization
		Create User
		View User
		Edit User
		Delete User
		Create Role
		View Role
		Edit Role
		Delete Role
		Create Group
		View Group
		Edit Group
		Delete Group
		Create Notification Destination
		View Notification Destination
		Edit Notification Destination
		Delete Notification Destination
		Create Notification Definition
		View Notification Definition
		Edit Notification Definition
		Delete Notification Definition
		View Audit Logs
		Create Membership
		View Membership
		Edit Membership
		Delete Membership
		Create Organization Settings

Table 1-2. Roles and Permissions (continued)

Group	Role	Permissions
		View Organization Settings
		Edit Organization Settings
		Delete Organization Settings
System Administrators	**Campaign Administrator**	**View Organization**
		Create Campaign
		View Campaign
		Edit Campaign
		Delete Campaign
		Abort Campaign
		Approve Campaign
		Start Campaign
		Stop Campaign
		Schedule Campaign
		Create Package
		View Package
		Edit Package
		Delete Package
		Create Raw File
		View Raw File
		Delete Raw File
		View Device
		View Device Template
		View Notification Definition
		View Notification Destination
		View Notification Instance
		Create Filter Definition
		View Filter Definition
		Edit Filter Definition
		Delete Filter Definition
		View Metrics
		View System Notification Instance
	Package Administrator	Create Package
		View Package

Table 1-2. Roles and Permissions (continued)

Group	Role	Permissions
		Edit Package
		Delete Package
		View Organization
	Device Administrator	View Device
		Create Device
		Edit Device
		Delete Device
		Edit Custom Properties
		Create Device Template
		Edit Device Template
		View Device Template
		Delete Device Template
		Create Device Credential
		Create Device Token
		Create Filter Definition
		View Filter Definition
		Edit Filter Definition
		Delete Filter Definition
		View Metrics
		View Audit Logs
		Get Device Command
		Send Device Command
		Edit Device Command
		Delete Device Command
		View Organization
		Update Agent
	Gateway Administrator	Create Device Credential
		Create Device Token
		View Device
		Create Device
		View Device Template
		View Organization
	Alert Administrator	Edit Alert

Table 1-2. Roles and Permissions (continued)

Group	Role	Permissions
		View Organization
		View User
		View Role
		View Group
		View Notification Destination
		View Notification Definition
		View Notification Instance
		View Device
		View Device Template
		View Alert Definition
		View Alert
		View Campaign
		View Package
		View Metrics
		View Audit Logs
		View Filter Definition
	Monitoring Administrator	View Organization
		View Device
		View Device Template
		View Notification Destination
		Create Notification Definition
		View Notification Definition
		Edit Notification Definition
		Delete Notification Instance
		View Notification Instance
		Create Alert Definition
		View Alert Definition
		Edit Alert Definition
		Delete Alert Definition
		Create Alert
		View Alert
		Edit Alert
		Delete Alert
		View Metrics

Table 1-2. Roles and Permissions (continued)

Group	Role	Permissions
		Create Filter Definition
		View Filter Definition
		Edit Filter Definition
		Delete Filter Definition

Terminology

Some of the terminologies that are frequently used in this guide are described in this section.

Gateway

A Gateway is a physical device that serves as a connection point between the cloud (public or on premises) and controllers, sensors, and intelligent devices. All data moving to and from the cloud goes through the Gateway. The SmartHub INFER IoTC Agent runs and collects information on behalf of other connected Thing devices through the Gateway.

Connected Device or Thing Device

A connected device or a Thing device is a nonstandard computing device that can transmit data and is connected to a Gateway. The Thing device connects to a Gateway and sends information to the server through the SmartHub INFER SDK Client that is running on the Gateway.

Registered Device

A registered device is a virtual Gateway that is created on the server. A registered device does not have a physical Gateway associated with it.

Enrolled Device

A registered Gateway is enrolled when a physical Gateway is associated with it.

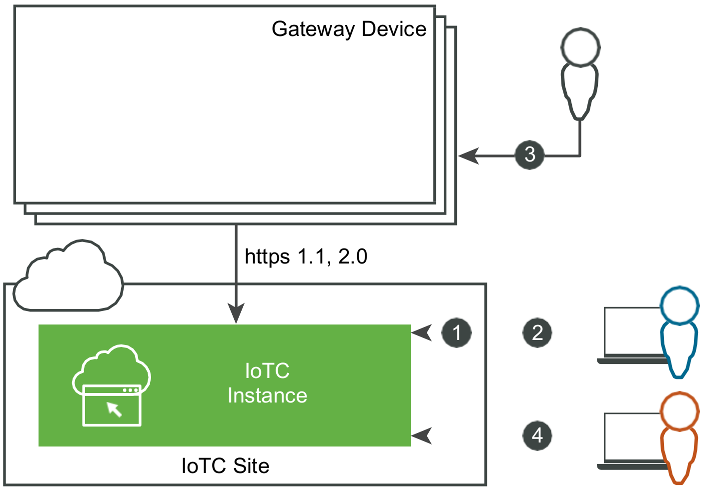
SmartHub INFER IoTC Agent

The SmartHub INFER IoTC Agent is a component that resides in the Gateway. It connects the SmartHub INFER IoT Center services to run commands and to send operational metrics to the IoTC services. The SmartHub INFER IoTC Agent offers an SDK that exposes APIs. Third-party applications can use these APIs on the Gateway to interact with SmartHub INFER IoT Center.

Typical Use Cases

You can perform the following role-specific operations with SmartHub INFER IoT Center.

Setting Up SmartHub INFER IoT Center



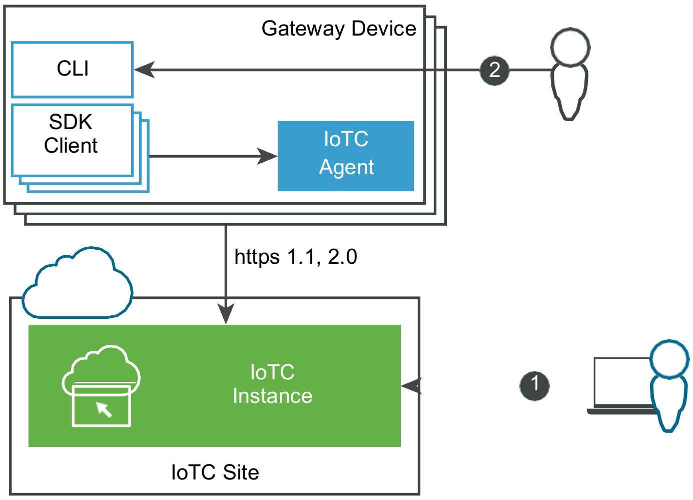
1 Organization Administrator creates organizations.

2 Organization Administrator creates users and assigns roles.

3 Gateway Administrator installs and powers on the Gateway.

4 Device Administrator creates device templates.

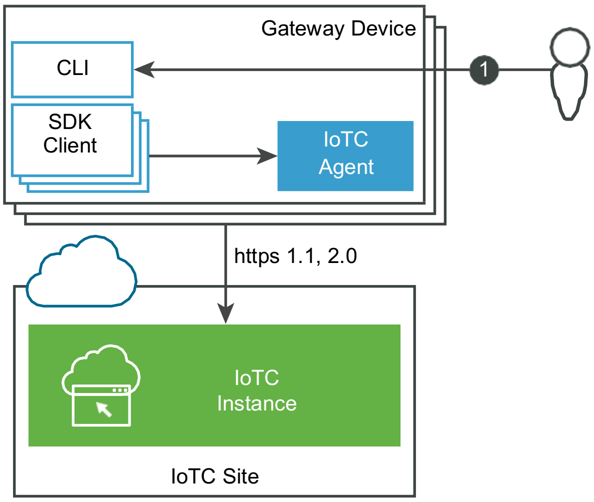
Enrolling a Registered Device



1 Device Administrator registers a device through the SmartHub INFER IoT Center console or through the API.

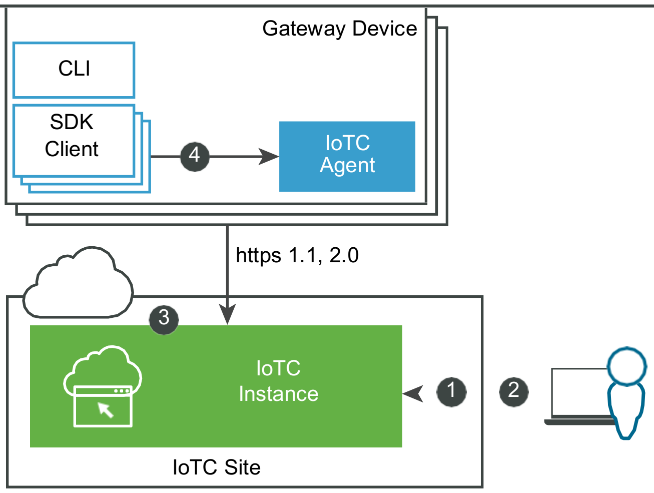
2 Gateway Administrator installs the SmartHub INFER IoTC Agent on the Gateway.

Enrolling a Non-Registered Device



1 Gateway Administrator enrolls the Gateway using the template name, user name, and credentials.

Controlling a Device



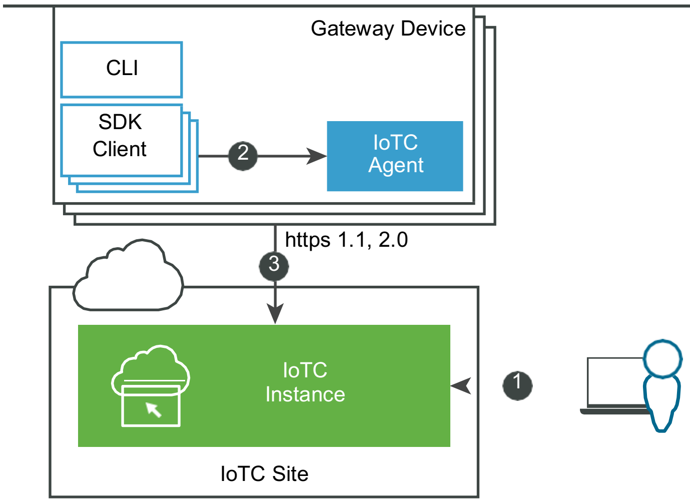
1 Device Administrator configures the device template with the allowed commands.

2 Device Administrator selects a device and sends a command.

3 The server pushes the command to the SmartHub INFER IoTC Agent. If the agent is not connected to the server, the server queues the command.

4 The SmartHub INFER IoTC Agent receives and runs the command, or delegates the command to the SDK Client.

Working with Metrics

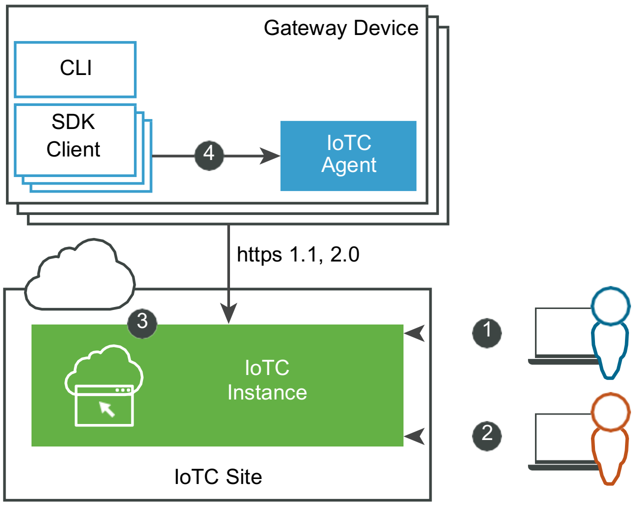


1 Device Administrator configures the metrics to be collected in the device template.

2 The SDK Client collects and publishes the metrics.

3 The SmartHub INFER IoTC Agent transfers the metrics to the IoTC instance.

Configuring Over The Air Updates



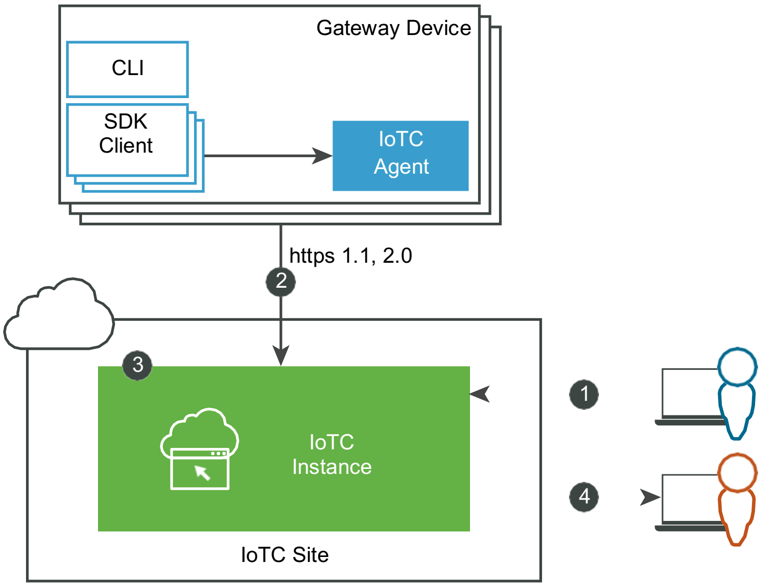
1 Package Administrator creates and uploads update packages.

2 Campaign Administrator creates and starts a campaign to update selected devices with the packages.

3 The server evaluates the active campaigns and queues the update for the device.

4 The SmartHub INFER IoTC Agent downloads, installs, and activates the package. Or, it delegates the command to the SDK Client.

Working with Alerts and Notifications



1 Monitoring Administrator configures the alert definitions and notification definitions.

2 The SmartHub INFER IoTC Agent publishes metrics from the enrolled Gateway.

3 The server evaluates the alert definitions, raises, and sends alert notifications.

4 Alert Administrator acknowledges the alert and initiates action.

Setting Up Your Organization 2

The **Identity and Access** tab allows you to configure Organizations, Users, Groups, and Roles.

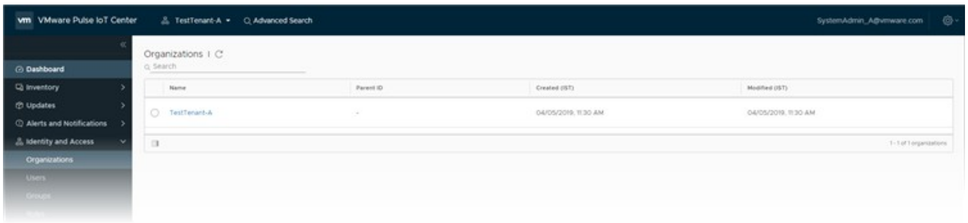
Users authenticate at the organization level, supplying credentials established by an organization administrator when the user was created or imported. System administrators create and provision organizations, while organization administrators manage organization users, groups, and devices.

This chapter includes the following topics:

- Organizations
- Users
- Groups
- Roles

Organizations

Use the **Organizations** tab to manage your organization's users and devices.



	Name	Parent ID	Created (IST)	Modified (IST)
○	TestTenant-A	-	04/05/2019, 11:30 AM	04/05/2019, 11:30 AM

1 - 1 of 1 organizations

Viewing Organization Details

This section lists the steps to view an organization and its details.

Prerequisites

You must have the View Organization permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Organizations.**

The Organizations page displays your organizations, sub organizations, and their status.

2 To view more details about an organization, click the organization name.

Results

Details about the organization are displayed.

Table 2-1. Organization Details

Basic Information	Displays details such as the name of the organization, its creation date and time, parent organization if any, and modified date and time.
Users	Displays the list of users under the organization with their display name, status, created date and time, and modified date and time.
Devices	Displays the device information such as device type, enrollment status, and the date of creation and modification.
Usage	Displays the usage meter of the various states of alerts, campaigns, devices, organizations, and notifications.

Creating an Organization

This section lists the steps to create an organization from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the permissions associated with the Organization Administrator role to perform this operation. For more information about roles and permissions, see Roles and Permissions.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Identity and Access >**

Organizations. The Organizations page is displayed.

2 Click **CREATE**.

The **Create Organization** wizard is displayed.

3 In the **Basic Information** step, perform the following steps:

a Enter the name of your organization.

b Select the parent organization from the **Parent Organization** drop-down menu.

Note The maximum depth till which a sub-organization can be created for an organization is four.

c Under **Organization Identifier**, enter a unique identifier name to identify your organization with. If you are a part of multiple organizations, use this identifier when signing in so that SmartHub INFER IoT Center associates you with the correct organization.

Note **Organization Identifier** is not a mandatory field.

d Click **Next**.

4 In the **Review** step, review the information that you have entered and click **SAVE**.

Results

You have successfully created an organization.

Editing an Organization

This section lists the steps to edit an organization from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the permissions associated with the Organization Administrator role to perform this operation. For more information about roles and permissions, see Roles and Permissions.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Identity and Access >**

Organizations. The Organizations page is displayed.

2 Click the organization to edit.

The details of the organization are displayed.

3 From the **Actions** drop-down menu, click **Edit**.

The Edit Organization window is displayed.

4 Update your organization details and click **SAVE**.

Results

You have successfully updated your organization details.

View Usage

You can measure the usage of services such as notifications, users, file records, devices, alerts, commands, metrics, alert definitions, campaigns, that are running in your organization. The values are displayed for the current organization and its suborganizations.

Prerequisites

You must be an Identity and Access administrator to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center console, go to **Identity and Access** > **Organizations** and select your organization.

2 From the organization details page, click the Usage tab.

3 To view the usage from the time the organization was created, click **ALL HISTORY**.

4 To view the usage for a particular duration, click **CUSTOM** and select the duration.

5 Click **VIEW DETAILS**.

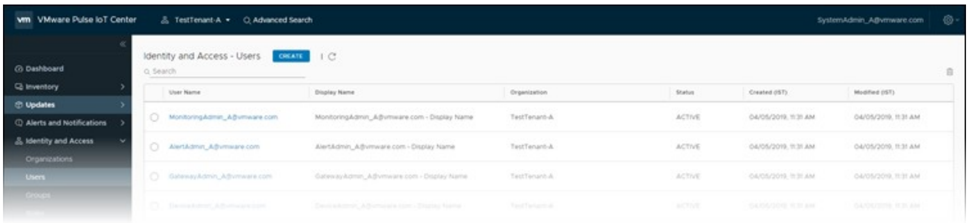
The services and their usage values are displayed.

6 To export the usage values in CSV format, click **EXPORT AS CSV**.

Users

SmartHub INFER IoT Center determines the level of access for the user based on the permissions that you assign to the user.

The permissions defined for these users apply whenever a user connects to SmartHub INFER IoT Center.



Identity and Access - Users

Search

User Name	Display Name	Organization	Status	Created (IST)	Modified (IST)
MonitoringAdmin_A@vmware.com	MonitoringAdmin_A@vmware.com - Display Name	TestTenant-A	ACTIVE	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM
AlertAdmin_A@vmware.com	AlertAdmin_A@vmware.com - Display Name	TestTenant-A	ACTIVE	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM
GatewayAdmin_A@vmware.com	GatewayAdmin_A@vmware.com - Display Name	TestTenant-A	ACTIVE	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM
DeviceAdmin_A@vmware.com	DeviceAdmin_A@vmware.com - Display Name	TestTenant-A	ACTIVE	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM

Creating a User

This section lists the steps to create a user from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the CREATE_USER permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Identity and Access** >

Users. The Identity and Access - Users page is displayed.

2 Click **CREATE**.

The **Create User** wizard is displayed.

3 In the **Details** step, enter the following details:

- **Display Name** - Enter the display name of the user.
- **Username** - Enter the user name to use for logging in to SmartHub INFER IoT Center.
- **Email Address** - Enter a valid email ID.
- **New Password** - Enter a password for the user. For information about password requirements, see the Password Requirements section.
- **Confirm New Password** - Confirm the password that you have entered.

Click **Next**.

4 In the **Groups** step, select the appropriate groups for the user and click **Next**.

5 In the **Review** step, review the information and click **SAVE**.

Results

You have successfully created a user.

Editing a User

This section lists the steps to edit user details from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the EDIT_USER permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Identity and Access** >

   Users. The Identity and Access - Users page is displayed.

2. Click the user to edit.

   The user details are displayed.

3. From the **Actions** drop-down menu, click **Edit**.

   The **Edit User** wizard is displayed. Here, you can update the display name of the user, change the user status, add or delete groups to the user, and add or delete roles.

4. Update your user details and click **SAVE**.

Results

You have successfully edited the user details.

Deleting a User

This section lists the steps to delete a user from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the DELETE_USER permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Identity and Access** >

   Users. The Identity and Access - Users page is displayed.

2. Select the user to delete.

3. Click the delete icon on the top-right side of the screen. You can also select **Delete** from the **Actions** drop-down menu.

   An action confirmation message is displayed.

4. To confirm the action, click **DELETE**.

   Note This action deletes the user permanently.

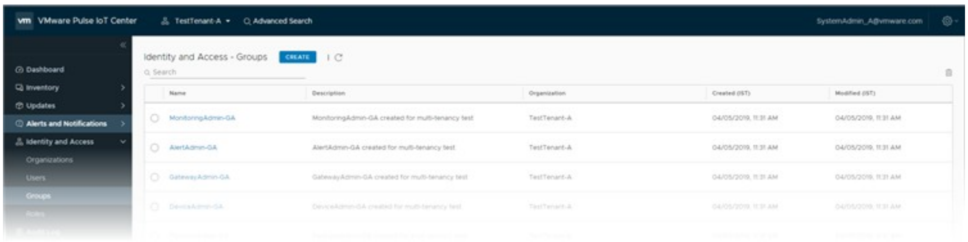
Results

You have successfully deleted a user.

Groups

You can manage a set of users with similar permissions through groups.

A user can be a member of more than one group. When you assign permissions to a group, all users in the group inherit those permissions. Using groups can reduce the time it takes to set up your permissions model.



	Name	Description	Organization	Created (IST)	Modified (IST)
○	MonitoringAdmin-GA	MonitoringAdmin-GA created for multi-tenancy test	TestTenant-A	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM
○	AlertAdmin-GA	AlertAdmin-GA created for multi-tenancy test	TestTenant-A	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM
○	GatewayAdmin-GA	GatewayAdmin-GA created for multi-tenancy test	TestTenant-A	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM
○	DeviceAdmin-GA	DeviceAdmin-GA created for multi-tenancy test	TestTenant-A	04/05/2019, 11:31 AM	04/05/2019, 11:31 AM

Creating a Group

This section lists the steps to create a user group from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the CREATE_GROUP permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Identity and Access** >

Groups. The Identity and Access - Groups page is displayed.

2 Click **CREATE**.

The **Create Group** wizard is displayed.

3 In the **Details** step, enter the group name and a short description about the group. Click **NEXT**.

4 In the **Roles** step, select a role for the group. Some of the default roles are Identity and Access Administrator, Campaign Administrator, Package Administrator, and Device Administrator. Click **Next**.

5 In the **Review** step, review the information and click **CREATE**.

Results

You have successfully created a user group.

Editing a Group

This section lists the steps to edit user details from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the EDIT_GROUP permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Identity and Access** > **Groups**. The Identity and Access - Groups page is displayed.
2. Click the group to edit.

   The group details are displayed.
3. From the **Actions** drop-down menu, click **Edit**.

   The **Edit Group** wizard is displayed.
4. Update the group details and click **SAVE**.

Results

You have successfully updated the group.

Deleting a Group

This section lists the steps to delete a group from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the DELETE_GROUP permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Identity and Access** > **Groups**. The Identity and Access - Groups page is displayed.
2. Select the radio button against the group to delete.
3. Click the delete icon on the top-right side of the screen. Or, select **Actions** > **Delete**.

   An action confirmation message is displayed.
4. To confirm the action, click **DELETE**.

   Note This action deletes the group permanently.

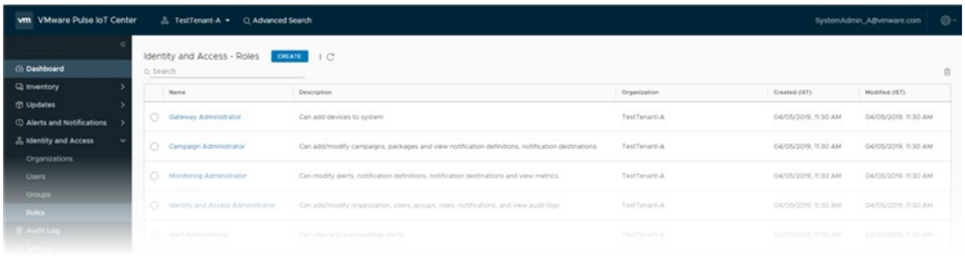
Results

You have successfully deleted the group.

Roles

When you assign a user or group permissions, you pair the user or group with a role. A role is a predefined set of privileges.

A single user might have different roles. For a list of default roles and their permissions, see Roles and Permissions.



	Name	Description	Organization	Created (IST)	Modified (IST)
○	Gateway Administrator	Can add devices to system	TestTenant-A	04/05/2019, 11:30 AM	04/05/2019, 11:30 AM
○	Campaign Administrator	Can add/modify campaigns, packages and view notification definitions, notification destinations	TestTenant-A	04/05/2019, 11:30 AM	04/05/2019, 11:30 AM
○	Monitoring Administrator	Can modify alerts, notification definitions, notification destinations and view metrics	TestTenant-A	04/05/2019, 11:30 AM	04/05/2019, 11:30 AM
○	Identity and Access Administrator	Can add/modify organization, users, groups, roles, notifications, and view audit logs	TestTenant-A	04/05/2019, 11:30 AM	04/05/2019, 11:30 AM
○	Alert Administrator	Can view and acknowledge alerts	TestTenant-A	04/05/2019, 11:30 AM	04/05/2019, 11:30 AM

Creating a Role

This section lists the steps to create a role from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the CREATE_ROLE permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Identity and Access** >

Roles. The Identity and Access - Roles page is displayed.

2 Click **CREATE**.

The **Create Role** wizard is displayed.

3 In the **Details** step, enter the role name and a short description about the role. Click **Next**.

4 In the **Permissions** step, select the permissions to assign to the role. Click **Next.**

5 In the **Review** step, review the information and click **SAVE**.

Results

You have successfully created a role.

Editing a Role

This section lists the steps to edit role details from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the EDIT_ROLE permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Identity and Access** >

Roles. The Identity and Access - Roles page is displayed.

2 Click the role to edit.

The role details are displayed.

3 From the **Actions** drop-down menu, click **Edit**.

The **Edit Roles** wizard is displayed.

4 Update the role details and click **SAVE**.

Results

You have successfully updated a role.

Deleting a Role

This section lists the steps to delete a role from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the DELETE_ROLE permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Identity and Access** >

Roles. The Identity and Access - Roles page is displayed.

2 Select the role to delete and click the delete icon on the top-right side of the screen. Or, select **Actions** > **Delete**.

An action confirmation message is displayed.

3 To confirm the action, click **DELETE**.

Note This action deletes the role permanently.

Results

You have successfully deleted a role.

Managing Dashboards and Widgets

3

You can create customized dashboards with only those widgets that you want to view.

By default, SmartHub INFER IoT Center provides a dashboard that displays the important information about your devices, alerts, and campaigns. But you can create your own dashboard and add only those widgets that you want to view.

Working with Widgets

Widgets are small applications on the home screen of the SmartHub INFER IoT Console that give you quick access to important information about your devices, alerts, and campaigns.

This chapter includes the following topics:

- Create a Dashboard
- Create a Widget

Create a Dashboard

The first step in customizing your dashboard view is to create a dashboard.

Prerequisites

You must have the **Create Dashboard** permission to perform this operation.

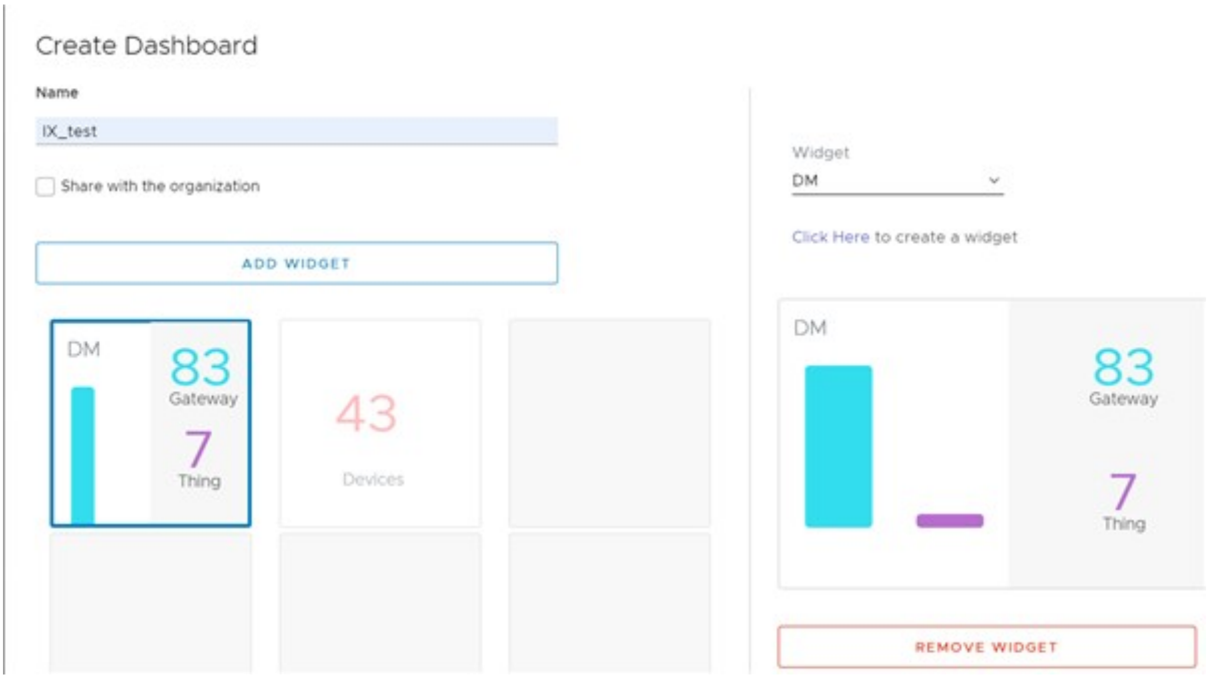
Procedure

1 From the SmartHub INFER IoT Center dashboard page, click the **MANAGE** button.

2 Select **CREATE** > **Create Dashboard**.

The Create Dashboard wizard is displayed.

3 Enter a name for your dashboard and select the **Share with the organization** check box if you want to share your dashboard with others who have the **View Dashboard** permission within your organization. If you select **Share with the organization**, you can also allow others to modify your dashboard. Select **Allow others to modify**.



Create Dashboard

Name: IX_test

□ Share with the organization

Widget: DM

Click Here to create a widget

4 To start adding the widgets that you created to your dashboard, click **ADD WIDGET**. You can select your **widget** from the **Widget** drop-down menu on the right.

5 To remove a widget from your dashboard, select the widget and click **REMOVE WIDGET**.

6 Click **SAVE**.

Your dashboard is saved.

Results

You have successfully created a dashboard and added widgets to it.

Create a Widget

You can create customized widgets to be displayed on your dashboard.

Prerequisites

You must have the **Create Dashboard Widget** permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center dashboard page, click the **MANAGE** button.

2 Select **CREATE** > **Create Widget**.

The Create Widget wizard is displayed.

3 In the **Details** step, enter the following information:

- **Name** - Type the name of your widget.
- To share your widget with others in your organization, select **Share with the organization**. To allow others to modify your dashboard, select **Allow others to modify**.
- **Select Entity** - Select the entity for which you want to create a widget. The entities are:
  - **Devices** - Creates a device widget.
  - **Alerts** - Creates an alerts widget.
  - **Device Template** - Creates a device template widget.
  - **Campaigns** - Creates a campaigns widget.

In this example, we create a **Devices** widget.

4 Select **Devices** and click **NEXT**.

5 In the **Query Group and Chart** step, you can build a query to narrow down the sample set of devices. In **Filter** step:

a Select devices of a particular **Device Type**, **Enrollment State**, **Device Template**, and **Properties** to be displayed.

You can search the properties by name and value.

b To group the devices according to their **Device Type** or **Enrollment State**, select **Group data**.

c Under **Data Visualization**, select a visualization type for your widget.

The count of all devices that meet the query parameters is in yourwidget.

6 Click **NEXT**.

7 In the **Review** step, review the information that you have entered, and click **SAVE**.

Results

You have successfully created a widget.

What to do next

Add the widget to your dashboard.

Onboarding a Gateway to SmartHub INFER IoT Center

4

Onboard your gateway to SmartHub INFER IoT Center using the steps described in this section. To onboard your gateway, you must perform the following steps:

1. Create a device template.
2. Download and install the IoTC Agent.
3. Onboard your gateway using one of the following authentication methods:
   a. Basic Authentication - A simple authentication scheme built into the HTTP protocol. The client sends HTTP requests with the Authorization header that contains the word `Basic` followed by a space and a base64-encoded string `username:password`.
   b. Token-Based Authentication - Creates a single use gateway credential with signature and expire time verification.
   c. Property-Based Authentication - Creates a single use gateway credential with device identity value verification.
   d. TPM-Based Authentication - Creates a single use gateway credential with a Trusted Platform Module (TPM) identity value verification. Using this authentication method, you can whitelist a gateway so that it is allowed for enrollment.
   e. Zero Touch Enrollment - Register gateways in bulk using zero touch enrollment credentials. You must upload a CSV file with the hardware ID and model number of each gateway.

This chapter includes the following topics:

- Register Multiple Devices
- Whitelisting a Device
- Edit a Device Template

Working with Device Templates

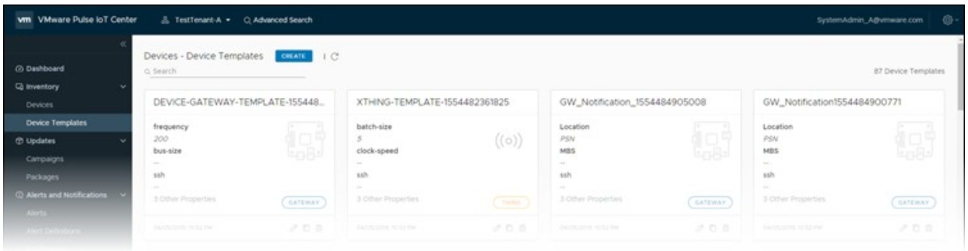
A device template is the blueprint of the device that is to be registered on the SmartHub INFER IoT Center. The device template contains the following properties:

- System Properties: Properties that are collected from the device and sent to the SmartHub INFER IoT Center server. These properties can be collected automatically by the IoTc Agent or can be sent by the User client code from the IoTC Agent. You can query these properties using Advanced Search.
- Custom Properties: Properties that an administrator creates for a device using the SmartHub INFER IoT Center console. These properties can be used for querying information or for sending configuration files to devices.
- Metrics: The metrics that are collected from the devices. You can configure metric keys such as Name, Value Type, and Display Unit. The SmartHub INFER IoTC Agent collects only the set of metrics that the server specifies. The metric value type is defined in the device template and is validated on the IoTC Agent.
- Connected Device Templates: You can configure the type of Thing templates that are allowed to connect to the gateway or to a Thing device.
- Commands: You can configure the list of commands to send to the gateway. The supported commands are:
    - Client Command
    - Custom Command
    - File Upload
    - Reboot
    - SSH

  For more information about these commands, see the Create a Device Template section.

A device created from a template receives the default configuration of the template. The device template helps in creating a simplified process for registering new devices.

A device can have a restricted list of Thing devices with a specified template. You can create a device based on the list of available Thing templates. To connect a Thing device, ensure that the Thing template is a part of the parent gateway template.



To view more details of a device template, click the name of the device template from the Devices - Device Templates page.

Create a Device Template

To enroll a gateway, you must first create a device template.

Prerequisites

You must have the CREATE_DEVICE_TEMPLATE permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Device Templates**. The Inventory - Device Templates screen is displayed.

2 Click **CREATE**.

The Create Device Template wizard is displayed.

3 In the **Details** step, enter the template name and select the device type. To associate an image with the device, click **UPLOAD IMAGE**. Click **Next**.

4 In the **Properties** step, perform the following steps:

a Under **System Properties**, enter the system property for your device. System Properties are the properties that are collected from the device and sent to the SmartHub INFER IoT Center server. The IoTC Agent collects these properties automatically or the client code supplies them. To add more properties, click **+ Add**.

Note The default system properties are pre-populated and you cannot delete them.

b Under **Custom Properties**, enter the custom property and value for your device. Custom Properties are the default properties and values that are associated with all the devices of this template. These properties are used for querying information or for sending configuration files to the devices. These properties are not collected from the device and can be edited on the server. To add more properties and values, click **+ Add**.

c Click **Next**.

5 In the **Metrics** step, select the metrics that you want to collect. The **CPU-Usage**, **Memory-Usage**, and **Disk-Usage** metrics are added by default if you select **Gateway** as your device. To add more metrics, click **+ Add** and enter the following information:

- **Display Name** - Display name of the metric.
- **Value Type** - Boolean, Integer, Double, or String.
- **Unit** - Enter the unit of measurement. Ensure that the unit is the same as the one collected from the device.

Note

- The Metric **Name** and **Value Type** collected from the device must match the **Allowed Metrics** configuration. Else, the configuration is rejected.
- The default metrics are pre-populated. You can delete them or add new metrics as required.

If you have selected **Gateway** as your device, perform the next steps. If you have selected **Thing** as your device, go to the last step.

6 In the **Connected Device Templates** step, add the thing device template that can be connected to this template. Click **Add Template**, select the connected device template, and click **Next**.

7 In the **Commands** step, click **Add Command**.

a Select the following **Type** of commands that you want to run on your device:

- **Client Command** - Set a command to communicate with the connected devices in your gateway. For example, you can set a command to turn on the LED that is connected to the gateway when an alert is raised. As a system administrator, you can set a list of allowed client identifiers to be used by the client application when initializing a session with the SmartHub INFER IoTC Agent. Ensure that the client identifier you enter matches the client identifier value in the operating system. You can add a client command for both gateway and thing devices. For a sample use case of using client commands, see the Container Management Sample Use Case section.

- **Custom Command** - Set a custom command. For example, set a command to configure the IP address of the device or enable DHCP.

  Note

  - You must enter the full path of the command. For example, enter `/usr/bin/cp` instead of `cp`.
  - You can add multiple comma-separated arguments for a command.

- **File Upload** - Set a command to upload log files to the agent. The **File Upload** command takes multiple file paths in one argument. The IoTC Agent archives the files and uploads them to the SmartHub INFER IoT Center server as a `.zip` file. Administrators can download the `.zip` file and extract its content.

- **Reboot** - Set a command to reboot the device.

- **SSH** - Set a command to enable or disable SSH on the device.

- **SSH**, **Reboot**, and **File Upload** commands run with `root user` privileges.

- **Custom Command** can run with `root user` or `iotc-user` privileges.

Note

- If you select the **Run with root privilege** option, the command runs with `root user` privilege on the gateway.
- If you do not select the **Run with root privilege** option, the command runs with `iotc user` privilege on the gateway.

For an example of using custom commands to redirect an output file, see Redirecting an Output Using Custom Commands. To run custom commands, enter the command name and command information under **Custom Command**. To add more commands, click **Add Command** and click **Next**

b In the **Name** text box, enter the name of the command.

c Enter the **Client Identifier.**

d In the **Arguments** step, enter the **Name** and **Value** of the argument and click **DONE**.

Note

- Custom and Client Commands can have the placeholders as part of the command argument values.
- The **Value** of the argument must be entered in the format `${<placeholder>}`, where placeholder must be one of the system property name. The SmartHub INFER IoTC agent parses and replaces the placeholder with the corresponding system property value.

8 In the **Enrollment Provider** step, perform the following steps:

a Select the **Provider Type** from the drop-down menu.

- **Token Based** - Creates a single-use device credential with signature and expire-time verification.
- **Property Based** - Creates a single-use device credential with a device identity value verification.
- **TPM Based** - Creates a single use device credential with a Trusted Platform Module (TPM) identity value verification.

b Under **Provider Config**:

- If you have selected **Token Based** as the provider type, enter the expiry time of the device credential and select the interval from the drop-down menu.
- If you have selected **Property Based** as the provider type, enter the identity key. The device presents the value of this identity key during onboarding.
- If you have selected **TPM Based** as the provider type, select **Requires Whitelisting** if you want to whitelist the gateways for an automatic enrollment.

Note Basic authentication is enabled by default.

9 Click **NEXT**.

10 In the **Settings** step, configure the IoTC Agent settings for your devices. These settings are applied when you onboard new devices.

- **Agent log level** - Select the logging level for collecting the IoTC Agent logs from a device.
- **Maximum number of clients** - Enter the maximum number of SDK clients that can communicate through the IoTC Agent with the SmartHub INFER IoT Center server.
- **Command fetch interval (seconds)** - Enter the time interval for the SmartHub INFER Agent to fetch commands from the SmartHub INFER IoT Center server.

Note The maximum value for the command fetch interval is 43200 seconds. The minimum value is 10 seconds and the default value for the command fetch interval is 30 seconds. If the time interval is not within the specified range, an error message is displayed.

- **Metrics interval (seconds)** - Enter the time interval between 60 and 43200 seconds for transmitting metrics from the IoTC Agent to the SmartHub INFER IoT Center server.

  Note The minimum value for the metrics interval is 60 seconds and the default value for the metric interval is 300 seconds. If the time interval is not within the specified range, an error message is displayed.

- **Server Request Timeout (seconds)** - Enter the timeout value between 60 and 3600 seconds for requests from the IoTC Agent to the SmartHub INFER IoT Center server.

  Note The minimum value for the timeout is 60 seconds and the default value for the timeout is 300 seconds. If the time interval is not within the specified range, an error message is displayed.

- **Network Bandwidth** - Enter the maximum network bandwidth allowed on the device for the IoTC Agent. The data rate is in Bytes per second (B/s). The value 0 denotes unlimited network bandwidth.

  Note The minimum value required for network bandwidth is 2500 bytes/second and the default value is 0.

- In the **Forward Proxy** step, click **Add** to enter the HTTP Proxy details.
  - **Server IP** - Enter the IP address
  - **Port** - Enter the Port Number
  - **User Name** - Enter your user name
  - **Password** - Enter the password
- Click **DONE**.

Note You can add multiple HTTP Proxy settings. These proxy settings details are stored in the SmartHub INFER IoTC Agent configuration file. The SmartHub INFER IoTC Agent uses these proxy settings to connect to the server for enrolling a gateway. The proxy settings details are then sent to the gateway while enrolling a device or every time the template is modified. The SmartHub INFER IoTC agent selects the first working proxy server from the list of proxy servers and updates that in the SmartHub INFER IoTC Agent configuration file.

If the current proxy server stops working, it fetches the next working proxy server. For enrollment, if the SmartHub INFER IoTC Agent cannot connect to the SmartHub INFER IoT Server without connecting to the proxy server, then you must manually enter the working proxy server details in the SmartHub INFER IoTC agent configuration file.

11 Click **NEXT**.

12 In the **Review** step, verify that the information that you have entered is correct. Click **SAVE**.

Results

You have successfully created a device template. You can view your device template in the Devices - Device Templates page.

What to do next

Download and install the SmartHub INFER IoTC Agent. For more information, see Install the SmartHub INFER IoTC Agent on a Linux Gateway.

Redirecting an Output Using Custom Commands

This section provides an example to redirect outputs using the SmartHub INFER IoT Center Custom Command feature.

You can run a script or a binary on a gateway and redirect its output to a file. You can then retrieve the output file using the Upload command. In this example, we run a ping on a gateway to detect its connectivity to a certain endpoint. To perform this operation, you must wrap the command into a shell by providing the `/bin/sh` path as the executable, and pass the actual binary and arguments to the shell. You must then pass the `-c` argument to interpret the rest of the arguments as binary and associate the arguments to the binary. Perform the following steps:

Procedure

1. In the Create Device Template wizard, proceed to the **Commands** step.
2. Select **Custom Command** from the drop-down menu and enter a name for your command.
3. Under **Executable**, enter the path as **`/bin/sh`**.
4. Under **Arguments**, enter the path to the output file. For example, `-c, /bin/ping -c 4 8.8.8.8 > /tmp/ping.txt`.

Results

The output for `ping -c 4 8.8.8.8` is redirected to the `ping.txt` file.

Installing the SmartHub INFER IoTC Agent

You can install the SmartHub INFER IoTC Agent on gateways that run on Windows and Linux operating systems.

Install the SmartHub INFER IoTC Agent on a Linux Gateway

Follow the steps listed in this section to download and install the SmartHub INFER IoTC Agent on your Linux gateway.

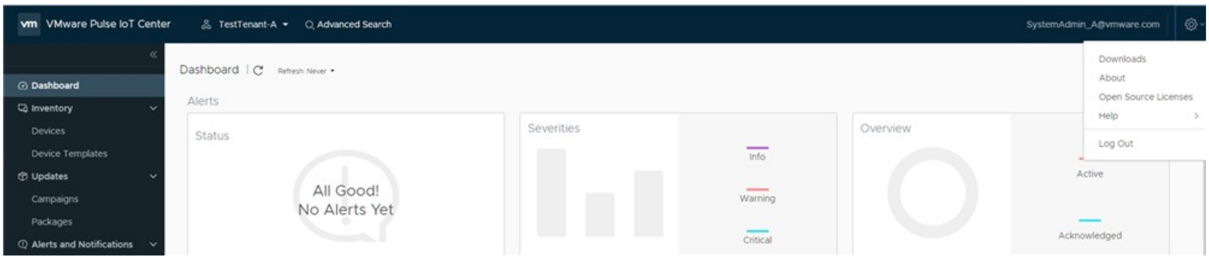
Prerequisites

To change `DefaultClient` from starting by default, run `install.sh` with the `disable-defclient` argument when installing the IoTC Agent.

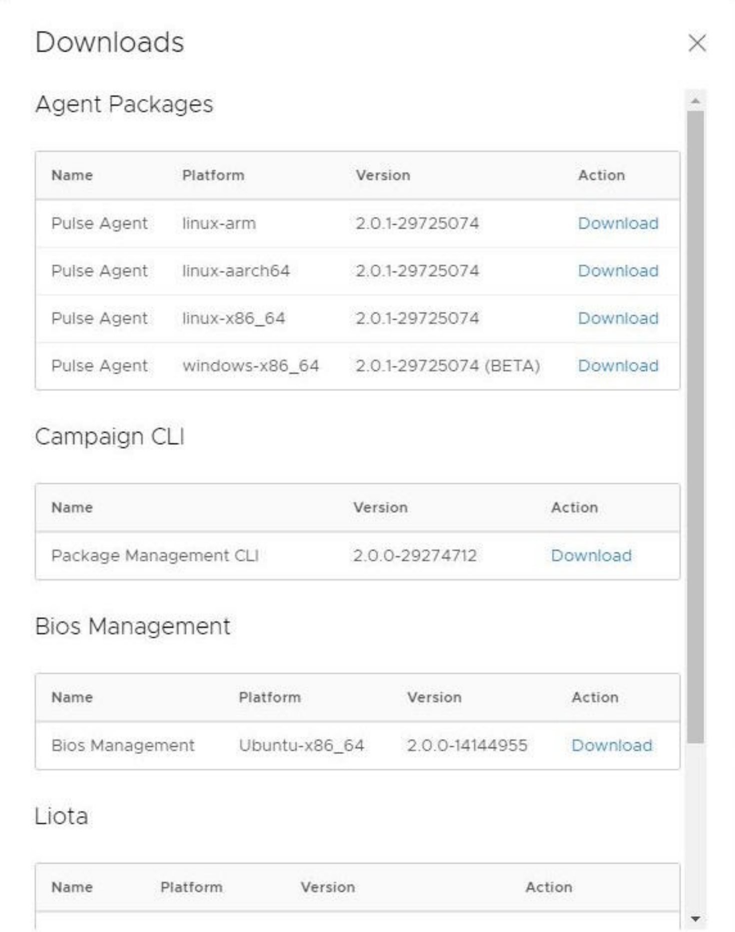
Procedure

1. Log in to the SmartHub INFER IoT Center UI.

2 From the top right of the home page, click the settings icon and click **Downloads**.



3 Download the IoTC Agent tarball **Pulse Agent (x86_64)**, **Pulse Agent (aarch64)**, or **Pulse Agent (arm)** to your local system.



Downloads

Agent Packages

Name	Platform	Version	Action
Pulse Agent	linux-arm	2.0.1-29725074	Download
Pulse Agent	linux-aarch64	2.0.1-29725074	Download
Pulse Agent	linux-x86_64	2.0.1-29725074	Download
Pulse Agent	windows-x86_64	2.0.1-29725074 (BETA)	Download

Campaign CLI

Name	Version	Action
Package Management CLI	2.0.0-29274712	Download

Bios Management

Name	Platform	Version	Action
Bios Management	Ubuntu-x86_64	2.0.0-14144955	Download

Liota

Name	Platform	Version	Action

4 Using an FTP/SFTP application such as WinSCP or FileZilla, copy the SmartHub INFER IoTC Agent tarball to the gateway file system.

Alternatively, you can copy the URL of the IoTC Agent from the SmartHub INFER IoT Center console. Create a folder for SmartHub INFER IoT Center using the `mkdir pulse` command and change the permission to `sudo chmod 777 pulse`. On the command-line interface, use the CURL or WGET commands to download the IoTC Agent. For example:

```
wget -O inferagent.tar.gz https://iotc001-infer.smarthub.ai/api/iotc-agent/<agent-file-name>.tar.gz
```

OR

```
curl -o inferagent.tar.gz https://iotc001-infer.smarthub.ai/api/iotc-agent/<agent-file-name>.tar.gz
```

Note To specify the output file name, WGET uses the upper case 'O' while CURL uses the lower case 'o'.

5 Extract the IoTC Agent tarball on the gateway. For example, `tar -xvzf inferagent.tar.gz`

6 Change the directory to `iotc-agent` and run `install.sh` as `sudo`.

```
sudo ./install.sh
```

7 Verify that the IoTC Daemon and the IoTC Agent services are running. View the **syslog** or use the `journalctl -f` command.

Note GRPC errors are common and expected at this stage.

Results

You have successfully installed the SmartHub INFER IoTC Agent.

Install the SmartHub INFER IoTC Agent on a Windows Gateway

Follow the steps listed in this section to download and install the SmartHub INFER IoTC Agent on a Windows gateway.

Prerequisites

Note This feature is available as a preview.

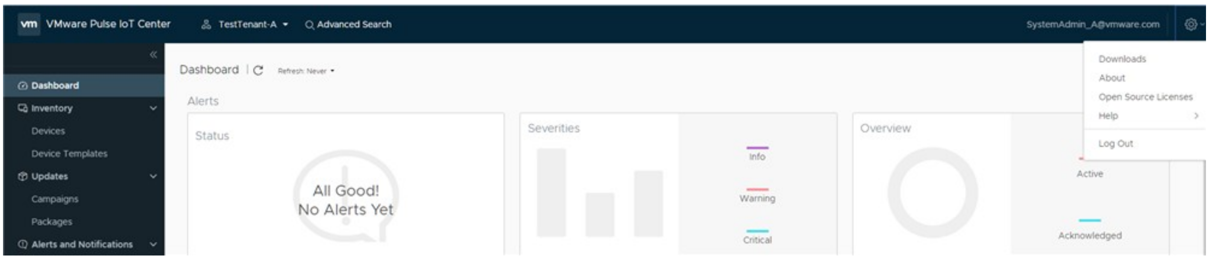
- Supported operating systems:
  - Windows 10 IoT Enterprise (x64)
  - Windows 10 IoT Core (x64, ARM)
  - Windows Embedded Standard 7 (x64)

    Note Windows Embedded Standard 7 (x64) requires Windows PowerShell v2.0 or later to be installed on your system.
- Available SmartHub INFER IoTC Agent binaries:
  - Windows 10 x64 (any edition): `iotc-agent-windows-x86_64-<version>.tar.gz`
  - Windows 10 IoT Core ARM: `iotc-agent-windows-arm-<version>.tar.gz`

Procedure

1 Log in to the SmartHub INFER IoT Center UI.

2 From the top right of the home page, click the settings icon and click **Downloads**.



3 From the Downloads page, click **Download** against **Pulse Agent (windows-x86_64)** and save the iotc-agent-windows-x86_64-<version>.tar.gz file to your local system.

4 Copy the tarball to the gateway file system.

5 Extract the archive on the gateway using the tar command.

The tar command is available on Windows 10 version 1803 and later.

Note If your gateway does not have the tar command, recreate the archive from .tar.gz to .zip.

6 Run the PowerShell script install.ps1 with administrator privileges.

```
Command prompt:
> powershell.exe -ExecutionPolicy RemoteSigned -File install.ps1

PowerShell prompt:
> Set-ExecutionPolicy RemoteSigned
> & install.ps1
```

Results

The SmartHub INFER IoTC Agent for Windows is installed at C:\Program Files\VMware\iotc-agent. The following services are installed:

- iotc-agent
- iotc-defclient (optional)

What to do next

You can view the event logs using one of the following methods:

1 Event Viewer - Create a custom view filter in Event Viewer with the event source as **iotc-agentd**.

2 PowerShell Script - To view the last 10 event logs, run the following PowerShell script:

```
Get-EventLog -LogName Application -Source iotc-agentd -Newest 10 | Select-Object -Property
TimeGenerated,Message | Format-Table -HideTableHeaders –Wrap
```

Onboard a Gateway Using Basic Authentication

This section lists the steps to onboard a gateway using the basic authentication method.

Prerequisites

- Ensure that you have the user name and password of the user in the organizaion where you want to enroll the device.
- You must have created a device template with **Basic Enrollment** as the **Provider Type**, and it must be available on the SmartHub INFER IoT Center console.
- You must have installed the IoTC Agent on your gateway.
- You must know the name that you want to assign to your gateway in the SmartHub INFER IoT Center console.

Procedure

1 From the SmartHub INFER IoT Center UI, go the **Inventory** > **Device Templates**.

2 Identify the device template to be associated with your gateway.

3 Log in to your gateway and change the directory to `/opt/vmware/iotc-agent/bin`.

4 Run the following command:

```
--auth-type=BASIC [--org-domain-name=<organization domain name>]
        --template=<template name> --name=<gateway name>
        --username=<user name> [ --password=<prompt|file:<path>> ]
```

Note If you are providing a file as input for the password, ensure that the file is accessible by the `DefaultClient`.

For a successful enrollment, the response must be `0`.

Results

You have successfully enrolled a gateway using the basic authentication method and have assigned a device ID to it. To verify that the gateway is enrolled, go to the SmartHub INFER IoT Center UI and click the **Devices** tab. The gateway must be listed in the Devices - All Devices page and its status must be **ENROLLED**.

Onboard a Gateway Using Token-Based Authentication

This section lists the steps to onboard your gateway using the token-based authentication method.

Prerequisites

- You must have the CREATE DEVICE permission to perform this operation.
- You must have created a device template with **Token Based** as the **Provider Type**, and it must be available on the SmartHub INFER IoT Center console.
- You must have installed the IoTC Agent on your gateway.
- You must know the name that you want to assign to your gateway in the SmartHub INFER IoT Center console.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** >

   Devices. The Devices - All Devices page is displayed.

2. Click **REGISTER**.

   The Register Gateway wizard is displayed.

3. Enter your gateway name and select the device template that has Token-Based Authentication enabled to associate with your gateway.

4. Click **REGISTER**.

   Your gateway is registered and is listed in the Devices - All Devices page.

5. Create a credential to enroll your gateway. From the Devices -All Devices page, click the gateway that you registered.

6. Click the **Actions** drop-down menu and select **Create Gateway Credentials**. Click **CREATE**.

7. Copy the token to the clipboard.

   The token expiry time that you set when creating the template is displayed.

8. Log in to your gateway and change the directory to `/opt/vmware/iotc-agent/bin`.

9. Run the following command:

   ```
   ./iotc-agent-cli enroll --auth-type=REGISTERED --token=<authenticationtoken>
   ```

Results

For a successful enrollment, the response must be `0`.

Onboard a Gateway Using Property-Based Authentication

This section lists the steps to onboard your gateway using the property-based authentication method.

Prerequisites

- You must have the CREATE DEVICE permission to perform this operation.
- You must have installed the IoTC Agent on your gateway.
- You must have created a device template with **Property Based** as the **Provider Type**, and it must be available on the SmartHub INFER IoT Center console.
- You must know the name that you want to assign to your gateway in the SmartHub INFER IoT Center console.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** >

   Devices. The Devices - All Devices page is displayed.

2 Click **REGISTER**.

The Register Gateway wizard is displayed.

3 Enter your gateway name and select the device template that has Property-Based Authentication enabled to associate with your gateway.

4 Click **REGISTER**.

Your gateway is registered and is listed in the Devices - All Devices page.

5 Create a credential to enroll your gateway. From the Devices -All Devices page, click the gateway that you registered.

6 Click the **Actions** drop-down menu and select **Create Gateway Credentials**. Click **CREATE**.

7 Enter a value for the keys that you defined when you create the device template. The key and value pair must be unique for all the devices that you have configured under your Organization. Thedevice must send the same key and value pair to the server.

8 Log in to your gateway and change the directory to `/opt/vmware/iotc-agent/bin`.

9 Run the following command:

```
./iotc-agent-cli enroll --auth-type=PROPERTY --key=<identitykey> --value=<correspondingvalue>
```

Results

For a successful enrollment, the response must be `0`.

Onboard a Gateway Using TPM-Based Authentication

This section lists the steps to onboard a gateway using the Trusted Platform Module based authentication method.

Prerequisites

- You must enable TPM from your gateway's BIOS settings.
- You must have installed the IoTC Agent on your gateway.
- To verify that TPM permissions and settings are in place, run the following command on your gateway:

```
[root@localhost ~]# su iotc -c /opt/vmware/iotc-agent/bin/tpm_verify
```

The following output must be displayed:

```
VMware secured.
```

- The following steps are mandatory for gateways running on Ubuntu operating systems:
    - If VMware secured is not displayed, run the following commands on the gateway and rerun the tpm_verify command:

      ```
      sudo groupadd --system tss
      #This command creates a system level 'tss' group.
      sudo useradd --system tss -g tss
      #This command creates a system level 'tss' user and adds it to the 'tss' group.
      sudo usermod -a -G tss iotc
      #This command adds 'iotc' user to the 'tss' group.
      sudo usermod -g tss iotc
      #This commands makes 'tss' as the primary group of 'iotc'
      ```

    - To run every time your gateway starts, add the following commands in a run script:

      ```
      sudo chown tss:tss /dev/tpmrm0
      #Changes the ownership of /dev/tpmrm0 from 'root:root' to 'tss:tss'.
      sudo chmod g+rw /dev/tpmrm0
      #Adds read+write permissions for group on the device /dev/tpmrm0.
      ```

- You must have the CREATE DEVICE permission to perform this operation.
- You must have created a device template with **TPM Based** as the **Provider Type**, and it must be available on the SmartHub INFER IoT Center console.
- You must know the name that you want to assign to your gateway in the SmartHub INFER IoT Center console.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** >

   Devices. The Devices - All Devices page is displayed.

2. Click **REGISTER**.

   The Register Gateway wizard is displayed.

3. Enter your gateway name and select the device template that has the TPM-Based authentication method enabled to associate with your gateway.
4. Click **REGISTER**.

   Your gateway is registered and is listed in the Devices - All Devices page.

5 To enroll your gateway, create a credential :

a Log in to your gateway and run the fingerprint command to generate the TPM Endorsement Public Key:

```
[root@localhost bin]# ./opt/vmware/iotc-agent/bin/fingerprint <Device Name>
{
"name": "<Device Name>",
"machine.address": [
"00:01:C0:23:22:CD",
"00:01:C0:23:22:E0"
],
"tpm.ek.public": "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtcjmahxIIdvvtIqgYn/
xeMS3xy7MSAmD1Th9MDbDR9HVepiDD92OJwb5ARDjYnEUCRlrNfWGV5iXLZDQ7i8h4953GZhPLwK4x0MVcjPtPD8Of5fXx
dPepuLFl8IPh4zPU9GAaTnY1+jkK8ccQP57iqBUDBZa2UU6z
+VcWjAf7fnXzHug0meWExMFSMTi5qne7sNVbf8x54W6AceGO1KKSEKZpUWs0heTKIQBg4dZ3kWgws8HSvg07bd81CZPnaL
0EHuaShboZnRXcTErN7a86bOJGUlD23CPJHk6Gt1XaqJrFthfWRWqLHbao3XxFkmh2vacXc5Ifs
+CssTjbSzdNQIDAQAB",
"tpm.pcrs.sha256": [
"85749DAD791A4125477BF1454958D4647A95FC41A08219E9387F6546C4121E19",
"7B7228F53616F5E08E28408195E4185A051769910303C7CF5C5F6F424D5852DB",
"3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969",
"3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969",
"3B797EDC3BFB790010B485156AA52936A3D7AA87B9106D7C162C17CA1C840B5F",
"B3A66804696158C623E1793BF07FB3157269C4F10A0F09EA405683E9D4B04097",
"3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969",
"B5710BF57D25623E4019027DA116821FA99F5C81E9E38B87671CC574F9281439",
"4A8DC3CBC1A0D2434FB61B103ED79A9B05702935D92C94643F84D312C100C75A",
"D01A23BE3296064075393AA584E5646B182210FF3D03594893F35542BB022B39",
"31E21E1644983F4E013CE13476AB20424362FFFCFB31CF22AE491E56E2C20A1D",
"0000000000000000000000000000000000000000000000000000000000000000",
"0000000000000000000000000000000000000000000000000000000000000000",
"0000000000000000000000000000000000000000000000000000000000000000",
"0000000000000000000000000000000000000000000000000000000000000000",
"0000000000000000000000000000000000000000000000000000000000000000"
]
}
JSON file successfully generated
```

b Copy the TPM Endorsement Public Key to your clipboard.

c In the SmartHub INFER IoT Center console, navigate to the **Inventory** > **Devices** page and select the gateway to enroll.

d In the Devices -All Devices page, click the **Actions** drop-down menu and select **Create Gateway Credentials**.

e Paste the TPM Endorsement Public Key and click **CREATE**.

6 To enroll, log in to your TPM enabled gateway and run the following command:

```
/opt/vmware/iotc-agent/bin/DefaultClient enroll --auth-type=TPM
```

7 To enable your TPM enabled gateway for an automatic enrollment, perform the following steps:

a Go to /opt/vmware/iotc-agent/conf/iotc-agent.cfg.

The iotc-agent.cfg file lists the details about your gateway enrollment.

b Scroll down to the autoEnrollmentType parameter and change its value to 1.

This change enables your registered gateway to be enrolled automatically.

c You can also configure the retry interval by specifying a autoEnrollRetryIntervalSeconds value. The SmartHub INFER IoT Center server tries to enroll your whitelisted gateway after the specified interval. The default interval value is 300 seconds.

d Save the configuration and restart the SmartHub INFER IoTC Agent.

The following example is a sample iotc-agent.cfg file for auto enrollment:

```
/opt/vmware/iotc-agent/conf/iotc-agent.cfg
Auto Enrollment:
# Auto enrollment of a registered gateway. 0 - No auto enrollment and 1 - TPM based
autoEnrollmentType = 1
# Enrollment retry interval in seconds, should be > 0
autoEnrollRetryIntervalSeconds = 300
```

Results

You have successfully enrolled a TPM enabled gateway.

Onboard a Gateway Using Zero Touch Enrollment

This section lists the steps to onboard your gateway using the Zero Touch Enrollment method.

Prerequisites

- You must have the CREATE DEVICE and ZERO TOUCH ENROLLMENT permissions to perform this operation.
- You must have created a device template using the **ZERO TOUCH ENROLLMENT** as the enrollment provider type.
- The gateway must be a Dell gateway running Ubuntu server and must be **ZERO TOUCH ENROLLMENT** enabled.
- You must have created a CSV file that contains the list of devices to enroll. The CSV file must contain the following columns: HardwareId (mandatory), Model Number (optional), and Property Value (optional).

Procedure

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory > Devices**

   The Devices - All Devices page is displayed.

2. Click **Register**.

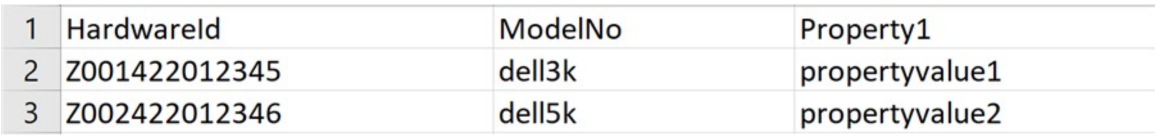
   The Register Gateway wizard is displayed.

3. Enter the batch name or file name and select the device template that has Zero Touch Enrollment enabled to associate with your gateway.

   Note The **Enrollment Type** is Zero Touch Enrollment and this cannot be changed.

4. Click **Upload** to upload the CSV file that contains a list of Hardware IDs of your gateways.The first row is reserved for the property name. The **HardwareId** field is mandatory. The **Model Number** and **Property value** fields are optional. Any values and names provided after the first column are displayed as custom properties on the registered gateways.

   Figure 4-1. CSV File



1	HardwareId	ModelNo	Property1
2	Z001422012345	dell3k	propertyvalue1
3	Z002422012346	dell5k	propertyvalue2

5. Click **REGISTER**.

Register Multiple Devices

You can use the Package Management CLI tool to register multiple devices to SmartHub INFER IoT Center using the Basic, Property-Based, TPM-Based, and Token-Based enrollment types.

Prerequisites

Ensure that you create a JSON file with the device template name, list of devices, their credentials, and properties, in the following format:

```
{
    "templateName": "property-template",
    "devices": [{
            "name": "Agent_x86_2",
            "credential": [{
                "key": "Serial Number",
                "value": "abc-56 4d fa a4 78 fa f2 88-24 3a 14 11 7d bd b8 b6"
            }],
            "property": [{
                "name":  "model",
                "value":  "xx  5K"
            }, {
                "name": "color",
                "value":  "white"
            }]
```

```
        },
        {
            "name": "Agent_x86_3",
            "credential": [{
                "key": "Serial Number",
                "value": "abc-56 4d 13 fb cc 73 82 2e-08 5d b1 c1 38 bf 1d 23"
            }],
            "property": [{
                "name":  "model",
                "value":  "xyi3b"
            }]
        }
    ]
}
```

The Package Management CLI tool uses this JSON file to read the devices list and register them to SmartHub INFER IoT Center.

Procedure

1 Download the Package Management CLI tool to your system. The Package Management CLI tool contains the following set of device commands that enable you to register your devices in bulk:

```
a01:iot-cli xyz$ ./bin/darwin_amd64/package-cli devices
Manage devices on Pulse IoT Center
Usage:
  package-cli devices [command]

Available Commands:
  register       Register device by given name on Pulse IoT Center
  register-all Register multiple devices to Pulse IoT Center. Expects JSON file with device
details.
  search         Search given device by name on Pulse IoT Center

Flags:
  -v, --api-version string      Pulse API version to use (default "1.0")
  -h, --help                    help for devices
  -s, --host-name string        Pulse IoT Center instance hostname <Required>
  -i, --insecure                Skip SSL certificate verification
  -l, --log-file-path string    Log file path (default "./iot-cli.log")

Use "package-cli devices [command] --help" for more information about a command.
```

2 Run the `register-all` command with the path to the JSON file that you created.

```
a01:iot-cli xyz$ ./package-cli devices register-all ./example-iotc-package/device-regd-property-
based.json -s https://10.92.85.41 -i
Username: sysadmin
Password:
Authentication successful.
Registering device...
Device registered with id: 30d75156-65b7-4658-b1e1-e7fba5008122 name: Agent_x86_2
Updating device property...
Property update successful for device
```

```
Creating device credentials...
Device credential successfully created for device
Registering device...
Device registered with id: a8f2dddb-8631-4d02-bfcd-b77febbf3a56 name: Agent_x86_3
Updating device property...
Property update successful for device
Creating device credentials...
Device credential successfully created for device

Successfully registered devices: 2
Total devices: 2
```

Results

The registered devices are listed in the **Inventory - Devices** page of the SmartHub INFER IoT Center console UI.

Whitelisting a Device

A whitelist is an explicit listing of gateways that are allowed for enrollment.

The whitelisting option allows you to control the gateways that are allowed to enroll and the gateways that are not permitted to enroll. A whitelisted gateway is a virtual gateway created on the SmartHub INFER IoT Center server. The virtual gateway is registered but not enrolled, and it does not have a physical gateway associated to it until a physical gateway is enrolled using the TPM-based authentication method. After registering a gateway using a device template that has the **Requires Whitelisting** option enabled, select **Whitelist** from the **Actions** drop-down menu to enroll the gateway.

Edit a Device Template

You can edit a device template from the **Inventory** > **Device Templates** page.

You can edit the system properties, custom properties, add metrics, add a connected device template, and add commands. However, you cannot edit the template name and device type.

Prerequisites

You must have the EDIT DEVICE TEMPLATE permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** > **Device Templates**. The Inventory - Device Templates screen is displayed.
2. Click on the device template that you want to edit.

   The device template's basic information is displayed in the **Basic Information** tab.
3. Scroll down to a property that you want to edit and click the edit icon.
4. Edit the device template properties by clicking the edit icon against each of the properties.

5 To save the changes, click **Save**.

Results

You have successfully edited a device template.

Example

1 Scroll down to the **System Properties** section and click the edit icon.

2 In the **Edit System Property** window, click the edit icon against a property to edit the property.

3 To delete a property, click the delete icon.

4 To add a new system property, click **+ ADD** and enter the property name. Click **DONE**.

5 To save the changes, click **SAVE**.

Working with Devices

5

After onboarding your device, you can perform the following operations:

- Collect metrics using the DefaultClient binary.
- Unenroll and delete a device
- Update the SmartHub INFER IoTC Agent version
- Send commands to the SmartHub INFER IoTC Agent
- Filter the list of devices based on the device type, enrollment state, and device template

This chapter includes the following topics:

Collect Metrics Using the DefaultClient Binary

After you install the SmartHub INFER IoTC Agent, a daemon process starts and the DefaultClient binary sends the default metrics such as CPU usage, memory usage, and disk usage to the SmartHub INFER IoTC Agent every 60 seconds. The IoTC Agent collects the metrics and sends them to the SmartHub INFER IoT Center server based on the metric interval time set in the device template. The default metric interval time is 60 seconds.

While creating a device template, ensure that you do not remove the **CPU-Usage**, **Memory-Usage**, and **Disk-Usage** metrics that are available in the template by default to monitor the performance of a gateway. Specifically, ensure that you do not change the metrics name and data type. The DefaultClient binary is available in the gateway at /opt/vmware/iotc-agent/bin/.

To run the binary with any other custom-defined metric in the device template, run the following command:

```
/opt/vmware/iotc-agent/bin/DefaultClient send-metric --device-id=<device Id> --name=<metric name> --
type=<string|integer|double|boolean> --value=<value>
```

Send a Command to the SmartHub INFER IoTC Agent

You can send a command to the SmartHub INFER IoTC Agent on your gateway from the SmartHub INFER IoT Center console.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Devices**.

The Inventory - Devices page lists the registered, enrolled, and unenrolled gateway and Thing devices.

2 Click the device for which you want to send a command.

3 From the **...** drop-down menu, select **Commands** and click **SEND COMMAND**.

4 In the Send Command window, select a command to send from the **Select Command** drop-down menu and enter its arguments.

5 Click **SEND COMMAND**.

The status of the command is displayed under **Command History**. Click the refresh button to refresh the status.

Send a Command to Multiple Devices

You can select up to 1000 devices of the same template and send a command from the SmartHub INFER IoT Center console.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Devices**.

The Inventory - Devices page lists the registered, enrolled, and unenrolled gateway and Thing devices.

2 Select the devices for which you want to send a command.

3 From the **...** drop-down menu, select and click **SEND COMMAND**.

4 In the Send Command window, select a command to send from the **Select Command** drop-down menu and enter its arguments.

5 Click **SEND COMMAND** to send the command to selected devices.

The status of the command is displayed under **Command History**. Click the refresh button to refresh the status.

Note You can verify the status of the device command update on multiple devices using the **Tasks** tab. For more information, see Chapter 13 Tasks.

View the List of Files

You can view a list of files uploaded by the devices in the SmartHub INFER IoT Center console.

Prerequisites

Ensure that your device is enrolled.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Devices**.

The Inventory - Devices page lists the registered, enrolled, and unenrolled gateway and Thing devices.

2 Click the device name for which you want to search a file associated with it.

3 From the **...** drop-down menu, select **Files** and click **Files**.

4 To search a file, enter the name of the file.

A list of files with the timestamp, size, modified date, and a download menu item are displayed.

5 Select a file and click download.

The file is downloaded on your local repository.

View the List of Devices Based on a State

You can view the list of the devices based on their state such as enrolled, registered, unenrolled, and deleted.

Prerequisites

You must be a **Device Administrator** to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory > Devices.**

The Devices - All Devices page lists the registered, enrolled, and unenrolled Gateway and Thing devices.

2 From the **Enrollment State** drop-down menu, select the state of the device and click **Apply**.

The list of devices with the selected state is displayed.

Note If you want to view all the deleted devices, select the **Deleted** check box. **Deleted** check box is not selected by default. You can only view the basic information of the deleted devices.

View the List of Devices Based on a Property

You can view the list of devices based on their property name and value.

Prerequisites

You must be a **Device Administrator** to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Devices**.

The Devices - All Devices page lists the registered, enrolled, and unenrolled Gateway and Thing devices.

2 From the **Properties** drop-down, select **Search Property Name** and enter the **Search Value**.

3 Click **OK** and **APPLY**.

The list of devices with the selected property is displayed.

Note

- The property keys listed in the drop-down is what is defined in the templates. Additional keys defined in the devices (not part of template) are not listed in the drop-down.
- The list of possible values for a given property key is from the actual device values for a given property key.

Update Bulk Custom Property on Multiple Devices

With SmartHub INFER IoT Center, you can add, delete, and update bulk properties on multiple devices.

Note The custom property that you update on a device does not impact the set of properties in the device template. To use the newly added keys in Advanced Search, you must edit the device template and add the keys.

Perform the following steps:

Prerequisites

You must be a **Device Administrator** to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory**> **Devices**

The Devices - All Devices page lists the registered, enrolled, and unenrolled Gateway and Thing devices.

2 Select the devices for which you want to update the custom property.

Note

- Devices can be from multiple templates.
- Device state can be Enrolled and Registered.
- Device Type can be GATEWAY and THING.

3 Navigate to the **...** drop-down menu and select **Edit Custom Property**.

The Custom Property dialog box with Edit and Delete key values is displayed.

4 To edit a custom property, select **Edit** and enter the **Name** and **Value** of the property.

5 Click **DONE** and **SAVE**.

A pop-up message with the number of properties added and deleted is displayed in the **Inventory**> **Devices** page.

6 To verify if the values are added, click the device name and **Properties**.

7 Click **Custom Properties**.

You can see that the properties are added.

8 To delete a custom property, select the devices from the list and navigate to **...** drop-down menu, select **Edit Custom Property**.

9 Click **Delete** and enter the name of the key.

10 Click **DONE** and **SAVE.**

A pop-up message with the number of properties added and deleted is displayed in the **Inventory**> **Devices** page.

11 To verify if the keys are deleted, click the devices name and **Properties** > **Custom Properties**.

The keys have been deleted from the **Custom Properties** for the devices.

Results

You can ADD, EDIT, or DELETE any custom property using the bulk command.

Unenroll a Device

To unenroll a gateway and its connected devices that is enrolled to the SmartHub INFER IoT Center server, perform the steps listed in this section.

Note The Gateway cannot communicate with the server after it is unenrolled. The data of the unenrolled device is still present on the SmartHub INFER IoT Center server. To delete the data, delete the device.

Prerequisites

- The device must be enrolled to the SmartHub INFER IoT Center server.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** >

   Devices. The Devices - All Devices page is displayed.
2. Click the gateway or Thing device that you want to unenroll.
3. Click the **Actions** drop-down menu and select **Unenroll**.
4. Confirm your action by clicking **UNENROLL**.

   Note This operation also unenrolls the connected devices.

Results

You have successfully unenrolled a device.

Delete a Device

After unenrolling a device, delete it so that the device no longer appears in the list of devices.

This action deletes the device data from the SmartHub INFER IoT Center server. You cannot retrieve the deleted data.

Prerequisites

You must have the Delete Device permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** >

   Devices. The Devices - All Devices screen is displayed.
2. Select the device and click the delete icon on the top-right side of the screen to delete.
3. In the confirmation dialog box, verify that you are deleting the correct device and click **DELETE**.

Results

You have successfully deleted a device.

View Metric Graphs

The metric graph data is aggregated if there are more than 1000 numeric metric values in a selected range. Each aggregated data point is an average of the values for a given time period within the range.

To view the graph of the metrics collected on your device, perform the following steps:

Prerequisites

Note String and Boolean values are not aggregated and are limited to the latest 1000 data points. To view all the data values, select a smaller time range.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Devices**.

  The Inventory - Devices page lists the registered and enrolled Gateway and Thing devices.

2 Click the device for which you want to view the metrics graphs and click the **Metrics** tab.

Results

By default, graphs are displayed for the following metrics:

- CPU-Usage (in %)
- Memory-Usage (in %)
- Disk-Usage (in %)

Working with the SmartHub INFER IoTC Agent

6

This section provides information about working with the SmartHub INFER IoTC Agent.

The SmartHub INFER IoTC Agent is a component that resides in the Gateway. It connects the SmartHub INFER IoT Center services to run commands and to send operational metrics to the IoTC services. The SmartHub INFER IoTC Agent offers an SDK that exposes APIs. Third-party applications can use these APIs on the Gateway to interact with SmartHub INFER IoT Center.

The SmartHub INFER IoTC Agent makes an outbound connection to the server on port 443

(HTTPS). This chapter includes the following topics:

- Using the IoTC Agent SDK

Using the IoTC Agent SDK

The IoTC Agent software developer's kit (SDK) provides C APIs to interact with the SmartHub INFER IoT Center through an agent called IoTC Agent from within a Gateway.

The IoTC Agent SDK contains the following:

- Libraries: Two libraries are available in the SDK:
  - Iotc-agent-sdk
  - iotc-agent-common
- A header file: iotcAgent.h
- A sample: DefaultClient.c

Writing a Client Application Using the IoTC Agent SDK

To write a client application using the IoTC Agent SDK, perform the following steps.

Procedure

1 Define an identifier for the client application:

```
IotcApplicationId clientAppId;
strncpy(clientAppId.id, "com.myclient", sizeof clientAppId.id);
```

2 Establish a session between the client application andIoTC Agent:

```
IotcSession *session;
session = Iotc_Init(&clientAppId);
if (session == NULL) {
    // Handle failure
}
```

3 After establishing a session, the client can invoke other APIs to perform operations.

Currently, the IoTC Agent API works in an asynchronous mode. When an API is invoked, a request is sent to the IoTC Agent and the API returns to the client. Now, the client invokes the `Iotc_GetResponse()` API to receive a response from the previously invoked API. For example:

```
/** Enrollment wrapper function */
static   int
EnrollGateway(IotcSession  *session,
        const char* templateName,
        const char* gatewayName,
        const char* username,
        const char* password)
{
    IotcEnrollmentRequest enrollmentRequest;
    IotcGetResponse getResponse;
    IotcEnrollmentResponse *resp;
    int status;

    enrollmentRequest.data.type = IOTC_NOT_REGISTERED;
    strncpy(enrollmentRequest.data.deviceDetails.deviceTemplate,
            templateName,
            sizeof enrollmentRequest.data.deviceDetails.deviceTemplate);
    enrollmentRequest.data.deviceDetails.deviceTemplate
        [sizeof enrollmentRequest.data.deviceDetails.deviceTemplate - 1] = '\0';
    strncpy(enrollmentRequest.data.deviceDetails.name, gatewayName,
            sizeof enrollmentRequest.data.deviceDetails.name);
    enrollmentRequest.data.deviceDetails.name
        [sizeof enrollmentRequest.data.deviceDetails.name - 1] = '\0';
    strncpy(enrollmentRequest.userCredentials.username,
            username,
            sizeof enrollmentRequest.userCredentials.username);
    enrollmentRequest.userCredentials.username
        [sizeof enrollmentRequest.userCredentials.username - 1] = '\0';
    strncpy(enrollmentRequest.userCredentials.password,
            password,
            sizeof enrollmentRequest.userCredentials.password);
    enrollmentRequest.userCredentials.password
        [sizeof enrollmentRequest.userCredentials.password - 1] = '\0';

    if (Iotc_Enroll(session, &enrollmentRequest) == -1) {
        fprintf(stderr, "Failed sending enroll request\n");
        return -1;
    }

    /* Invoke GetResponse by supplying type of response */
    status  =  Iotc_GetResponseByType(session, IOTC_ENROLL_RESPONSE,
                                         CLIENT_TIMEOUT, &getResponse);
    if (status == -1) {
        fprintf(stderr, "Enroll response failed for this client\n");
        return -1;
    }

    /* if the GeResponse succeeded, fetch the response */
    resp = getResponse.response;
    printf("Device Id: %s\nParent Device Id: %s\n",
            resp->deviceId.id, resp->parentId.id);
    printf("Status of enroll response: %d\n", status);

    /* Cleanup the memory used by the response object */
    Iotc_FreeGetResponse(&getResponse);
    return 0;
}
```

4 To disconnect a client from the IoTC Agent, invoke the following API:

```
Iotc_Close(session);
```

Example: Sample MyClient Source Code

```
/* ******************************************************
 * Copyright (C) 2019 VMware, Inc. All rights reserved.
 * -- VMware  Confidential
 * *****************************************************/

/**
 * @file  MyClient.c
 * @brief This file contains simple example code to demonstrate use of
 * iotc-agent-sdk APIs.
 * This example show how to use Iotc_Enroll API. Users can invoke
 * other APIs in similar manner.
 * This file also offers a utility function to read responses for a
 * API request.
 *
 * note: This is a simple demo example code.
 */

#include <stdio.h>
#include <string.h>

#include "iotcAgent.h"

/* timeout for waiting response from agent (in milliseconds) */
#define CLIENT_TIMEOUT 30000

/** Enrollment wrapper function */
static  int
EnrollGateway(IotcSession  *session,
        const char* templateName,
        const char* gatewayName,
        const char* username,
        const char* password)
{
    IotcEnrollmentRequest enrollmentRequest;
    IotcGetResponse getResponse;
    IotcEnrollmentResponse *resp;
    int status;

    enrollmentRequest.data.type = IOTC_NOT_REGISTERED;
    strncpy(enrollmentRequest.data.deviceDetails.deviceTemplate,
            templateName,
            sizeof enrollmentRequest.data.deviceDetails.deviceTemplate);
    enrollmentRequest.data.deviceDetails.deviceTemplate
        [sizeof enrollmentRequest.data.deviceDetails.deviceTemplate - 1] = '\0';
    strncpy(enrollmentRequest.data.deviceDetails.name, gatewayName,
            sizeof enrollmentRequest.data.deviceDetails.name);
    enrollmentRequest.data.deviceDetails.name
        [sizeof enrollmentRequest.data.deviceDetails.name - 1] = '\0';
    strncpy(enrollmentRequest.userCredentials.username,
            username,
            sizeof enrollmentRequest.userCredentials.username);
    enrollmentRequest.userCredentials.username
        [sizeof enrollmentRequest.userCredentials.username - 1] = '\0';
    strncpy(enrollmentRequest.userCredentials.password,
            password,
            sizeof enrollmentRequest.userCredentials.password);
    enrollmentRequest.userCredentials.password
        [sizeof enrollmentRequest.userCredentials.password - 1] = '\0';

    if (Iotc_Enroll(session, &enrollmentRequest) == -1) {
        fprintf(stderr, "Failed sending enroll request\n");
        return -1;
    }

    /* Invoke GetResponse by supplying type of response */
    status  =  Iotc_GetResponseByType(session, IOTC_ENROLL_RESPONSE,
                                      CLIENT_TIMEOUT, &getResponse);
    if (status == -1) {
        fprintf(stderr, "Enroll response failed for this client\n");
        return -1;
    }

    /* if the GeResponse succeeded, fetch the response */
    resp = getResponse.response;
    printf("Device Id: %s\nParent Device Id: %s\n",
            resp->deviceId.id, resp->parentId.id);
    printf("Status of enroll response: %d\n", status);

    /* Cleanup the memory used by the response object */
    Iotc_FreeGetResponse(&getResponse);
    return 0;
}


int main(int argc, char *argv[])
```

```
{
    IotcSession *session;
    IotcApplicationId clientAppId;
    const char *usage = "<template name> <gateway name> <username> <password>";

    if (argc != 5) {
        fprintf(stderr, "Usage:\n %s %s\n", argv[0], usage);
        return 1;
    }

    strncpy(clientAppId.id, "com.myclient", sizeof clientAppId.id);
    session = Iotc_Init(&clientAppId);
    if (session == NULL) {
         /* Handle failure */
        fprintf(stderr, "Could not initialize a session with iotc-agent\n");
        return 1;
    }

    /* Invoke a iotc-agent sdk API
        Note password is consumed as command line parameter for
        keeping thus example program simple */
    if (EnrollGateway(session, argv[1], argv[2], argv[3], argv[4]) == -1) {
        fprintf(stderr, "Enrollment failed\n");
    }

    /* Close the session */
    Iotc_Close(session);
    return 0;
}
```

What to do next

Build a client that uses the IoTC Agent SDK.

Building a Client That Uses the IoTC Agent SDK

Use the following steps to build a client that uses the IoTC Agent SDK.

Procedure

1 Extract the IoTC Agent SDK to a directory such as IOTC_DIR=/opt/iotc-sdk.

2 Compile the client application by entering the include directory and the libraries to link.

```
LD_LIBRARY_PATH=../lib gcc -o MyClient MyClient.c -I $IOTC_DIR/include/ -L $IOTC_DIR/lib -liotc-agent-sdk
```

Running a Client That Uses the IoTC Agent SDK

Clients using IoTC Agent SDK require the iotc group privilege or the root user privilege.

To run a client program with a non-root user privilege, you must include the iotc group in the supplemental groups and run the client program with the iotc group permission:

```
sudo usermod -a -G iotc $USER
sudo runuser $USER -G iotc -m -c "LD_LIBRARY_PATH=/opt/vmware/iotc-agent/lib ./MyClient"
```

Working with IoTC Agent CLI

The IoTC Agent CLI tool is a wrapper around the IoTC Agent's default client binary DefaultClient.

This tool provides a command-line interface (CLI) to perform IoTC Agent SDK operations. With the IoTC Agent CLI tool, you can build a client that operates with SmartHub INFER IoT Center using the IoTC Agent SDK. You can use the DefaultClient binary as a reference for building your client.

The IoTC Agent CLI tool provides the following CLI options:

```
/opt/vmware/iotc-agent/bin# ./iotc-agent-cli help
Usage:
DefaultClient <command> <params>

Available commands and parameters:
enroll
        --auth-type=REGISTERED --token=<authentication token>
enroll
        --auth-type=PROPERTY --key=<property key> --value=<property value>
enroll
        --auth-type=BASIC --template=<template name> --name=<gateway name>
        --username=<user name> [ --password=<prompt|file:<path>> ]
enroll-device
        --device-id=<device Id> --parent-id=<parent Id>
enroll-device
        --template=<template name> --name=<device name>
        --parent-id=<parent Id>
unenroll
        --device-id=<device Id>
schedule
        --type=<download|execution|activation>
        --campaign-id=<campaign Id>
        [ --start-time=<start time window> --end-time=<end time window> ]
get-commands
send-notification
        --entity-id=<entity Id> --definition-id=<definition Id>
        --key=<key> --value=<value>
send-metric
        --device-id=<device Id> --name=<metric name>
        --type=<string|integer|double|boolean> --value=<value>
        [ --device-id=<device2 Id> ...  ]
set-progress
        --campaign-id=<campaign Id> --progress=<progress string>
send-properties
        --device-id=<device Id> --key=<key> --value=<value>
        [ --key=<key2> --value=<value2> ... ]
get-properties
        --device-id=<device Id> --type=<system|custom>
```

```
start-daemon
        [ --config=<config file> ]
        [ --interval=<overriding metric interval> ]
stop-daemon
        [ --config=<config file> ]
sync
get-devices
        [ --parent-id=<Parent Id> ]

Note: Parameter names can be shortened as long as they are unique.
```

Use the IoTC Agent CLI tool to perform operations such as enrolling a device and setting properties for a device quickly.

Note Declare the library path explicitly if you see error messages such as: “error while loading shared libraries: libiotc-agent-sdk.so: cannot open shared object file: No such file or directory”. Run the following command:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/vmware/iotc-agent/lib/
```

The IoTC Agent CLI tool is available in the bin directory of IoTC Agent: /opt/vmware/iotc-agent/bin/iotc-agent-cli.

Example: DefaultClient in the IoTC Agent Package

The IoTC Agent package consists of a directory that contains the source code of the DefaultClient binary file and a makefile to build your client. You can modify this source code according to your requirement. The IoTC Agent package also contains a wrapper script to run DefaultClient. The iotc-agent/example/ directory contains the following files:

- client
- DefaultClient.c
- DefaultClient.h
- DefaultClientDaemon.c
- base64.c
- Makefile

Example: Using the DefaultClient Daemon

You can run the DefaultClient binary file as a daemon process in the background. In the daemon mode, DefaultClient connects to the IoTC Agent daemon and authorizes campaign call-backs automatically. It also fetches commands from the server at regular intervals. When additional options are specified, DefaultClient gathers the default CPU and Memory Usage metrics from the Gateway device and sends them periodically. You can perform the following operations using the DefaultClient daemon:

- Start the DefaultClient daemon without sending the default metrics:

```
$ DefaultClient start-daemon
```

- Start the DefaultClient daemon with default metrics every 10 minutes:

```
$ DefaultClient start-daemon --device-id=<device_id> --interval=600
```

- Stop the DefaultClient daemon.

```
$ DefaultClient stop-daemon
```

Using the IoTC Agent connection, the DefaultClient daemon accepts requests from the following pipe files if necessary:

- /tmp/iotc-defclient/input for an input request.
- /tmp/iotc-defclient/output for an output request.

The following sample illustrates how to get system properties using the DefaultClient daemon:

```
$ echo "get-properties --device-id=13c425e1-873a-43f0-a529-cb05289a8a40 --type=system" > /tmp/iotc-
defclient/input
$ cat /tmp/iotc-defclient/output
```

Example: Send Metrics API Example

The following client program demonstrates the use of the Send Metrics API:

```
/* *******************************************************
 * Copyright (C) 2019 VMware, Inc. All rights reserved.
 * -- VMware Confidential
 * *****************************************************/

/**
 * @file ExampleMetric.c
 * @brief This file contains simple example code to demonstrate use of
 * iotc-agent-sdk send metrics API.
 *
 */

#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
```

```
#include <unistd.h>
#include <sys/sysinfo.h>

#include "iotcAgent.h"

/* This sample client's application id */
#define TEST_CLIENT_ID "com.agent.test.metric"
#define MEM_USAGE "Memory-Usage"
/* timeout for waiting response from agent (in milliseconds) */
#define CLIENT_TIMEOUT 30000


/**
 * Return the current time in number of milliseconds since UNIX
 * epoch time.
 * @return A uint64_t that represents the current time.
 */
static uint64_t GetTimeStampMs(void)
{
    struct timeval timeVal = {.tv_sec = 0, .tv_usec = 0};
    uint64_t timeStamp;

    gettimeofday(&timeVal, NULL);
    //use milliseconds. Make sure the constants are unsigned long
    //long so that the computation does not overflow.
    timeStamp = (timeVal.tv_sec * 1000ULL) + (timeVal.tv_usec / 1000ULL);

    return timeStamp;
}


int main(int argc, char *argv[])
{
    IotcSession *session;
    IotcApplicationId clientAppId;
    struct sysinfo si;
    double memUsage;
    IotcMetric *memMetric;
    IotcGetResponse getResponse;
    int status;

    if (argc != 2) {
        printf("Usage: %s <deviceId>\n", argv[0]);
        return 1;
    }

    strncpy(clientAppId.id, TEST_CLIENT_ID, sizeof clientAppId.id);

    /* Initialize a session with the iotc-agent sdk */
    session = Iotc_Init(&clientAppId);
    if (session == NULL) {
        printf("Iotc_Init() failed");
        return -1;
    }
    /* Create a memory metric object */
    memMetric = malloc(sizeof (*memMetric) + sizeof (IotcDoubleValue));
    strncpy(memMetric->deviceId.id, argv[1],
            sizeof memMetric->deviceId.id);
    memMetric->deviceId.id[sizeof memMetric->deviceId.id - 1] = '\0';
    strncpy(memMetric->name, MEM_USAGE, sizeof memMetric->name);
    memMetric->name[sizeof memMetric->name - 1] = '\0';
    memMetric->type = IOTC_METRIC_DOUBLE;

    while (1) {
        if (sysinfo(&si) < 0) {
            printf("Error reading sysinfo\n");
            break;
        }

        /* Get the memory metric data */
        memUsage = (double) (si.totalram - si.freeram) * (double) 100 / (double) si.totalram;
        printf("mem: total=%ld free=%ld\n", si.totalram, si.freeram);
        memMetric->doubles[0].ts = GetTimeStampMs();
        memMetric->doubles[0].value = memUsage;
        /* Send the collected metric data */
        Iotc_SendMetric(session, memMetric);
        /* Get response for the send metric data */
        status = Iotc_GetResponseByType(session, IOTC_SEND_METRIC, CLIENT_TIMEOUT, &getResponse);
        if (status == -1) {
            fprintf(stderr, "Failed receiving send metric response\n");
        }

        Iotc_FreeGetResponse(&getResponse);
        sleep(5); // for 5 seconds
    }

    free(memMetric);
    /* Close the session with the agent */
    Iotc_Close(session);
    return 0;
}
```

Updating the SmartHub INFER IoTC Agent

You can update the SmartHub INFER IoTC Agent from the **Inventory** > **Devices** tab or by using OTA campaigns.

For information about compatible SmartHub INFER IoT Center Server and Agent versions, see the *SmartHub INFER IoT Center Release Notes*.

Update the SmartHub INFER IoTC Agent Using Campaigns

Update the SmartHub INFER IoTC Agent to a newer version using OTA Campaigns.

When upgrading the SmartHub INFER IoT Center server, IoTC Agent packages are created for each target OS and architecture, and are displayed in the **Packages** tab of the SmartHub INFER IoT Center console. These packages contain the specifications required to upgrade the SmartHub INFER IoTC Agent. As a Campaign Administrator, you can create a campaign with this IoTCP package and target it to run on those gateways that require an agent upgrade.

Note Ensure that you select the correct version of the IoTC Agent package when upgrading the SmartHub INFER IoTC Agent.

For more information about creating and running campaigns, see Chapter 8 Working with Campaigns.

Prerequisites

You must be a **Campaign Administrator** to perform this operation.

Update the SmartHub INFER IoTC Agent on Multiple Devices

Update the SmartHub INFER IoTC Agent on multiple devices from the **Inventory** > **Devices**

tab. Perform the following steps:

Prerequisites

You must be a **Device Administrator** to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** >**Devices**.

The Devices - All Devices page lists the registered, enrolled, and unenrolled Gateway and Thing devices.

2 Select the devices for which you want to update the agent.

Note You can also select devices from different templates.

3 From the **...** drop-down menu, select **Update Agent**.

4 Click **Confirm**.

The **Update Agent Package** confirmation dialog box displays the current version and the upgrade version of the agent.

Note You can verify the status of the SmartHub INFER IoTC Agent update on multiple devices using the **Tasks** tab. For more information, see Chapter 13 Tasks.

Update the SmartHub INFER IoTC Agent on a Device

Update the SmartHub INFER IoTC Agent on a specific gateway device on which it is

installed. Perform the following steps:

Prerequisites

You must be a **Device Administrator** to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Devices**.

The Devices - All Devices page lists the registered and enrolled Gateway and Thing devices.

2 Click the device for which you want to update the agent.

3 From the **Actions** drop-down menu, select **Update Agent**.

The **Update Agent Package** confirmation dialog box displays the current version and the upgrade version of the agent.

4 Click **CONFIRM**.

Results

The agent upgrade process is initiated. To view the status of the agent update, click the **Commands** tab. To download, click the download icon against the command.

Working with Update Packages

7

A package is an update unit containing all actions required for managing a device over the air.

Using packages, you can update the operating system of a gateway device, install or update an application, reconfigure the gateway settings, and update the BIOS and firmware. Create a package file using the Package Management CLI tool. The tool uses a specification file that defines all actions and any other metadata to run the campaign. For information about creating a specification file, see the Create a Specification File section. An update campaign consists of a package with its own specification file that indicates how the package is applied. When running, the package is processed in the order given by the lifecycle management (LCM) server.

Note For creating and uploading packages, ensure that you download the latest version of the Package Management CLI tool from the SmartHub INFER IoT Center Console.

This chapter includes the following topics:

- Create a Specification File
- Download the Package Management CLI Tool
- Generate an IoTCP Package
- Sample Script for Running a Campaign on a Thing Device

Create a Specification File

To create an IoTC package, a specification file (YML) is required. The .yml file describes the content of the package and its associated metadata. You must create a .yml file before creating an IoTC package.

The package-cli.zip archive contains a example-iotc-package folder. Review the contents in this folder before creating the IoTC package.

Alternatively, you can use the following sample YML file as a template to create a file named test_package.yml. Change the values in the test_package.yml file according to your organization's requirements.

```
package:
    manifest:
        headlessExecution: true
        lifecycle:
            # Note the paths written in the action sections. If they don't match
            # any of the install paths in the attachments section, the tool will
            # warn you, but it will make a package. This is so you can address
            # executables that are on the GW  but not in this package, but this
            # requires a full path to be specified in the action section.
            - phase: verify
              action: <parent directories of build machine>/example-iotc-package/package-source/
verify.sh
            - phase: execute
              action: <parent directories of build machine>/example-iotc-package/package-source/
execute.sh

              # This phase's action matches the install path of the validate.sh
              # attachment, so no warning will be issued.
            - phase: validate
              action: <parent directories of build machine>/example-iotc-package/package-source/
validate_package.sh

              # This phase's action points to an attachment that doesn't match
              # any install path, even though there is an attachment named activate.sh
            - phase: activate
              action:  activate.sh
            - phase: reset
              action: <parent directories of gateway>/reset.sh
    attachments:
        # path describe where the attachments are on the system you're building the package.
        # installPath describes where on the gateway the attachment will be installed.
        # If no installPath is added, the 'path' value will be used.
        # Any the directories in the installPaths that don't exist on the gateway
        # will be created.
        - path: <parent directories of build machine>/example-iotc-package/package-source/
test_file.txt
        - path: <parent directories of build machine>/example-iotc-package/package-source/verify.sh
        - path: <parent directories of build machine>/example-iotc-package/package-source/execute.sh
        - path: <parent directories of build machine>/example-iotc-package/package-source/validate.sh
          # This will install the validate.sh attachment
          # in the same directories but named validate_package.sh
          installPath: <parent directories of build machine>/example-iotc-package/package-source/
validate_package.sh
        - path: <parent directories of build machine>/example-iotc-package/package-source/activate.sh
        - path: <parent directories of build machine>/example-iotc-package/package-source/reset.sh
          # You can specify a completely different directory
          # for attachment installation
          installPath: <parent directories of gateway>/reset.sh

    name: hello_iotcp

    # This is one of the many ways you can create a multiline string in yaml
    description: "A test IoT Center package with\n
    a multiline description."

    version: 1.1.0

    # This tag is empty. That means the tool will default to noarch.
    architecture:
```

```
    # This is just a string so you could write anything, but there are
    # standardized names for many of the operating systems.
    # If this tag is empty, the tool will use the value which was
    # used to compile it for the certain OS (windows, linux or darwin).
    # If it can't read the compile-time value or it's empty, it will
    # default to noos.
    os: linux

    # This is  a simple array of strings which are just that -  labels by
    # which you could search a package in Pulse IoT Center
    labels:
      - test
      - example
```

In this example:

The attachments section lists the files to be included in the package:

- path - The path on the disk where the file to be included in the package is located.
- installPath - The path on the gateway where the attachments are installed.

Note

- If installPath is not specified, the path value is used.
- If any of the directories specified in the installPath do not exist, they are created on the gateway if the iotc-user has the required permissions.

The manifest section describes the package lifecycle and execution. It allows custom actions to be associated with lifecycle events.

The headlessExecution flag:

- Controls the automatic transition of each lifecycle phase, without any interaction. By default, the value is true.
- If headlessExecution is set to true and the IoTC Agent is configured with manifestExecution = ENABLED, then the campaign runs automatically without any interaction.
- If headlessExecution is set to false and the IoTC Agent is configured with manifestExecution = ENABLED, then the campaign scheduling depends on an external input such as DefaultClient or SDK client that must be registered with the IoTC Agent. The executable specified for a particular phase is run by the IoTC Agent at each lifecycle phase.
- If the IoTC Agent is configured with manifestExecution = DISABLED, then the headlessExecution property and the executable steps are ignored. Here, all the associated executables are disabled and an SDK client must be configured to run the campaign.

action - An executable file that performs the required tasks for the current phase. For example, the executable file performs tasks such as verifying the downloaded content, setting up the environment, running the installer, and validating whether the installation is successful. The executable file is run in an isolated shell that has the environmental variable DATADIR set to the path of the directory that contains the extracted package files. If relative paths are used, DATADIR is set to access the files. For example, the path to access the update_data.tar.gz file is:

```
$DATADIR/update_data.tar.gz
```

Note

- All the files from the package with relative paths are deployed in a unique directory at the default path that is configured in the IoTC Agent. The default path can be found in the iotc-agent configuration file, at agent host: /opt/vmware/iotc-agent/conf/iotc-agent.cfg.

  ```
  agentDataDirPath = /opt/vmware/iotc-agent/data
  ```

- Ensure that you provide appropriate access and execution rights to the files, if needed. You can provide permissions through the executables for the lifecycle phases.
- You can specify a relative or an absolute install path for the attachments. If you do not specify the install path, the Package Management CLI tool creates an install path for each attachment.

The lifecycle section defines the different lifecycle phases and the corresponding action to take for each phase. For the IoTC Agent to locate an action executable after the payload is extracted, and to run the executable, the action path must match the installPath in the attachments section. Or, the action path must point to an existing executable on the gateway. The specification file also specifies the external executables to run at each lifecycle phase.

The lifecycle phases are:

- ENTRYPOINT
- VERIFY
- EXECUTE
- VALIDATE
- ACTIVATE
- RESET

You can attach executables to all lifecycle phases except the ENTRYPOINT phase. The executable for the ENTRYPOINT phase must be present on the Gateway's file system and ready to run.

These lifecycle phases are optional. If you use lifecycle phases, you cannot modify the phases or the order in which the phase actions are run. If you do not specify executables in the action field, then no action is performed and the phase is considered to pass successfully and the lifecycle moves to the next phase. For example, if you do not provide a verify.sh executable in the VERIFY phase, the package runs without verification (other than the default verification steps provided in the packaging format, such as checksum and RPM signatures), and moves to the VALIDATE state. This process continues until the package moves to the ACTIVATE phase. The phases ACTIVATE and RESET are mutually exclusive. The update is either activated or reset depending on the VALIDATE phase result.

Note

- For all the executables that are attached for the action field, the IoTC Agent sets the execute permission to (700 / -rwx------) for the iotc user by default.
- If there are other executables listed in packages/files/scripts beside the executables that are specified in the action field, the author of the executables must manage the required permissions.

The architecture and os sections are strings that describe the operating system and architecture that the package is built for. If the architecture or os sections are not present or have empty values, the Package Management CLI tool detects the values. These values are supplied to the Package Management CLI tool when building the tool itself. The Package Management CLI tool is available in the following variants. These variants are available in a downloadable file within the package-cli.zip file:

- OS = linux, Architecture = amd64
- OS = darwin, Architecture = 386
- OS = windows, Architecture = amd64
- OS = windows, Architecture = 386

You cannot use a variant of the Package Management CLI tool that is not built for the specific system architecture or operating system. For example, you cannot use the Windows Package Management CLI tool on a Linux machine. This way, the Package Management CLI tool detects the system architecture or operating system and populates them with the values that are built into the tool. The default operating system values that are built in for detection are Windows, MacOS, and Linux. The tool defaults to noos if it is unable to detect an operating system. Similarly, the Package Management CLI tool defaults to noarch if it is unable to detect a system architecture.

Download the Package Management CLI Tool

This section lists the steps to download the Package Management CLI tool.

Procedure

1. Log in to SmartHub INFER IoT Center.
2. From the home screen, click the settings icon on the top right corner and click **Downloads**.
3. Under **Campaigns CLI**, download the **Package Management CLI** file to your local disk.
4. Extract the package-cli.zip file and run the package-cli file for your desired operating system.

Generate an IoTCP Package

Generate an *.iotcp package, upload it to the SmartHub INFER IoT Center, and run campaigns using the package.

The package command contains two subcommands:

```
> ./iot-cli package
Package software for Pulse IoT Center

Usage:
   package-cli package [command]

Available Commands:
   create        Generate an IoT Center package according to a package manifest
   upload        Upload a created package to Pulse IoT Center.

Flags:
   -h, --help     help for package

Use "package-cli package [command] --help" for more information about a command.
```

The Package Create Subcommand

The package create subcommand creates a package using a specification file. It contains a flag without any shorthand name:

```
> ./iot-cli help package create
Generate an IoT Center package according to a package manifest.

Usage:
   package-cli package create <path to package.yml> [flags]

Flags:
   -h, --help              help for create
       --no-approval       Disables user approval before creation
   -o, --output string     Set output path (default "./")
```

The create command has the following modification flags:

- -o, --output - Sets the output path for the package that is created.

The package create command requires a confirmation to create the package.

Example 1: Any input other than y stops the package creation process.

```
> ./iot-cli package create example-iotc-packagepackage-spec.yml

You are creating a package with:
   Name: hello_iotcp
   Version: 1.1.0
   OS: linux
   Architecture: noarch
File will be created as: hello_iotcp-1.1.0.linux.noarch.iotcp
```

```
Do you want to continue ? [y/n] y

Creating package hello_iotcp-1.1.0.linux.noarch.iotcp
```

Example 2: In this example, `y` is sent as an input to the command using bash `here string`.

```
> ./iot-cli package create example-iotc-package/package-spec.yml <<<y

You are creating a package with:
   Name: hello_iotcp
   Version: 1.1.0
   OS:  linux
   Architecture: noarch
File will be created as: hello_iotcp-1.1.0.linux.noarch.iotcp

Do you want to continue ? [y/n]
Creating package hello_iotcp-1.1.0.linux.noarch.iotcp
```

Example 3: The `--no-approval` flag is present:

```
> ./iot-cli package create example-iotc-package/package-spec.yml --no-approval

You are creating a package with:
   Name: hello_iotcp
   Version: 1.1.0
   OS:  linux
   Architecture: noarch
File will be created as: hello_iotcp-1.1.0.linux.noarch.iotcp

Creating package hello_iotcp-1.1.0.linux.noarch.iotcp
```

The Package Management CLI tool also supports pipes that do not require an approval. The approval flag is not required here:

```
> ./iot-cli package create
If the path to the yaml config is not passed as argument, you can use pipes to pass the yaml config
file.
Usage with pipes:              cat package.yaml | iot-cli package create
Usage with yaml parameter:     iot-cli package create test_pac.yml
```

For example:

```
cat example-iotc-package/package-spec.yml | ./iot-cli package create

You are creating a package with:
   Name: hello_iotcp
   Version: 1.1.0
   OS:  linux
   Architecture: noarch
File will be created as: hello_iotcp-1.1.0.linux.noarch.iotcp

Creating package hello_iotcp-1.1.0.linux.noarch.iotcp
```

Create an IoTC Package

Use the following steps to create an IoTC package.

Prerequisites

- Ensure that you have created a specification file.
- Download and install the Package Management CLI tool. You must have executable permissions to run this tool.

Procedure

- From the **Package Management CLI** tool, run the following command:

  ```
  package-cli package create <path-to-spec>
  ```

  Here, <path-to-spec> is the path to the YML file.

  Optionally, you can specify an output file using the -o flag. By default, the current directory is used as the output path. The resulting file is named {name}-{version}.{os}.{architecture}.iotcp.

What to do next

Upload the package using the SmartHub INFER IoT Center UI or by using the Package Management CLI tool. Run the $ package-cli upload package <path to package> <Pulse IoT Host> command. For example:

```
$ package-cli upload package UpdateVIPonGW-3.linux.noarch.iotcp https://<Pulse IoT Host IP>
```

Note The upload package command creates the package in the Root organization.

Upload the IoTC Package

This section lists the steps for uploading an IoTC Package.

The upload command contains two subcommands to handle the package and manifest uploads.

```
$ package-cli upload
Upload files to Pulse IoT Center

Usage:
  package-cli upload [command]

Available Commands:
  package     Upload a created package to Pulse IoT Center.

Flags:
  -h, --help    help for upload

Use "package-cli upload [command] --help" for more information about a command.
```

Prerequisites

- Download and install the **Package Management CLI** tool from SmartHub INFER IoT Center.

Procedure

- Run the $ package-cli upload package <path to package> <Pulse IoT Host> command.

  ```
  $ package-cli upload package myPackage.iotcp https://<Pulse IoT Host IP>
  ```

  Note The <Pulse IoT Host> must contain a valid schema (https:).

  You are prompted to enter a user name and password. Ensure that the user credentials you enter has sufficient privileges to upload packages.

Results

After the package uploads, the package's UUID is displayed on the console.

Sample Script for Running a Campaign on a Thing Device

The package to update a Thing device contains scripts that are run on the gateway where the Thing device is connected.

To enable package script development, SmartHub INFER IoTC Agent runs the package scripts with the *TARGET_THINGS* environment variable. This environment variable contains the space-separated Thing IDs that the campaign targets for updates.

Base on these Thing IDs, you as a script developer can obtain the required properties from the SmartHub INFER IoTC Agent's iotc-agent-cli command-line tool.

Sample Campaign Script

This following sample script provides information about updating the IP cameras that are connected to a gateway:

```
#!/usr/bin/env bash
if [ -z "$TARGET_THINGS" ];
then
    echo "No cameras are provided"
    exit 0
fi

# convert target thing ids to array
camera_ids=("$TARGET_THINGS")

for camera_id in "${camera_ids[@]}"
do
    echo "Updating IP camera with device ID=$camera_id"

    # Get the required thing properties
    camera_ip=`/opt/vmware/iotc-agent/bin/iotc-agent-cli get-properties --device-id="$camera_id" --
```

```
type=custom --property-name="IP"`
    # The get operation might fail, so appropriate error handling can be added here.
    camera_ip_successfully_retrieved=$?
    echo "Camera IP: $camera_ip"
    curl http://${camera_ip}/cgi/UpdateFirmware filename=firmware.bin
    update_result=$?

    if [ 0 -ne $update_result ]
    then
        failed_updates+=("$camera_id")
    fi
done

if [ ${#failed_updates[@]} -ne 0 ];
then
    echo "The update failed for:  ${failed_updates[@]}"
    exit 1
fi
echo "Successful"
exit 0
```

In this example, camera_ip=`/opt/vmware/iotc-agent/bin/iotc-agent-cli get-properties --device-id="$camera_id" --type=custom --property-name="IP"` gets the required properties of the IP camera such as IP address using the IP camera ID.

curl http://${camera_ip}/cgi/UpdateFirmware filename=firmware.bin sends the firmware updates to the IP cameras from the location mentioned in the script.

The following part of the script describes the error handling information when an update fails for one of the IP cameras:

```
if [ 0 -ne $update_result ]
    then
        failed_updates+=("$camera_id")
    fi
done

if [ ${#failed_updates[@]} -ne 0 ];
then
    echo "The update failed for: ${failed_updates[@]}"
    exit 1
fi
```

Working with Campaigns

8

You can create, update, or delete a campaign from the SmartHub INFER IoT Center UI.

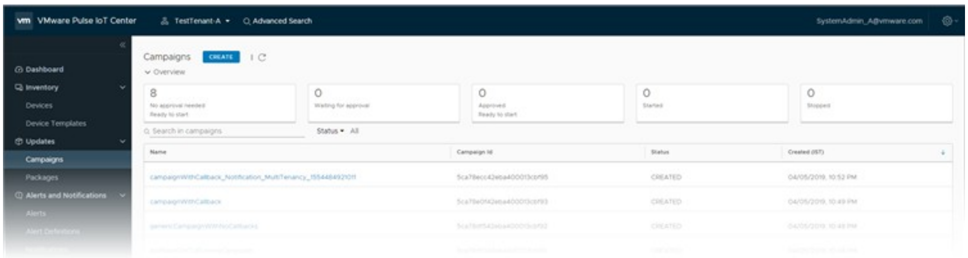
Using the **Updates** tab, you can run campaigns to update the software, firmware, operating system, and BIOS of your gateway device. You can also track the update status from here.

This chapter includes the following topics:

- Managing Campaigns
- Packages

Managing Campaigns

Use the **Campaigns** tab to create a campaign, create or add a Distribution Select Query to the campaign, associate update packages to the devices in your campaign, edit a campaign, or delete a campaign. You can also schedule a campaign to run at a specific date and time.



Name	Campaign Id	Status	Created (IST)
campaignWithCallback_Notification_MultiTenancy_1554484921011	5ca78ecc42eba400013cbf95	CREATED	04/05/2019, 10:52 PM
campaignWithCallback	5ca78e0f42eba400013cbf93	CREATED	04/05/2019, 10:49 PM
genericCampaignWithNoCallbacks	[illegible]	CREATED	04/05/2019, 10:48 PM

Using the **Campaigns** tab, you can specify an existing Distribution Select Query or create a new query to run a campaign. The Distribution Select Query periodically matches a subset of all the registered devices that fulfil the query. To add specific devices to the campaign, call the `addTargetGateways` API. For more information about Campaign Management APIs, see the *SmartHub INFER IoT Center API Reference Guide*.

If your campaigns require an approval before starting, toggle the **Enable Approval** option under the **Settings** > **Updates** tab. For information about enabling campaign approvals, see Update Settings. Users with permissions to edit organization settings can set the approvals for campaigns. Users with campaign approval permissions can approve a campaign.

Note Approval settings for existing campaigns will not change after updating your organization settings.

By default, campaigns do not require an approval when you create it unless you configure your organization's settings to check for campaign approvals. By default, the Campaign Administrator's role has permissions to approve campaigns. If necessary, a System Administrator can create a role with campaign approval permissions and remove the campaign approval permissions from the Campaign Administrator's role.

Use Cases for Campaign Approvals

The following use cases describe the campaign approval process.

Your organization does not require approvals for campaigns.

You need not make any changes.

Your organization requires approvals for campaigns.

A user with the Campaign Administrator role must approve the campaigns:

1. The System Administrator creates a user with the role to edit organization settings.
2. The user edits the Approve Campaign settings.
3. A user with the Campaign Administrator role approves the campaign.

Your organization requires approval for campaigns from a special role that has the Approve Campaign permission.

1. The System Administrator creates a special role with Approve Campaign permission. The user who approves campaigns must also have the following permissions:
   - View Campaign
   - View Package
   - View Organization Settings
   - View Package
   - View Filter Definition
   - Edit Filter Definition
2. The System Administrator removes the Approve Campaign permission from the Campaign Administrator role.
3. The System Administrator creates a user and assigns the special role that has the Approve Campaign permission.
4. The System Administrator creates a user with the Edit Organization Settings role. This user edits the Approve Campaign settings.
5. The user with the Campaign Administrator role creates campaigns.
6. The user with the special role approves the campaigns.

You must be a Campaign Administrator to perform the campaign operations.

Create a Campaign

This section lists the steps to create a campaign from the SmartHub INFER IoT Center UI.

Prerequisites

Create an IoTC Package. For information about creating an IoTC Package, see Chapter 7 Working with Update Packages.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Updates** >

   Campaigns. The Campaigns page is displayed.

2. Click **CREATE**.

   The **Create Campaign** wizard is displayed.

3. In the **Details** step:

   a. Enter a name and an optional description for your campaign.

   b. Optionally, select **Launch after creation** if you want to start the campaign immediately. This option is not available if **Campaign Approval** is enabled.

      If you do not enable approvals from the **Settings** > **Updates** page, the campaign wizard allows you to schedule a start date and time for your campaign.

      If you enable approvals and after the campaign is approved, you can set the scheduled start date and time for the campaign from the **Campaign Details** page.

   c. Click **Next**.

4. In the **Distribution Select Query** step:

   a. To run the campaign on all registered gateways, select **All registered gateways**.

   b. To run the campaign on a gateway or a Thing device, select **GATEWAY** or **THING** from the **Update device template type:** drop-down menu.

      Note If you select **THING**, add the `parentGatewayId` property in the columns when creating the distribution query.

c Select the distribution query for the campaign. Distribution queries are search filter definitions whose results are the devices on which the campaign runs. You can create distribution queries from the **Advanced Search** option. Alternatively, you can create a distribution query by selecting **+ Create New** from the **Distribution Select Query** drop-down menu:

1 In the **Device Template** drop-down menu, select the type of device template to filter.

2 In the **Key** drop-down menu, select metric that you want to filter the devices by.

3 In the **Operator** drop-down menu, select from **Contains**, **Equal to**, and **Not equal to** operators.

4 In the **Value** text box, enter the metric values.

5 Enter a name for the distribution list and click **SAVE AS**. The query is saved as a new Advanced Search query. You can select this query the next time you want to create a campaign that requires similar devices to be included.

d Click **NEXT**.

5 In the **Select Package** step, select the update package that you want to associate your campaign with. You can select multiple packages of the same type. Click **Next**.

Note You can edit packages and distribution lists for the campaigns that are created, but you cannot edit them for those campaigns that are in the **Approved** or **Started** state. To edit these packages and distribution lists, you must delete the campaign first.

6 In the **Scheduling** step, select **Enable schedule start** to select a time window for campaign to start and end. This step is enabled if the campaign does not require an approval.

7 In the **Review** step, review your campaign information and click **CREATE**.

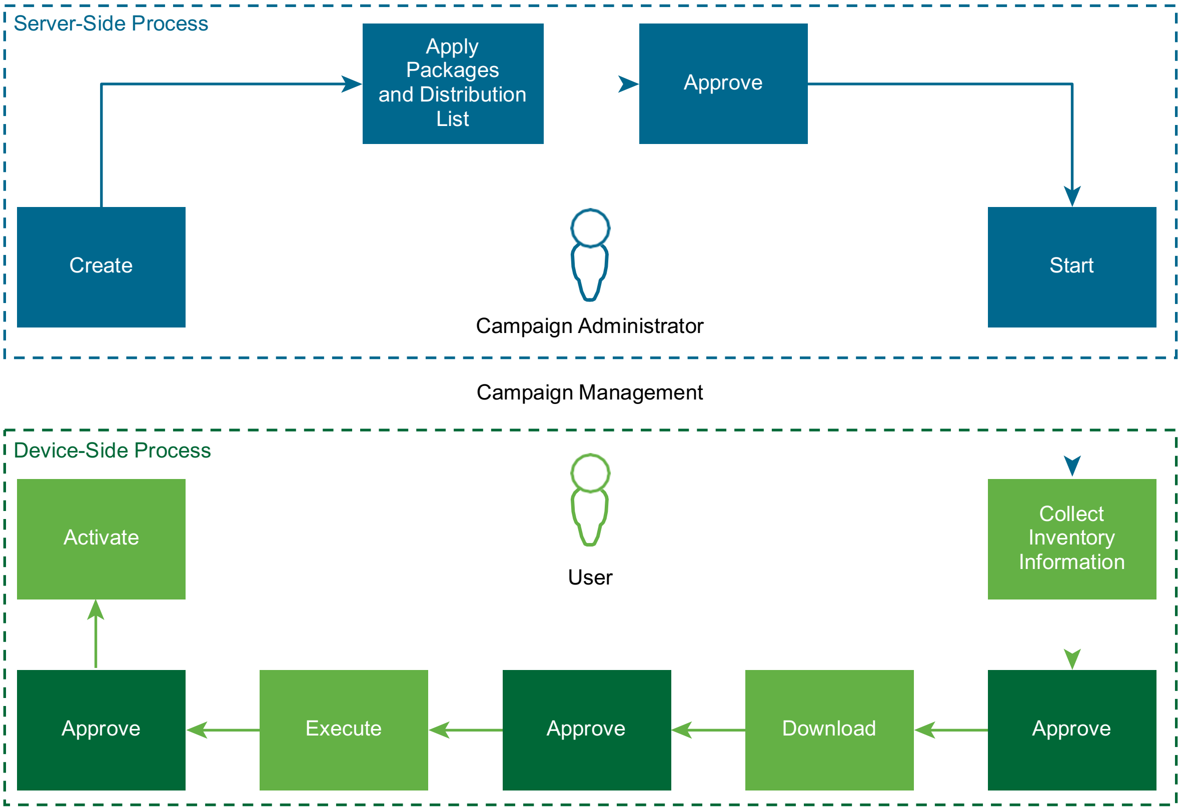
Results

You have successfully created a campaign.

Note After you start a campaign, the distribution list is evaluated and the resulting devices are added to the campaign. It takes 30 minutes for the newly enrolled devices that match the distribution list's criteria to be added to the campaign. You cannot edit the filter definition list after the campaign starts. You cannot start a scheduled campaign manually.

Example: Campaign State Transition Scheme

The following diagram illustrates the different states of a campaign.



Note For a headless campaign execution, approvals for downloading, executing, and activating packages are not required.

What to do next

Run the campaign that you have created.

Run the Campaign

Run the campaign that you have created. The campaign adds and processes only the gateways or Thing devices that are in the ENROLLED state.

Prerequisites

- You must have created a campaign and it must be in the Approved state.
- You cannot start a scheduled campaign manually.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Updates** >

   Campaigns. The Campaigns page is displayed.

2. Click that campaign that you have created.

   Details about the campaign are displayed.

3 Click the **Actions** drop-down menu and click **Start**.

Results

The campaign runs on all the enrolled Thing devices on a gateway whose **parentGatewayId** is listed in the distribution search query. The campaign is processed as a whole on all the devices that match the search criteria and are connected to one gateway, and they transition through the campaign states together. When you enroll a new Thing device to your gateway and if it matches the campaign distribution query, the campaign restarts on all the connected Thing devices irrespective of their states. To view the state of your campaign, click the **Devices** tab.

Edit a Campaign

This section lists the steps to edit a campaign's details.

Prerequisites

Note You can edit only those campaigns that are in the CREATED state. When a campaign transitions from the STOPPED state to the STARTED state, you can modify only its name. You can edit an approved campaign when the system does not require approvals. You cannot edit the distribution query.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Updates** >

Campaigns. The Campaigns page is displayed.

2 Select the campaign, and under the **Actions** drop-down menu, click **Edit**.

The Edit Campaign wizard is displayed.

3 Edit the campaign's details and click **SAVE**.

Results

You have successfully edited a campaign.

Delete a Campaign

You can delete campaigns in the CREATED, STOPPED, and APPROVED states.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Updates** >

Campaigns. The Campaigns page is displayed.

2 Select the campaign, and under the **Actions** drop-down menu, select **Delete**.

A confirmation dialog box is displayed. If the campaign is incomplete, select the check box before confirming the delete operation.

3 Click **DELETE**.

Results

You have successfully deleted a campaign.

Packages

The **Packages** tab allows you to view, upload, and download update packages for your devices.

To upload an update package, perform the following steps:

Prerequisites

You must have the permissions of a `Package Administrator` to perform this operation.

For creating a campaign, ensure that you upload only `.iotcp` files from the **Packages** tab.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Updates** >

   Packages. The Packages page is displayed.
2. Click **UPLOAD**.
3. Select the packages file from your system and click **Open**.

   A progress bar displays the progress of the upload. After the package is uploaded, it appears in the Packages page.

   Note The package progress disappears if you switch to any other page or tab in the SmartHub INFER IoT Center console UI.

Results

You have uploaded the update package successfully.

What to do next

- Use the package to create the campaign.
- Download the package if required. To download a package, select the package and click the download icon on the right side of the page.
- On the SmartHub INFER IoT Center console UI, the file upload and download operations run in the background and the results are displayed after the operations are complete.
- You cannot overwrite packages or delete them when they are associated with a campaign that is approved or started at least once.
-

Note From release 2019.09 onwards, you cannot upload packages to SmartHub INFER IoT Center using the 2019.07 version of the Package Management CLI tool. Ensure that you download the latest version.

Running a Campaign Using the Agent SDK

9

This chapter provides the prerequisites and steps to run over-the-air (OTA) updates on a Gateway, using the Agent SDK.

Campaign services use the following properties from the IoTC Agent:

- `commandFetchIntervalSeconds`: The IoTC Agent makes periodic `get-command` request to the Pulse micro services for every `commandFetchIntervalSeconds` expiry. You can configure the property value through the **Device Template** tab in the SmartHub INFER IoT Center console.

By default, the IoTC Agent runs with the following property values:

```
commandFetchIntervalSeconds=30

manifestExecution=ENABLE
```

If you start the IoTC Agent with the default properties, then the IoTC Agent requests for the command instructions from the SmartHub INFER IoT Center server by calling the `get-command` every 30 seconds. For each lifecycle phase, the IoTC Agent receives a command from the server to perform the download, execute, and activate operations.

Run a Campaign Using Default Properties

Perform the following steps to run an OTA update for the IoTC Agent using default properties.

Prerequisites

- Using the `package-cli` tool, perform the following steps:

  a Create an IoTC Package. For more information about creating an IoTC Package, see Create an IoTC Package.

  b Upload the IoTC Package to the repository. Alternatively, use the SmartHub INFER IoT Center UI to upload to the repository. For more information about uploading the IoTC Package, see Upload the IoTC Package.

- Enroll devices.

Procedure

1 Create a campaign using a distribution select query and the packages that you uploaded.

2 Start the campaign.

Results

The IoTC Agent auto-polls the command instructions every 30 seconds. The campaign states flow from INITIALIZED to COMPLETED after a series of get-commands calls to the Campaign server.

Run a Campaign in the On-Demand Mode

Perform the following steps to run an OTA update for the IoTC Agent in the On-Demand mode, that is, with the commandFetchIntervalSeconds property set to 0. This property value is defined in the device template.

Prerequisites

- In the specification file, set the value of the headlessExecution execution property to false.
- Using the package-cli tool, perform the following steps:
  a Create an IoTC Package. For more information about creating an IoTC Package, see Create an IoTC Package.

  b Upload the IoTC Package to the repository. Alternatively, use the SmartHub INFER IoT Center UI to upload to the repository. For more information about uploading the IoTC Package, see Upload the IoTC Package.
- Set the value of the commandFetchIntervalSeconds to 0 when creating the device template.

```
commandFetchIntervalSeconds = 0
```

- Enroll the device.

Procedure

1 Create a campaign using a distribution select query and the packages that you uploaded while creating the campaign.

2 Start the campaign.

Results

The IoTC Agent invokes the get-commands when initiated from the DefaultClient binary. The following example outlines the different states of the Gateway during an OTA update. The state of the Gateway is INSTANTIATED when the OTA campaign starts.

Example: Sample Workflow

1 Invoke the get-commands to call from the DefaultClient or an Agent SDK extension. The state of the Gateway changes to INVENTORY_UP_TO_DATE.

2 Invoke the get-commands to call from the DefaultClient or an Agent SDK extension. The state of the Gateway changes to WAITING_FOR_*_APPROVAL.

In the WAITING_FOR_*_APPROVAL state, schedule the next state. For example:

```
DefaultClient schedule --type=download --campaign-id=<campaign id>
DefaultClient schedule --type=download --campaign-id=<campaign id> --start-time=0 --end-time=0
DefaultClient schedule --type=download --campaign-id=<campaign id> --start-time=5000 --end-
time=80000
```

Based on the campaign scheduled time, the state of the device changes from SCHEDULED_DOWNLOAD to WAITING_FOR_DOWNLOAD.

3. Invoke the get-commands to call from the DefaultClient or an Agent SDK extension. The Gateway starts downloading the package and the state of the device changes from DOWNLOADING to DOWNLOAD_COMPLETE.

4. Invoke the get-commands to call from the DefaultClient or the Agent SDK extension. The state of the Gateway changes to WAITING_FOR_EXECUTION_APPROVAL.

   Here, you can schedule a start and end time for running the campaign using the following command:

   ```
   DefaultClient schedule --type=<download|execution|activation> --campaign-id=<campaign Id> [--
   start-time=<start time window> --end-time=<end time window>]
   ```

   For example:

   ```
   DefaultClient schedule --type=execution --campaign-id=<campaign id>
   DefaultClient schedule --type=execution --campaign-id=<campaign id> --start-time=0 --end-time=0
   DefaultClient schedule --type=execution --campaign-id=<campaign id> --start-time=5000 --end-
   time=80000
   ```

   Based on the campaign scheduled time, the state of the device changes from SCHEDULED_EXECUTION to WAITING_TO_EXECUTE.

   Here, you can schedule a start and end time for activating the campaign using the following command:

   ```
   DefaultClient schedule --type=<download|execution|activation> --campaign-id=<campaign Id> [--
   start-time=<start time window> --end-time=<end time window>]
   ```

   For example:

   ```
   DefaultClient schedule --type=activation --campaign-id=<campaign id>
   DefaultClient schedule --type=activation --campaign-id=<campaign id> --start-time=0 --end-time=0
   DefaultClient schedule --type=activation --campaign-id=<campaign id> --start-time=5000 --end-
   time=80000
   ```

Based on the campaign scheduled time, the state of the device changes from SCHEDULED_ACTIVATION to WAITING_TO_ACTIVATE.

Note Contact your Device Administrator or Campaign Administrator if the state of the Gateway changes to one of the following states:

- DOWNLOAD_FAILED
- EXECUTION_FAILED
- ACTIVATION_FAILED

Run a Campaign in the Headless Mode

This section lists the prerequisites for running a campaign for the IoTC Agent in Headless Mode.

Prerequisites

- Run the IoTC Agent with the manifestExecution property set to ENABLE:

  ```
  manifestExecution=ENABLE
  ```

Results

On any campaign, the get-commands call ensures that the OTA updates are auto-delivered to the IoTC Agent. The get-commands calls from the IoTC Agent listens to the Campaign commands and the campaign downloads, executes, and activates updates.

What to do next

To monitor the progress of a campaign on the gateway, set the agentLogLevel to 6 in the iotc-agent.cfg file. You can then monitor the system logs to view the progress of the campaign using tools such as journalctl -u or iotc-agent -f.

Approving the OTA Update Phases

Depending on the IoTC Agent configuration and the package property for headless execution, there are check points in the device or gateway that may require an approval for the campaign to run.

You can configure your OEM or SI application to use these checkpoints to schedule a maintenance window for updates, or for approving the campaign to run the updates. You can monitor the device or gateway's campaign progress from the **Campaigns** tab in the SmartHub INFER IoT Center UI. To view the progress of the campaign, select the campaign from the list and click the **Devices** tab.

Note The default interval for the IoTC Agent to fetch new commands from the server is 30 seconds. You can change the interval value through the Device Templates settings in the SmartHub INFER IoT Center console.

Use the following commands to configure the campaign execution settings using the IoTC Agent SDK or the IoTC Agent CLI:

- After the campaign reaches the **Waiting for Download Approval** state:

  ```
  iotc-agent-cli schedule --type=download --campaign-id=<campaign Id>
  ```

  Note Copy the campaign ID from the Campaigns page of the SmartHub INFER IoT Center UI.

- After the campaign reaches the **Waiting For Execution Approval** state:

  ```
  iotc-agent-cli schedule --type=execution --campaign-id=<campaign Id>
  ```

- After the campaign reaches the **Waiting For Activation Approval** state:

  ```
  iotc-agent-cli schedule --type=activation --campaign-id=<campaign Id>
  ```

Working with Alerts and Notifications 10

Use the **Alerts and Notifications** tab to monitor alerts and notifications, and to create alert and notification definitions.

Alert Definition: An alert definition is a specification of conditions that are necessary to trigger an alert. An alert is a specific instance in time when a device meets the conditions specified in the alert definition. Alert definitions consist of a device template, device metric, condition expression, and the number of times the condition must be true for a device to trigger that alert. For example, you can define an alert to trigger whenever the temperature of a device exceeds 130 degrees. You can set a pre-defined notification definition in your alert definition to notify the users through email or a user-defined callback API whenever the alert is triggered.

Device Offline Alerts: You can create alerts definitions on device templates to trigger an alert when a device does not ingest any metrics for a particular time duration.

Note For a newly added device that does not have any ingested metrics, at least two data points must be ingested before the absence of more metrics triggers a Device Offline alert.

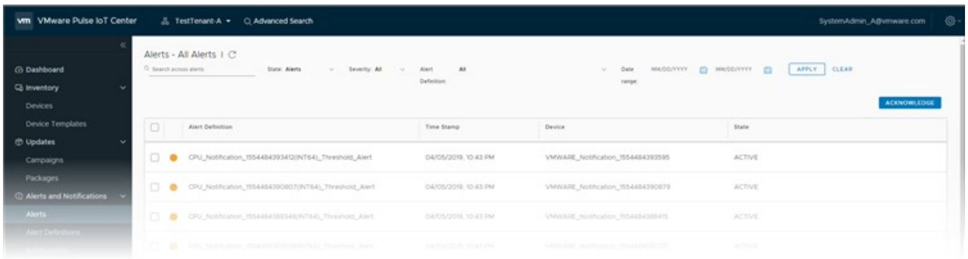
Notification Definition: Notification definitions define where the notification must be sent, who the sender is, and the number of times to retry. The Notifications feature enables you to receive timely notifications without logging in to the SmartHub INFER IoT Center, or without providing an integration point into the existing monitoring systems. The Notifications feature acts as a primary interface for all HTTP and email (SMTP) notifications from the SmartHub INFER IoT Center server. All the other SmartHub INFER IoT Center services communicate with the notifications service to send notifications to the external servers.

This chapter includes the following topics:

- Alerts
- Alert Definitions
- Notifications

Alerts

From the **Alerts and Notifications** > **Alerts** tab in the SmartHub INFER IoT Center UI, you can search for alerts by their definition, by the update time range, and by their states (active, acknowledged, or canceled). You can also acknowledge alerts from this tab.



☐		Alert Definition	Time Stamp	Device	State
☐	●	CPU_Notification_1554484393412(INT64)_Threshold_Alert	04/05/2019, 10:43 PM	VMWARE_Notification_1554484393595	ACTIVE
☐	●	CPU_Notification_1554484390807(INT64)_Threshold_Alert	04/05/2019, 10:43 PM	VMWARE_Notification_1554484390879	ACTIVE
☐	●	CPU_Notification_1554484388348(INT64)_Threshold_Alert	04/05/2019, 10:43 PM	VMWARE_Notification_1554484388415	ACTIVE

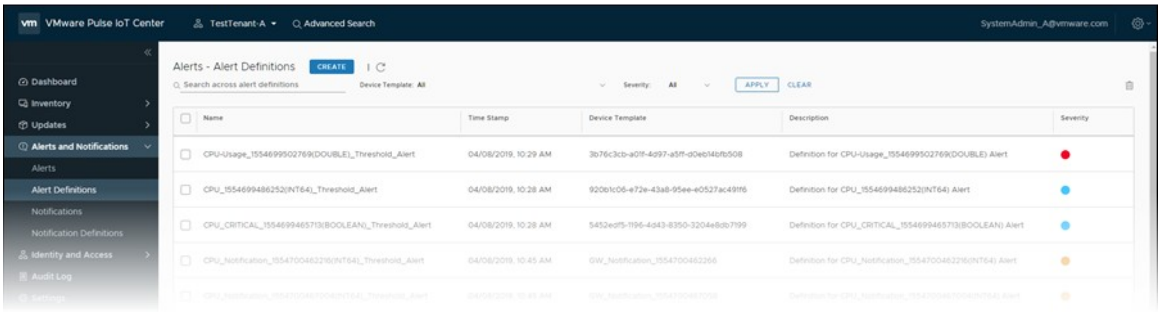
To acknowledge an active alert from the list of alerts, select the alert and click **ACKNOWLEDGE**. The alert's state changes to **Acknowledged** and the user name of the person who acknowledged the alert is displayed.

You can view the alert history of a device and determine if an alert is new or an existing one. You can also determine the number of times the device has raised this alert, identify the metric, value, and the time stamp when the alert was triggered. Select the alert definition and click the **History** tab. A graph indicating the alert states is also displayed.

A notification is sent out when an alert transitions to the **CANCELLED** state.

Alert Definitions

Alert definitions are a combination of symptoms and recommendations that you combine to identify problem areas and generate alerts.



☐	Name	Time Stamp	Device Template	Description	Severity
☐	CPU-Usage_1554699502769(DOUBLE)_Threshold_Alert	04/08/2019, 10:29 AM	3b76c3cb-a01f-4d97-a5ff-d0eb14bfb508	Definition for CPU-Usage_1554699502769(DOUBLE) Alert	●
☐	CPU_1554699486252(INT64)_Threshold_Alert	04/08/2019, 10:28 AM	920b1c06-e72e-43a8-95ee-e0527ac491f6	Definition for CPU_1554699486252(INT64) Alert	●
☐	CPU_CRITICAL_1554699465713(BOOLEAN)_Threshold_Alert	04/08/2019, 10:28 AM	5452edf5-1196-4d43-8350-3204e8db7199	Definition for CPU_CRITICAL_1554699465713(BOOLEAN) Alert	●
☐	CPU_Notification_1554700462216(INT64)_Threshold_Alert	04/08/2019, 10:45 AM	GW_Notification_1554700462266	Definition for CPU_Notification_1554700462216(INT64) Alert	●

You can perform the following actions:

- Create an alert definition.
- Edit an alert definition.
- Delete an alert definition.

Creating an Alert Definition Across Multiple Templates

You can create a single alert definition that works across multiple device templates. When you create alert definitions for multiple device templates, ensure that the metrics or properties for the alert definition symptom is common across all the device templates that you have selected.

Prerequisites

You must have the **Create Alert Definition** permission to perform this operation. For more information about roles and permissions, see Roles and Permissions.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Alert Definitions**. The Alerts - Alert Definitions page is displayed.
2. Click **CREATE**.

   The **Create Alert Definition** wizard is displayed.
3. In the **Details** step, enter the following information:
   a. **Name** - Enter a name and description for your alert.
   b. **Description** - Enter the alert description.
   c. Click **Next**.
4. In the **Devices** step, click **TEMPLATES** and select the templates to be added. Click **NEXT**.
5. In the **Conditions** step, add conditions to trigger your alert.
   - For information about creating a THRESHOLD alert definition, see Creating a Threshold Alert Definition.
   - For information about creating an OFFLINE alert definition, see Creating an Offline Alert Definition.
6. In the **Recommendation** step, enter the action that the technician or administrator must take when the alert is triggered, and click **Next**.
7. In the **Notification** step, perform the following steps:
   a. To send notifications when an alert is triggered or canceled, select **Enable Notifications**.
   b. From the **Select Notification** drop-down menu, select the notification to be sent.
8. Click **Next**.
9. In the **Review** step, review the information that you have provided and click **SAVE**.

Results

You have successfully created an alert definition across multiple device templates.

Creating an Alert Definition for Devices in an Advanced Search Query

If you have a saved advanced search query for devices, you can create an alert definition for those devices that are part of the query.

Prerequisites

You must have the **Create Alert Definition** permission to perform this operation. For more information about roles and permissions, see Roles and Permissions.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Alert Definitions**. The Alerts - Alert Definitions page is displayed.
2. Click **CREATE**.

   The **Create Alert Definition** wizard is displayed.
3. In the **Details** step, enter the following information:
   a. **Name** - Enter a name and description for your alert.
   b. **Description** - Enter the alert description.
   c. Click **Next**.
4. In the **Devices** step, click **SAVED SEARCH** and select the Advanced Search Query to be added. Alerts are triggered on all the devices that appear in this advanced search query result. Click **NEXT**.
5. In the **Conditions** step, add conditions to trigger your alert.
   a. Under **Symptom**, select a metric or system property, its condition, and enter the metric value to trigger the alert.

      Note Since system properties have string values, the allowed conditional operators are = and **!**=.

   b. From the **Severity** drop-down menu, select the severity of the alert:

      The alert severity types are:

      - INFO - Blue
      - WARNING - Orange
      - CRITICAL - Red
      - NORMAL - Defines the normal threshold for the alert to cancel.

   c. Click **+ ADD** to add multiple conditions and severity options to the symptom.
   d. From the **Trigger Count** drop-down menu, select the number of times the condition must be met to trigger the alert.

      Note For system property-based alerts, it is recommended to set the trigger count to 1.

   - For information about creating a THRESHOLD alert definition, see Creating a Threshold Alert Definition.
   - For information about creating an OFFLINE alert definition, see Creating an Offline Alert Definition.

6 In the **Recommendation** step, enter the action that the technician or administrator must take when the alert is triggered, and click **Next**.

7 In the **Notification** step, perform the following steps:

a To send notifications when an alert is triggered or canceled, select **Enable Notifications**.

b From the **Select Notification** drop-down menu, select the notification to be sent.

8 Click **Next**.

9 In the **Review** step, review the information that you have provided and click **SAVE**.

Results

You have successfully created an alert definition for an advanced search query.

Creating a Threshold Alert Definition

This section lists the steps to create a threshold alert definition for a single device template or across multiple device templates from the SmartHub INFER IoT Center UI.

- **Creating alert definitions for system properties**: You can create an alert definition for triggering a threshold alert whenever a system property value meets the alert definition symptom.

Prerequisites

You must have the **Create Alert Definition** permission to perform this operation. For more information about roles and permissions, see Roles and Permissions.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Alert Definitions**. The Alerts - Alert Definitions page is displayed.

2 Click **CREATE**.

The **Create Alert Definition** wizard is displayed.

3 In the **Details** step, enter the following information:

a **Name** - Enter a name and description for your alert.

b **Description** - Enter the alert description.

c Click **Next**.

4 In the **Devices** step:

- To create an alert definition across multiple templates, see Creating an Alert Definition Across Multiple Templates.

- To create an alert definition for devices in an advanced search query, see Creating an Alert Definition for Devices in an Advanced Search Query.

5 In the **Condition** step, add conditions that trigger your alert:

To set the threshold for triggering alerts, select the **THRESHOLD** tab and enter the following information:

a Under **Symptom**, select a metric or system property, its condition, and enter the metric value to trigger the alert.

Note Since system properties have string values, the allowed conditional operators are = and **!=**.

b From the **Severity** drop-down menu, select the severity of the alert:

The alert severity types are:

- INFO - Blue
- WARNING - Orange
- CRITICAL - Red
- NORMAL - Defines the normal threshold for the alert to cancel.

c Click **+ ADD** to add multiple conditions and severity options to the symptom.

d From the **Trigger Count** drop-down menu, select the number of times the condition must be met to trigger the alert.

Note For system property-based alerts, it is recommended to set the trigger count to 1.

6 Click **Next**.

7 In the **Recommendation** step, enter the action that the technician or administrator must take when the alert is triggered, and click **Next**.

8 In the **Notification** step, perform the following steps:

a To send notifications when an alert is triggered or canceled, select **Enable Notifications**.

b From the **Select Notification** drop-down menu, select the notification to be sent.

9 Click **Next**.

10 In the **Review** step, review the information that you have provided and click **SAVE**.

Results

You have successfully created a threshold alert definition.

Creating an Offline Alert Definition

This section lists the steps to create an offline alert definition from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the **Create Alert Definition** permission to perform this operation. For more information about roles and permissions, see Roles and Permissions.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Alert Definitions**. The Alerts - Alert Definitions page is displayed.

2 Click **CREATE.**

Create Alert Definition wizard is displayed.

3 In the **Details** step, enter the following information:

a **Name** - Enter a name and description for your alert.

b **Description** - Enter the alert description.

c Click **Next**.

4 In the **Devices** step:

- To create an alert definition across multiple templates, see Creating an Alert Definition Across Multiple Templates.
- To create an alert definition for devices in an advanced search query, see Creating an Alert Definition for Devices in an Advanced Search Query.

5 You can trigger device offline alerts if the SmartHub INFER IoT Center server does not receive metrics or system property values for a specified duration. In the **Condition** step, click **OFFLINE** and enter the duration (in minutes) that the SmartHub INFER IoT Center server must wait to receive metrics and system property values from a device before triggering the alert.

Note You can create device offline alert definitions only for the allowed metrics that are on the device template.

6 Select the Alert severity. The severity types are:

- INFO - Blue
- WARNING - Orange
- CRITICAL - Red

7 Click **Next**.

8 In the **Recommendation** step, enter the action that the technician or administrator must take when the alert is triggered, and click **Next**.

9 In the **Notification** step, perform the following steps:

a To send notifications when an alert is triggered or canceled, select **Enable Notifications**.

b From the **Select Notification** drop-down menu, select the notification to be sent.

10 Click **Next**.

11 In the **Review** step, review the information that you have provided and click **SAVE**.

Results

You have successfully created an offline alert definition.

Editing an Alert Definition

When you edit the values of an alert definition such as Template, Symptom, or Trigger Count, the system cancels all active and acknowledged alerts triggered by the old alert definition. When you change the alert severity, the existing alerts retain the old severity and the new alerts use the new severity. Changing the informational values of the alert definition such as name, description, or recommendation does not affect existing alerts.

Prerequisites

You must have the EDIT_ALERT_DEFINITION permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications > Alert Definitions**. The Alerts - Alert Definitions page is displayed.

2 Click the alert definition to edit, and under the **Actions** drop-down menu, click**Edit**.

The **Edit Alert Definition** wizard is displayed.

3 Edit the alert definition details and click **SAVE**.

Results

You have successfully edited an alert definition.

Note If you change the condition of your alert definition, all active alerts from the previous condition are automatically canceled.

Deleting an Alert Definition

This section lists the steps to delete an alert definition. When you delete an alert definition, all active and acknowledged alerts that are triggered from this alert definition are canceled.

Prerequisites

You must have the DELETE_ALERT_DEFINITION permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications > Alert Definitions**. The Alerts - Alert Definitions page is displayed.

2 To delete an alert definition, select the alert definition from the list and click the delete icon.

A confirmation dialog box is displayed.

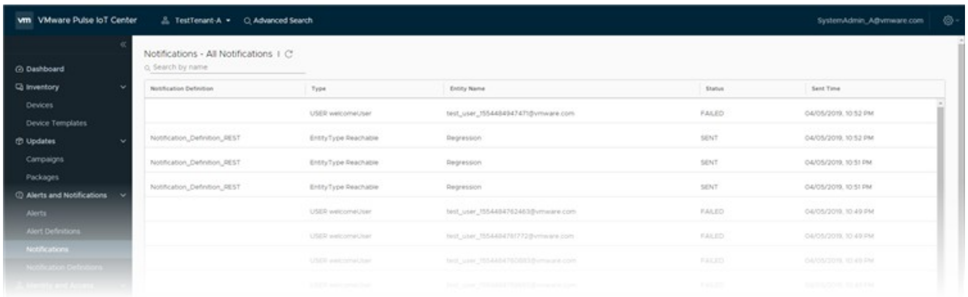
3 In the confirmation dialog box, confirm your action and click **DELETE**.

Results

You have successfully deleted an alert definition.

Notifications

Use the **Notifications** tab to view email and REST notifications.



Notifications - All Notifications

Notification Definition	Type	Entity Name	Status	Sent Time
	USER welcomeUser	test_user_1554484947471@vmware.com	FAILED	04/05/2019, 10:52 PM
Notification_Definition_REST	EntityType Reachable	Regression	SENT	04/05/2019, 10:52 PM
Notification_Definition_REST	EntityType Reachable	Regression	SENT	04/05/2019, 10:51 PM
Notification_Definition_REST	EntityType Reachable	Regression	SENT	04/05/2019, 10:51 PM
	USER welcomeUser	test_user_1554484762463@vmware.com	FAILED	04/05/2019, 10:49 PM
	USER welcomeUser	test_user_1554484761772@vmware.com	FAILED	04/05/2019, 10:49 PM
	USER welcomeUser	test_user_1554484760683@vmware.com	FAILED	04/05/2019, 10:49 PM
	USER welcomeUser	[illegible]	[illegible]	[illegible]

The Notifications feature enables you to receive timely notifications without logging in to the SmartHub INFER IoT Center, or without providing an integration point into the existing monitoring systems. The Notifications feature acts as a primary interface for all HTTP and email (SMTP) notifications from the SmartHub INFER IoT Center server. All the other SmartHub INFER IoT Center services communicate with the notifications service to send notifications to the external servers.

Viewing Notifications

This section lists the steps to view a notification.

Prerequisites

You must have the VIEW_NOTIFICATION_INSTANCE permission to perform this operation.

Procedure

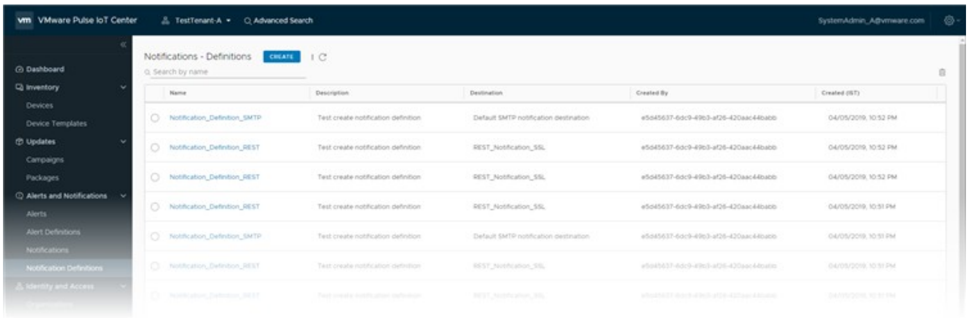
1 From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Notifications**. The Notifications - All Notifications page is displayed.

2 To view more information about a notification, click the notification.

Details about the notification are displayed.

Notification Definitions

Notification definitions contain details such as the type of notification to be sent, the sender's details, and the number of times to retry.



Name	Description	Destination	Created By	Created (IST)
○ Notification_Definition_SMTP	Test create notification definition	Default SMTP notification destination	e5d45637-6dc9-49b3-af26-420aac44babb	04/05/2019, 10:52 PM
○ Notification_Definition_REST	Test create notification definition	REST_Notification_SSL	e5d45637-6dc9-49b3-af26-420aac44babb	04/05/2019, 10:52 PM
○ Notification_Definition_REST	Test create notification definition	REST_Notification_SSL	e5d45637-6dc9-49b3-af26-420aac44babb	04/05/2019, 10:52 PM
○ Notification_Definition_REST	Test create notification definition	REST_Notification_SSL	e5d45637-6dc9-49b3-af26-420aac44babb	04/05/2019, 10:51 PM
○ Notification_Definition_SMTP	Test create notification definition	Default SMTP notification destination	e5d45637-6dc9-49b3-af26-420aac44babb	04/05/2019, 10:51 PM
○ Notification_Definition_REST	Test create notification definition	REST_Notification_SSL	e5d45637-6dc9-49b3-af26-420aac44babb	04/05/2019, 10:51 PM
○ Notification_Definition_REST	Test create notification definition	REST_Notification_SSL	e5d45637-6dc9-49b3-af26-420aac44babb	04/05/2019, 10:51 PM

Using Notification Definitions, you can define and send repeated notifications for a fixed number of times to the users until a corrective action is taken.

Creating a Notification Definition

This section lists the steps to create a notification definition from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the CREATE_NOTIFICATION_DEFINITION permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Notification Definitions**. The Notifications - Definitions page is displayed.
2. Click **CREATE**.

   The **Create Definition** wizard is displayed.
3. In the **Details** step, enter the following details:
   - **Name**: Notification definition name.
   - **Description**: Enter a brief description about the notification definition.

   Click **Next**.
4. In the **Settings** step, enter the following information:
   - **Type**: Select whether the notification type is **Rest Notification** or **Email Notification**.

     If you select **Email Notification**, enter the following information:
     - **Recipient Email Address**: Enter the email address of the recipient.
     - To add more recipients, click **+ Add Recipient**.
     - To add the sender name, sender email address, email subject, and to add a customized email template, click **Advanced Settings**. Click **Next** and go to Step 5.

     If you select **Rest Notification**, enter the following information:
     - Select **Secure Protocol Secure (TLS) is recommended**.
     - Enter the **Host URL** and **Path**.

  - In the **Certificate** text box, copy the certificate.
  - In the **Authentication Type** text box, select the type of authentication. Enter the **username** and **password.**
  - Click **Next**.
  - You can configure the retry interval if a notification fails to reach the endpoint or the server. With the retry interval configured, notifications are sent to the server or endpoint until the maximum retry count value is reached. To add a retry interval, select **Retry Schedule** under **Advanced Settings** and enter the **Retry Interval** time, **Request Timeout** time, and the **Max Retry Count** value. The default value for **Retry Interval** and **Request Timeout** is 5 seconds.
  - To add a header and a body template, click **Advanced Settings** .
    - **Header Name**: Enter the header name.
    - **Header Value**: Enter the header value.
    - To add more headers, click **+ Add New Header**.
    - **Body Template**: Use the default REST template to create custom REST notifications. You can insert new text boxes using the **Insert field** drop-down menu.
    - Click **Next**.
  - In the **Link** step, enter the following details to enable a linked notification:
    - **On Success**: Select the notification definition that you want to associate.
    - **On Failure**: Select the notification definition that you want to associate.
    - **On Completion**: Select the notification definition that you want to associate.
  - Click **Next** and in the **Review** step, click **Save**.

  Note The **Link** menu item is enabled only if you have selected the **Type** as **Rest Notification** in the previous step.

5 In the **Recurrence** step, select **Recurrence notification** and enter the following information to schedule a recurrent notification:

  - **Recurrence Interval**: Enter the days, hours, and minutes.
  - **Max Recurrence Count**: Enter the number of times the notification has or reoccur.

  Click **Next**.

  Note The **Recurrence** menu item is enabled only if you have selected the **Type** as Email Notification in the previous step.

6 In the **Review** step, review the information and click **SAVE**.

Results

You have successfully created a notification definition.

Editing a Notification Definition

This section lists the steps to edit a notification definition using the SmartHub INFER IoT Center UI.

Prerequisites

You must have the EDIT_NOTIFICATION_DEFINITION permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Notification Definitions**. The Notifications - Definitions page is displayed.
2. Click the notification definition to edit.

   The notification definition details are displayed.
3. From the **Actions** drop-down menu, click **Edit**.

   The **Edit Definition** wizard is displayed.
4. Update the notification definition details and click **SAVE**.

Results

You have successfully updated the notification definition details.

Deleting a Notification Definition

This section lists the steps to delete a notification definition using the SmartHub INFER IoT Center UI.

Prerequisites

You must have the DELETE_NOTIFICATION_DEFINITION permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Alerts and Notifications** > **Notification Definitions**. The Notifications - Definitions page is displayed.
2. Select the check box against the notification definition to delete.
3. Click the delete icon on the top-right side of the screen.

   An action confirmation message is displayed.
4. To confirm the action, click **DELETE**.

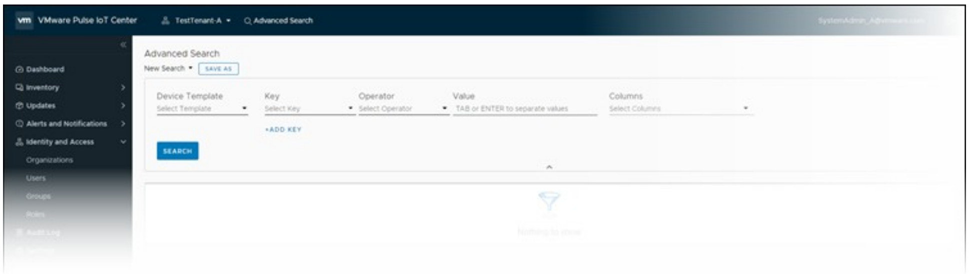
Results

You have successfully deleted the notification definition.

Using Advanced Search

11

With Advanced Search, you can search for a list of devices that meet multiple search parameters.



You can apply conditions such as the device template to search, and keys such as campaign name, campaign state, metric names, and device type. Select the values for each key you want to search with by using the **Equal To**, **Not Equal To**, **Greater Than**, **Greater Than Equal To**, **Less Than**, **Less Than Equal To**, and **Contains** operators. Advanced Search finds only those devices that meet all the conditions. You can select multiple values for a key condition, and the search finds the device whose key meets any of the values. If your keys include a metric name, you must set a time range for your search for that metric condition to be met. Or, you can enable the **Relative Timestamp** option to select a time range starting from the current date and time. Finally, select the columns that you want the search results to display, such as **Campaign Name**, **Campaign State**, **Device Type**, **Location**, and so on. **Device Id** always appears as the first column. For example, you can search for all Dell Edge 3000 gateways that exceeded 90% of the CPU utilization in the last 24 hours.

You can save your filter criteria as a filter definition. When performing an over-the-air campaign, you can use that saved filter definition to select the devices for the distribution list.

Note

- Ensure that the filter definition name is 35 characters or less in length.
- You cannot delete or modify the search criteria of a distribution list if they are associated with a campaign. You must delete the campaign first.

To perform advanced search operations, click **Advanced Search** on the top menu bar of the SmartHub INFER IoT Center UI.

Audit Log

12

The **Audit Log** tab provides a view into the actions performed by all SmartHub INFER IoT Center users and devices.

You must have the View Audit Logs permission to perform this operation.

You can filter the audit logs based on the following parameters:

- Entity Type
- Audit Type
- Device
- Date Range

To view more details about an entity and audit type, click a search result. For example, when you edit a device or a device template, you can view additional information about the changes made under the **Audit Details** section.

To export audit logs in the CSV format, perform the following steps:

1. From the Audit Log page, click **EXPORT**.
2. Select **All** to export all audit logs, or **Time Range** to select audit logs within a time range.
3. Click **EXPORT**.

Tasks

13

The **Tasks** tab provides a view of all the tasks run by the SmartHub INFER IoT Center. It lists the tasks with the overall progress, user name, organization, and time of creation of the task.

The Tasks tab displays basic information and entities of the device commands you send from the SmartHub INFER IoT Center.

Prerequisites

You must have **View Tasks** permissions to perform this operation.

Procedure

1. To verify the basic information of a task, go to **SmartHub INFER IoT Center** >

   Tasks. List of all the tasks is displayed.

2. From the **Search** menu item, search and click the task name.

   Basic information of the task such as Task Name, Task Id, Progress (Completed/Skipped/Pending/ Failed), and name of the command run is displayed at the device level.

3. To view the progress of the command on each device, select **Entities**.

   The list of devices on which the command is run is displayed. The status of the command such as completed, skipped, pending or failed is displayed against each device.

4. Click the **Device** name.

   The Inventory - Device page is displayed.

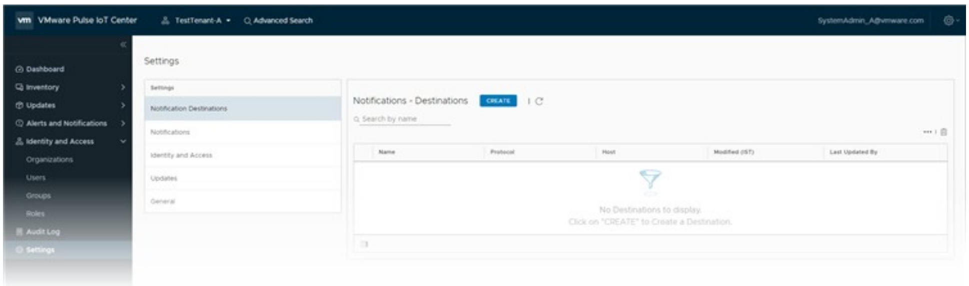
5. To verify the progress of the command, click **..** drop-down menu and select **Commands**.

   Command history with the status is displayed.

Settings

14

Use the **Settings** tab to configure settings such as notification retention period, domain name of the organization, approvals for OTA updates, and audit log retention period.



No Destinations to display.
Click on "CREATE" to Create a Destination.

An Organization Administrator can define organization settings at the root level so that the settings are applied to all the sub-organizations under the root organization. This way, the Organization Administrator need not apply settings individually to all sub-organizations under an organization. The sub-organizations can either use the applied settings or override them.

This chapter includes the following topics:

- Notification Settings
- Identity and Access Settings
- Update Settings
- General Settings
- System Notifications Settings
- Notification Destinations

Notification Settings

To update the notification retention period for your organization and all its sub-organizations, perform the following steps:

Prerequisites

You must be an Organization Administrator to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Click **Notifications**.

The notification retention period for your organization is displayed.

3 By default, the notification retention period is 30 days. Enter the new notification retention period and click **SAVE**.

Results

You have successfully updated the notification retention period for your organization.

Identity and Access Settings

You can configure one of the following authentication methods to authenticate access to the SmartHub INFER IoT Center console.

LDAP Authentication

To configure the Lightweight Directory Access Protocol (LDAP) settings for your organization, perform the following steps:

Prerequisites

Note

- SmartHub INFER IoT Center is integrated with LDAP.
- LDAP is supported on on-premise versions of SmartHub INFER IoT Center.

You must be an **Organization Administrator** to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Settings** and select **Identity and**

Access. The **Identity and Access Settings** page for your organization is displayed.

2 To use an external identity provider to manage authentication for your organization, select **Enable External Identity Provider**.

3 From the **IDP Type** drop-down menu, select **LDAP**.

4 By default, **Enable JIT user creation** is enabled. With this option enabled, SmartHub INFER IoT Center creates a shadow user if the user does not exist in any of the organizations. If you disable this option, the user cannot access SmartHub INFER IoT Center even though the user credentials are configured in the external IDP. All valid IDP users can log in to SmartHub INFER IoT Center when this option is enabled. To disable Just In Time (JIT) user creation, deselect **Enable JIT user creation**.

5. Under **LDAP Settings**, enter the following information:
   a. In the **Domain Name** text box, enter a valid domain name.
   b. Under **Server Details**, enter the following information:
      - **Directory Type** - Select a directory type:
        - **Active Directory**
        - **Open LDAP**
      - **Host** - Enter the host IP address.
      - **Port** - Enter the port number.
      - **Authentication Type** - Select one:
        - **SIMPLE**
        - **NONE**
        - **STRONG**
      - **Encryption Type** - Select one:
        - **SSL**
        - **NONE**
6. Under **User Details**, enter the following information:
   a. **Bind Username**: Enter the BIND user name.
   b. **Bind Password**: Enter the BIND password.
   c. **User Object Class**: Enter the object class associated with the user.
7. To ensure that the connection is successful, click **TEST CONNECTION**.
8. To save the settings, click **SAVE**.

Results

You have successfully configured the LDAP settings for your organization. You can now log in to the SmartHub INFER IoT Center UI with the external IDP credentials.

SAML Authentication

Security Assertion Markup Language (SAML) single sign-on (SSO) uses third-party authentication service providers to provide access to users. SAML SSO works by transferring the user's identity from the identity provider (IDP) to the authentication service provider, through the exchange of digitally signed XML metadata. To configure the SAML SSO settings for your organization, perform the following steps:

Prerequisites

You must be an **Organization Administrator** to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Settings** and select **Identity and Access**. The **Identity and Access Settings** page for your organization is displayed.
2. To use an external identity provider to manage authentication for your organization, select **Enable External Identity Provider**.
3. From the **IDP Type** drop-down menu, select **SAML**.
4. By default, **Enable JIT user creation** is enabled. With this option enabled, SmartHub INFER IoT Center creates a shadow user if the user does not exist in any of the organizations. If you disable this option, the user cannot access SmartHub INFER IoT Center even though the user credentials are configured in the external IDP. All valid IDP users can log in to SmartHub INFER IoT Center when this option is enabled. To disable Just In Time (JIT) user creation, deselect **Enable JIT user creation**.

   Note If you decide to update the JIT user creation settings at a later stage, you must reconfigure the SAML settings.

5. Under **SAML Settings**, perform the following steps:
   - **Step 1: SAML Certificates**

     Note If you do not provide a certificate, SmartHub INFER IoT Center generates a self-signed certificate. To skip this step, click **NEXT**.

     - **Signing Key**- Click **Choose File** and select the custom certificate from your local folder. This certificate is used as a signing key to access the SmartHub INFER IoT Center metadata.
     - **Signing Key Password** - If the certificate is password protected, enter the password to access it.
     - **Encryption Key** - Click **Choose File** and select the encryption key for the certificate.
     - **Encryption Key Password** - Enter the password for the encryption key.
   - **Step 2: Service Provider Metadata Download** - To download the metadata of SmartHub INFER IoT Center, click **DOWNLOAD**. Alternatively, you can copy the metadata content from the **SAML Service Provider Metadata** text box.
   - **Step 3: Identity Provider Setup** - Navigate to your IDP and configure SmartHub INFER IoT Center as a service provider.

Copy the downloaded service provider metadata to a text file and save it with the `.xml` extension. For example, `pulseSP_metadata.xml`. Use the saved service provider (SP) metadata to configure the service provider settings on the IDP. To authenticate the user, you must assign the user to the IDP. This authenticates the user to log in to SmartHub INFER IoT Center for the particular organization.

Note To set the SAML SSO authentication for your user on multiple suborganizations, you must register the service provider in the IDP for each of the suborganizations. Use the suborgnanization's SP metadata to register.

- **Step 4: SAML Setup**
  - **SAML Authentication URL** - The external IDP's authentication URL to which you post the request to.
  - **SAML Metadata XML** - The URL or the metadata of the external IDP. You can access the metadata by sending a GET request to the external IDP.
  - **Attribute Mapping** - Add the attribute keys for creating the user. **UserName**, **DisplayName**, and **Email** are mandatory keys. These keys must be mapped to the **UserName**, **DisplayName**, and **Email** keys in the IDP.

6 To save the changes, click **SAVE**.

Results

You have successfully configured the SAML SSO authentication settings in SmartHub INFER IoT Center.

Update Settings

To enable approvals for the OTA updates that are run in your sub-organizations, perform the following steps.

Prerequisites

You must be an Organization Administrator to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Click **Updates**.

The update settings information for your organization is displayed.

3 Select the Enable Approvals option to enable approvals for all OTA updates that your sub-organizations run.

4 Click **SAVE**.

Results

You have successfully enabled approvals for OTA updates.

General Settings

To update the audit log retention period for your organization and all its sub-organizations, perform the following steps:

Prerequisites

You must be an Organization Administrator to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Click **General**.

The audit log retention period for your organization is displayed.

3 By default, the audit log retention period is 30 days. Enter the new retention period and click **SAVE**.

Results

You have successfully updated the audit log retention period.

System Notifications Settings

You can send email notifications to user groups, users, or email addresses of your organization.

Perform the following steps to send system notification emails to users belonging to a particular group, listed users, or email addresses:

Prerequisites

You must be an **Organization Administrator** to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Click **System Notifications**.

The System Notifications page is displayed.

3 In the System Notifications page, add the following information:

- **Groups** - Click **Add Groups**, and select the group from the drop-down.
- **Users** - Click **Add User**, and select the user from drop-down.

- **Email**- In **Recipients**, add recipient email Ids.

4 Click **SAVE**.

Results

You have successfully created a system notification setting.

Notification Destinations

The **Notification Destinations** tab contains the common details of the external servers such as host name, port, certificate, and credentials where the notifications are delivered.

SmartHub INFER IoT Center supports HTTP and REST servers for this release. Other types of servers will be supported in the future releases.

Creating a Notification Destination

This section lists the steps to create a notification destination from the SmartHub INFER IoT Center UI.

Prerequisites

You must have the CREATE NOTIFICATION DESTINATION permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Click **Notification Destinations**.

The notification destinations for your organization is displayed.

3 Click **CREATE**.

The **Create Destination** wizard is displayed.

4 In the **Details** step, enter the following details:

- **Name** - Notification Destination name.
- **Description** - A brief description about the notification destination.

Click **NEXT**.

5 In the **Settings** step, enter the following information:

- Select **Secure Protocol** to send notifications using an encrypted connection. The recommended protocol is TLS.
- Enter the host URL and port information.
- If you have selected Secure Protocol, enter the certificate information in the **Certificate** field. The certificate must be in the Privacy Enhanced Mail (PEM) format.
- **Authentication Type**: Select whether the authentication type is **Basic** or **No Authentication**.

- Enter the **Username** and **Password** of the destination instance.

Click **Next**.

6 In the **Review** step, review the information and click **SAVE**.

Results

You have successfully created a notification destination.

Viewing a Notification Destination

This section lists the steps to view a notification destination using the SmartHub INFER IoT Center UI.

Prerequisites

You must have the VIEW_NOTIFICATION_DESTINATION permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Click **Notification Destinations**.

The notification destinations for your organization is displayed.

3 Under the Notification Destinations section, select the radio button against the notification destination to view.

4 Click the **Actions** drop-down menu and select **View**.

Details of the Notification Destination are displayed.

Editing a Notification Destination

This section lists the steps to edit a notification destination using the SmartHub INFER IoT Center UI.

Prerequisites

You must have the EDIT_NOTIFICATION_DESTINATION permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Select the radio button against the notification destination to edit.

3 Click **Notification Destinations**.

The notification destinations for your organization is displayed.

4 Click the **Actions** drop-down menu and select **Edit**.

The Edit Destination wizard is displayed.

5 Update the notification destination details and click **SAVE**.

Results

You have successfully updated the notification destination details.

Deleting a Notification Destination

This section lists the steps to delete a notification destination using the SmartHub INFER IoT Center UI.

Prerequisites

You must have the DELETE_NOTIFICATION_DESTINATION permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center UI, go to

Settings. The Settings page is displayed.

2 Click **Notification Destinations**.

The notification destinations for your organization is displayed.

3 Select the radio button against the notification destination to delete.

4 Click the **Actions** drop-down menu and click **Delete**.

An action confirmation message is displayed.

5 To confirm the action, click **DELETE**.

Results

You have successfully deleted the notification destination.

Managing BIOS Attributes 15

Use the Bios Management package for updating or querying BIOS parameters locally.

To download the BIOS Management .deb file, perform the following steps:

1 Log in to the SmartHub INFER IoT Center console.

2 Click the **Settings** icon at the top right corner and click **Downloads**. The Downloads window lists the items that can be downloaded from the SmartHub INFER IoT Center console.

3 Download the Bios Management - x64 .deb file from the list and save it to your local drive.

This chapter includes the following topics:

- Installing the BIOS Management Package in Linux
- Installing the BIOS Management Package on Windows
- Uninstalling the BIOS Management Package
- BIOS Management Command Line Interface

Installing the BIOS Management Package in Linux

This section lists the prerequisites and steps for installing the BIOS Management package in Linux.

Prerequisites

Install the Open Management Infrastructure (OMI) package on the target system. If the OMI package is not installed, the BIOS Management package installation displays an error message. The steps to install the Dell Command | Monitor package on an Ubuntu Desktop is listed here:

1 Download the Dell Command | Monitor package from the SmartHub INFER IoT Center to your target system.

2 The Dell Command | Monitor package contains three sub-packages:

   - omi-1.1.0.ssl_100.x64.deb - Run the following command to install the OMI package:

     ```
     dpkg –i omi-1.1.0.ssl_100.x64.deb
     ```

- srvadmin-hapi_9.1.0-1_amd64.deb - Run the following command to install the server administrator package:

  ```
  dpkg –i srvadmin-hapi_9.1.0-1_amd64.deb
  ```

- command-monitor_10.0.0-208.ubuntu16_amd64.deb - Run the following command to install the Dell Command | Monitor package:

  ```
  dpkg –i
  command-monitor_10.0.0-208.ubuntu16_amd64.deb
  ```

The required namespace (root-dcim-sysman) is installed on the system.

Note The BIOS Management package is developed specifically for Dell BIOS.

Procedure

1 To install the BIOS Management package, run the following command:

```
dpkg –i bios-mgmt-amd64-1.0.0.deb
```

The BIOS Management package is installed at /opt/ bios_mgmt_server.

2 To verify the installation, run the following command:

```
dpkg -l "vmware-bios-mgmt-server"
```

This command displays the installed version of the BIOS Management package.

3 Configure the export log file environment variable LOG_CFG_PATH. For example:

```
export LOG_CFG_PATH=/opt/vmware/bios_mgmt/cfg
```

4 Add the log filename with its path in the log configuration file Log4cxxConfig.cfg. For example:

```
log4j.appender.rootFileAppender.File=/opt/vmware/bios_mgmt/cfg/sampletest.log
```

5 Run the BIOS Manager CLI:

```
/opt/vmware/bios_mgmt/bin/BiosManager_CLI -h
```

Results

You have successfully installed the BIOS Management package.

Installing the BIOS Management Package on Windows

This section lists the steps for installing the BIOS Management package on Windows.

Procedure

1. Log in to the SmartHub INFER IoT Center console.
2. Go to **Settings** on the top right corner and click **Download**.
3. In **Bios Management**, select Win64 and click **Download**.

   The MSI file for the BIOS management setup downloads on your local.
4. Run the MSI file.
5. Follow the instructions in BIOS management setup wizard and complete the installation.

   You have successfully installed the BIOS Management package on windows.

Uninstalling the BIOS Management Package

Follow the instructions to uninstall the BIOS Management package.

To uninstall the BIOS Management package, run the following commands:

- **On Linux**:

  ```
  dpkg --purge vmware-bios-mgmt-server
  ```

- **On Windows**:

  a. Navigate to **Control Panel** > **Programs and Features** > **Uninstall a Program**.

  b. Search and click BIOS Management package for windows.

  c. Click **Uninstall** and **YES**.

     You have successfully uninstalled the BIOS management package.

BIOS Management Command Line Interface

This section lists the CLI for managing BIOS attributes. Use the BIOS Management CLI to update the BIOS parameters locally.

```
    $BiosManager_CLI   [options]
Options:
-h, --help                  Display this usage message
-e, --enumeration       List all Bios Attribute Name and Value
-s, --set                     Set Particular Bios Attribute
-f, --find                    Find Particular Bios Attribute using attribute name
-a, --attribute             Attribute Name you want to set
-v, --value                 Value for the attribute
-j, --json                    Json file for change multiple attributes
-o, --output                Result store as Json file
-p, --password           Bios password need in set attribute
```

Show All BIOS Attributes

To enumerate all BIOS attributes:

```
$BiosManager_CLI –e
```

Output

```
{
  "Attributes": {
    "AC Power Recovery Mode": "1",
    "Admin Setup Lockout": "1",
    "Allow BIOS Downgrade": "2",
    "Auto On": "1",
    "Auto On Hour": "1",
    "Auto On Minute": "1",
              …
              ….
   }
}
```

To enumerate all BIOS attributes and save in a JSON file:

```
$BiosManager_CLI –e –o <Json file name>
```

Example

```
BiosManager_CLI –e –o enumeration.json
```

Set BIOS Attributes

To change a single BIOS attribute:

```
$BiosManager_CLI –s –a <attribute name> -v <attribute value> -p <bios
Password>
```

Example

```
$BiosManager_CLI –s –a “Num Lock” -v “1” -p “adminv”
```

Output

```
{
  "ReturnValue": 0,
  "SetResult": 0,
  "AttributeName": "Num Lock",
  "AttributeValue": "1"
}
```

To change multiple attributes you require a JSON file as input:

```
$BiosManager_CLI –s –j <json file name>
```

Example

```
$BiosManager_CLI –s –j Attribute.json
```

Note

The JSON file format must be:

```
{
   "Num Lock":"1",
   "Wake-On-LAN":"1"
}
```

To change the attribute and save as JSON:

```
$BiosManager_CLI –s –a <attribute name> -v <attribute value> -p <Bios
Password> -o <json file name>
$BiosManager_CLI –s –j <input json file name> -o <output json file name>
```

Example

```
$BiosManager_CLI –s –a “Num Lock” -v “1” -p “Pass”-o “out.json”
$BiosManager_CLI –s –j input.json-o output.json
```

Find BIOS Attributes

To find a specific BIOS attribute:

```
$BiosManager_CLI –f –a <attribute name>
```

Example

```
$BiosManager_CLI –f –a “Num Lock”
```

Output

```
{
   "AttributeName": "Num Lock",
   "CurrentValue": "1",
   "PossibleValues": [
      "1",
      "2"
   ],
   "possibleValuesDescription": [
      "Disable",
      "Enable"
   ]
}
```

To find an attribute and store as JSON:

```
$BiosManager_CLI –f –a <attribute name> -o <Json file name>
```

Example

```
$BiosManager_CLI –f –a “Num Lock” -o output.json
```

Error Messages

The following table lists the `SetResult` error messages for these operations:

Description	Error Code (SetResult value)
Success	0
Invalid Attribute Name	1
Invalid Attribute Value	4294967295
Read Only or Out of Range Value	1
Authentication Error	2

Container Management 16

The Container Management feature allows you to run your applications in containers, within your gateways.

In the SmartHub INFER IoT Center console, the containers are displayed as virtual Things. You can perform operations such as create, start, stop, and remove, on these containers. You can even monitor their status and view their metrics. Using a container platform tool such as Docker, write an application for specific tasks that you want to perform on your gateways, and archive it as a container image. You can then use Container Management to run the container image without interrupting other applications that are already running on the gateway. Container Management can also pull images from the remote Docker Trusted Registry.

Using Container Management, you can perform the following operations:

- `pull` - Pull an image and make it available on the gateway.
- `create` - Create a container and specify the parameters.
- `start` - Start a container by name.
- `run` - Pull, create, and start a container.
- `stop` - Stop a container by name.
- `rm` - Remove a container by name.
- `rmi` - Remove a container image by name.

Note Before you proceed, ensure that Docker service is enabled and running on your gateway.

To work with Container Management, use the following workflow:

1 Create a container Thing template with Client Commands. For more information, see Create a Container Thing Template.

2 Create a Docker gateway template with Client Commands. For more information, see Create a Docker Gateway Template.

3 Register and enroll your Docker gateway to the SmartHub INFER IoT Center server. For more information, see Onboard Your Docker Gateway.

4 Install Container Management. For more information, see Install Container Management.

5 Perform Container Management operations. For more information, see Container Management Sample Use Case.

This chapter includes the following topics:

- Create a Container Thing Template
- Create a Docker Gateway Template
- Onboard Your Docker Gateway
- Install Container Management
- Container Management Sample Use Case

Create a Container Thing Template

Create a container Thing Template with the following system properties and metrics, using the SmartHub INFER IoT Center console.

Prerequisites

You must have the CREATE DEVICE TEMPLATE permission to perform this operation.

Procedure

1 From the SmartHub INFER IoT Center console, go to **Inventory** > **Device Templates**.

2 In the Inventory - Device Templates screen, click **CREATE**. The **Create Device Template** wizard is displayed.

3 In the **Details** step, enter the template name and select the device type as **Thing**. You can add an optional image. Click **NEXT**.

4 In the **Properties** step, add the following system properties and click **NEXT**:

   - **time-since-creation**: The time that has passed since the Thing is created.
   - **up-time**: The duration of time the Thing device is up.
   - **name**: The name of the Thing device.
   - **command**: The commands associated with the Thing device.
   - **id**: The Thing device id.
   - **image**: An image associated with the Thing device.
   - **status**: The status of the Thing device.

   Note The system properties are case-sensitive.

5 In the **Metrics** step, add the metrics that you want to collect for your container. For example, if you are registering an NGINX server to SmartHub INFER IoT Center, you can add the network-related metrics to be collected. Click **NEXT**.

6 In the **Connected Devices Templates** step, click **NEXT**.

7 In the **Commands** step, add the following commands:

- `start` - Start the container.
- `stop` - Stop the container.

Note The Client Identifier must match the container client ID when a SmartHub INFER IoTC Agent connection is established. The template client ID and the container client IDs are configurable, but they must be the same. The client ID can be found at the following location in your gateway:

```
/opt/vmware/iotc-client/conf/plugin_conf/container-management.cfg
```

For example:

```
root@photon-machine [ ~ ]# cat /opt/vmware/iotc-client/conf/plugin_conf/container-management.cfg
{
    "interface":{
        "type":"c",
        "name":"container-management",
        "init":"InitializePlugin"
    },
    "clientId": "com.vmware.containermgmt.client",
    "commandInterval":300,
    "propertyInterval":600,
    "metricInterval":300,
    "properties":[],
    "metrics" : [
      {"name": "num-containers", "type": "integer"},
      {"name": "num-running-containers", "type": "integer"},
      {"name": "num-images", "type": "integer"},
      {"name": "network-bytes-in", "type": "integer"},
      {"name": "network-bytes-out", "type": "integer"},
      {"name": "cpu", "type": "double"},
      {"name": "memory", "type": "double"}
    ]
}
```

You can configure the default values in the `container-management.cfg` file. Ensure that you restart the Container Management service for the changes to take effect.

8 In the **Review** step, review the information that you have entered and click **SAVE**.

Results

You have successfully created a container Thing template.

What to do next

Create a Docker gateway template.

Create a DockerGateway Template

Create a Docker gateway template with container-specific system properties and metrics, and associate the container Thing template to it.

Prerequisites

- The gateway template must have commands relevant to the container runtime actions such as, pull, create, rm, and rmi.
- The Client ID must match the container client ID when a SmartHub INFER IoTC Agent connection is established. The template client ID and the container client IDs are configurable, but they must be the same.
- You must have the CREATE DEVICE TEMPLATE permission to perform this operation.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** > **Device**

   Templates. The Inventory - Device Templates screen is displayed.
2. Click **CREATE**.

   The Create Device Template wizard is displayed.
3. In the **Details** step, enter the template name and select the device type as **Gateway**. To associate an image with your gateway, click **UPLOAD IMAGE**. Click **Next**.
4. In the **Properties** step, add the following system properties and click **NEXT**:
   - **downloaded-images**: The container images that are downloaded to the gateway.
   - **running-containers**: The containers that are running on the gateway.
   - **all-containers**: A list of all containers on the gateway.

   Note The system properties are case-sensitive.
5. In the **Metrics** step, add the following Docker-related metrics and click **NEXT**.
   - **num-containers**: the total number of containers.
   - **num-running-containers**: The number of containers that are running on the gateway.
   - **num-images**: The number of images on the gateway.
6. In the **Connected Device Templates** step, add the container Thing template that you created. You can add multiple container Thing templates here. Click **NEXT**.

7 In the **Commands** step, enter the following Docker commands for your gateway.

Note

- Ensure that you select **Client Command** as the command type.
- The Client Identifier must match the container client ID when a SmartHub INFER IoTC Agent connection is established. The template client ID and the container client IDs are configurable, but they must be the same. The client ID can be found at the following location in your gateway:

  ```
  /opt/vmware/iotc-client/conf/plugin_conf/container-management.cfg
  ```

  For example:

  ```
  root@photon-machine [~ ]# cat /opt/vmware/iotc-client/conf/plugin_conf/container-
  management.cfg
  {
      "interface":{
          "type":"c",
          "name":"container-management",
          "init":"InitializePlugin"
      },
      "clientId": "com.vmware.containermgmt.client",
      "commandInterval":300,
      "propertyInterval":600,
      "metricInterval":300,
      "properties":[],
      "metrics" : [
        {"name": "num-containers", "type": "integer"},
        {"name": "num-running-containers", "type": "integer"},
        {"name": "num-images", "type": "integer"},
        {"name": "network-bytes-in", "type": "integer"},
        {"name": "network-bytes-out", "type": "integer"},
        {"name": "cpu", "type": "double"},
        {"name": "memory", "type": "double"}
      ]
  }
  ```

- You can configure the default values in the container-management.cfg file. Ensure that you restart the Container Management service for the changes to take effect.

Command Name	Argument Name	Value
pull: The command to pull an image.	image_name	The image name. This value can be changed when you send the command.
create: The command to enroll the container to SmartHub INFER IoT Center.	■ pulse_template ■ container_name ■ image_name ■ options	■ The container template name. ■ The container name. ■ The image used for creating the container. ■ Any Docker related options.

Command Name	Argument Name	Value
run: The command to pull, create, and start a container.	■ pulse_template ■ container_name ■ image_name ■ options	■ The container template name. ■ The container name. ■ The image used for creating the container. ■ Any Docker related options.
rm: The command to remove a container from SmartHub INFER IoT Center.	container_name	The name of the container to remove.
rmi: The command to remove a container image from SmartHub INFER IoT Center.	image_name	The image name to remove.

8 In the **Enrollment Provider** step, select an appropriate method to authenticate yourgateway enrollment and click **NEXT**.

9 In the Settings step, click **NEXT**.

10 In the **Review** step, review the information that you have entered and click **SAVE**.

Results

You have successfully created a Docker gateway template and associated a container Thing template to it.

What to do next

Onboard your Docker gateway.

Onboard Your Docker Gateway

Register your gateway using the Docker gateway template that you have created. After you register, enroll your Docker gateway to SmartHub INFER IoT Center.

For information about onboarding your gateway to SmartHub INFER IoT Center, see Chapter 4 Onboarding a Gateway to SmartHub INFER IoT Center.

Note If you re-enroll a Docker Gateway, you must restart the Container Management service by running the following command:

```
systemctl restart container-management.service
```

Prerequisites

Ensure that Docker service is enabled and running on your gateway.

Install Container Management

The Container Management file is available as a downloadable bundle from the SmartHub INFER IoT Center console. This section lists the steps to download and install Container Management.

Prerequisites

- Ensure that your Docker gateway is enrolled.
- Ensure that the Docker service is enabled and running on your gateway.

Procedure

1 Log in to the SmartHub INFER IoT Center console UI.

2 From the top right of the home page, click the settings icon and click **Downloads**.

3 From the Downloads window, scroll down to the **Container Management** section and click **Download** against the container management bundle.

4 Using a linux machine, download the Container Management bundle to your local disk.

5 Create a campaign:

a Create a package specification YML file. The following example illustrates a sample package specification YML file:

package-spec.yml

```
root@photon2gw [ ~/code/iotc-clientd/deployment/campaign ]# cat package-spec.yml
package:
  name: container-management-deploy
  description: container management deploy for photon
  version: 1.0.0
  os: Linux
  architecture: x86_64
  manifest:
    headlessExecution: true
    lifecycle:
      - phase: execute
        action: data/campaign_install.sh
  attachments:
    - path: data/campaign_install.sh
    - path: data/iotc-client-x86_64-2.0.0.13.tar.gz
    - path: data/container-management.service
```

b Create an installation script. The following example illustrates a sample installation script using a Photon gateway:

campaign_install.sh

```
root@photon2gw [ ~/code/iotc-clientd/deployment/campaign ]# cat data/campaign_install.sh
#!/usr/bin/env bash
#example campaign on a photon gateway

logdir=/var/log/vmware
mkdir -p $logdir

logfile=$logdir/container_management_campaign_log.txt

echo $(date "+%Y-%m-%d %H:%M:%S") Execution script started >> $logfile

#ensure docker installed
tdnf install docker -qy
systemctl daemon-reload
systemctl restart docker

#clear staging dir
rm -rf /tmp/iotc-client
tar -xf $DATADIR/data/iotc-client-x86_64-0.0.13.tar.gz -C /tmp

#execute install
/tmp/iotc-client/install.sh
```

```
#remove staging
rm -rf /tmp/iotc-client

echo $(date "+%Y-%m-%d %H:%M:%S") Execution script finished >> $logfile
```

c Create a campaign package. The following example illustrates a sample campaign package:

```
root@photon2gw [ ~/campaign ]# ./package-cli package create ./package-spec.yml

root@photon2gw [ ~/campaign ]# ls -al *.iotcp
-rw-r----- 1 root root 68891 Jun 18 16:58 demo-1.0.0.iotcp
```

d Upload the campaign package to the SmartHub INFER IoT Server.

```
root@photon2gw [ ~/campaign ]# ./package-cli upload package ./demo-1.0.0.iotcp <server
address>
```

6 Run a campaign on your gateways using the uploaded package.

7 Verify that the Container Management package is running on your gateways.

Results

When Container Management starts, it sends the container environment properties and metrics to the gateways. The **num-images**, **num-containers**, and **num-running-containers** metrics are displayed for each gateway.

Container Management Sample Use Case

In this example, we pull container images to a Docker Gateway and create containers using the container images. We also perform the start, stop, and remove operations on the containers and container images, using the SmartHub INFER IoT Center console.

Prerequisites

- Enable and run the Docker service in your gateway.
- Install Container Management in your gateway using **Campaigns**.
- Create a Docker gateway template with system properties, metrics, and add the Docker commands using the Client Commands feature. Use this template to enroll your gateway to the SmartHub INFER IoT Center server.
- Create Thing templates (container images) with system properties, metrics, and `start` and `stop` client commands.

Note After sending a command, you must wait for its status to change from **Pending** to the **Executed** state before sending another command. Click the refresh button for the status to change.

Procedure

1 From the SmartHub INFER IoT Center console, enroll the Docker gateway to the SmartHub INFER IoT Center server.

2 After the Docker gateway is enrolled, click the **Properties** tab and verify that the **downloaded-images** and **running-containers** properties do not have any values.

3 Click the **Metrics** tab and verify that the **num-images**, **num-containers**, and **num-running-containers** metric values are 0.

4 To pull the metrics and properties information of the container images, perform the following steps:

   a Select **Commands** from the **…** drop-down menu and click **SEND COMMAND**.

   b In the **Send Command** window, select **pull** from the **Select Command** drop-down menu and enter the name of the container image under **Arguments**.

   c Click **SEND COMMAND**.

   The **Command History** section displays the pull command as **Executed**. Click the **Properties** tab of the Docker gateway. The container images are displayed under **System Properties**. The **Metrics** tab displays the number of images as 1.

5 To create containers using the container images, perform the following steps:

   a From the Docker gateway, select **Commands** from the **…** drop-down menu and click **SEND COMMAND**.

   b In the **Send Command** window, select **create** from the **Select Command** drop-down menu.

   c Under **pulse_template**, type the container Thing template name that you created.

   d Under **container_name**, enter a name for the container.

   e Under **image_name**, enter the container image name that you have pulled.

   f Click **SEND COMMAND**.

   The **Command History** section displays the create command as **Executed**. Click the **Properties** tab of the Docker gateway. The containers are enrolled and are displayed under **All Containers**. Select any of the containers from the Devices - All Devices page and click **Properties**. The status of the container is **Created**. Select the Docker gateway and click Metrics. The **num-images**, and **num-containers**, metric values are 1. The value of **num-running-containers** is 0.

6 To start the containers, perform the following steps:

   a From the Devices - All Devices page, select a container, go to **…** > **Commands**, and click **SEND COMMAND**.

   b In the **Send Command** window, select **start** from the **Select Command** drop-down menu and click **SEND COMMAND**.

   The **Command History** section displays the start command as **Executed**. Click the **Properties** tab of the Docker gateway. The containers are displayed under **running-containers**. Select the **Metrics** tab. The value of **num-running-containers** is 1.

7. To combine the pull, create, and start operations, use the `run` command. This command pulls, creates, and starts the containers using the container images. Perform the following steps:

   a. From the Docker gateway, select **Commands** from the **…** drop-down menu and click **SEND COMMAND**.

   b. In the **Send Command** window, select **run** from the **Select Command** drop-down menu.

   c. Under **pulse_template**, type the container Thing template name that you created.

   d. Under **container_name**, enter a name for the container.

   e. Under **image_name**, enter the container image name that you have pulled.

   f. Click **SEND COMMAND**.

   The **Command History** section displays the `create` command as **Executed**. Click the **Properties** tab of the Docker gateway. The containers are enrolled and are displayed under **All Containers**. Select any of the containers from the Devices - All Devices page and click **Properties**. The status of the container is **Created**. Select the Docker gateway and click **Metrics**. The value of the **num-images**, **num-containers**, and **num-running-containers** metrics are 2.

8. To stop the containers, perform the following steps:

   a. From the Devices - All Devices page, select a container and go to **…** > **Commands**, and click **SEND COMMAND**.

   b. In the **Send Command** window, select **stop** from the **Select Command** drop-down menu and click **SEND COMMAND**.

   The **Command History** section displays the `stop` command as **Executed**. Click the **Metrics** tab of the Docker gateway. The value of the **num-running-containers** metric is 1.

9. To remove the containers, perform the following steps:

   a. From the Docker gateway, select **Commands** from the **…** drop-down menu.

   b. In the **Send Command** window, select **rm** from the **Select Command** drop-down menu.

   c. Under **Arguments**, enter the container name to remove and click **SEND COMMAND**.

   The **Command History** section displays the remove command as **Executed**. The **Enrollment State** of the containers are **Unenrolled**. Click the **Metrics** tab of the Docker gateway. The value of the **num-all-containers** metric is 1.

10. To remove the container images, perform the following steps:

    a. From the Docker gateway, select **Commands** from the **…** drop-down menu.

    b. In the **Send Command** window, select **rmi** from the **Select Command** drop-down menu.

    c. Under **Arguments**, enter the container image name to remove and click **SEND COMMAND**.

    The **Command History** section displays the remove image command as **Executed.** Click the **Metrics** tab of the Docker gateway. The value of the **num-images** metric is 1.

TPM-Based Attestation

17

TPM-based attestation is a process to detect gateway tampering for file systems.

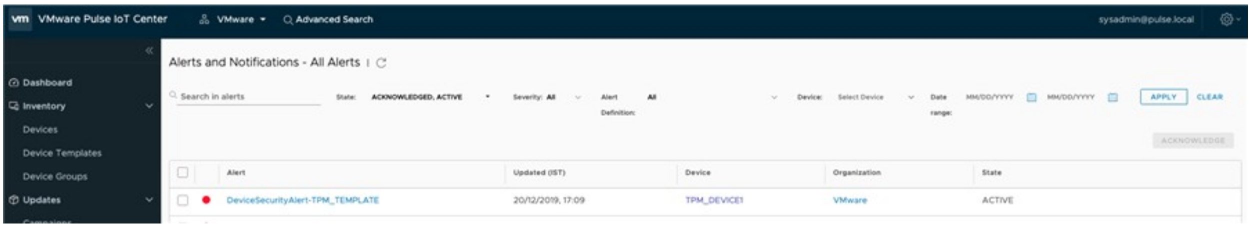
With the TPM-based attestation configured, SmartHub INFER IoT Center triggers an alert whenever a gateway is tampered.

TPM-based attestation are of two types:

- Boot Attestation
- Runtime Attestation

What is Boot Attestation

Boot attestation is a secure mechanism to verify the integrity of an IoT gateway during boot time. Boot attestation enables the detection of gateway file tampering every time the gateway boots. When a tampering is detected, SmartHub INFER IoT Center raises an alert.

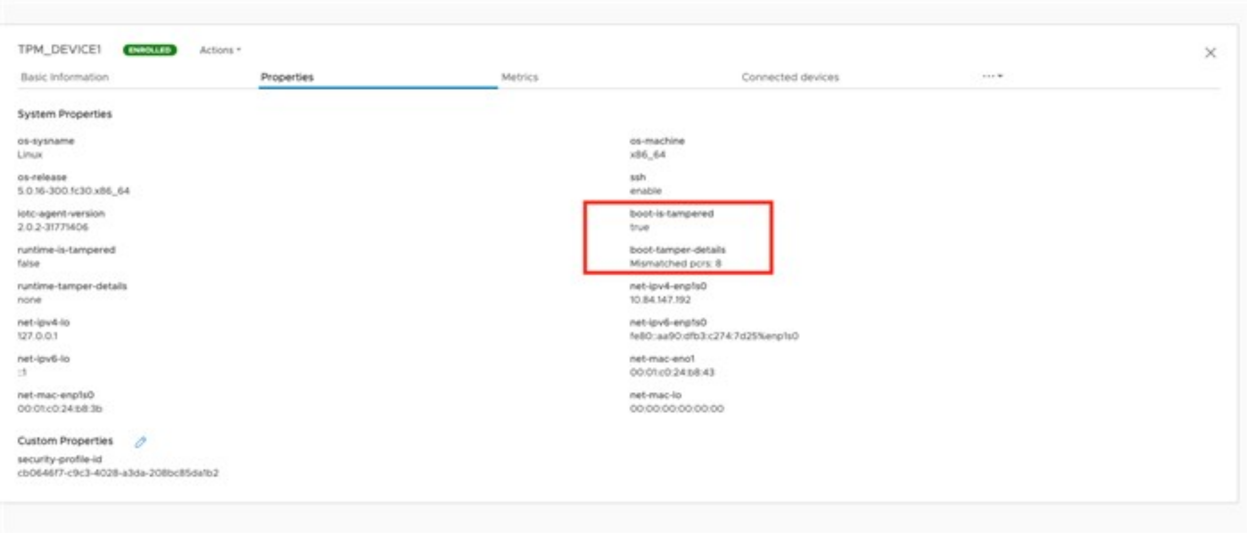


Alerts and Notifications - All Alerts

State: ACKNOWLEDGED, ACTIVE
Severity: All
Alert Definition: All
Device: Select Device
Date range: MM/DD/YYYY MM/DD/YYYY

		Alert	Updated (IST)	Device	Organization	State
□	●	DeviceSecurityAlert-TPM_TEMPLATE	20/12/2019, 17:09	TPM_DEVICE1	VMware	ACTIVE

The cause of failure is updated in the gateway properties.



System Properties

os-sysname
Linux

os-machine
x86_64

os-release
5.0.16-300.fc30.x86_64

ssh
enable

iotc-agent-version
2.0.2-31771406

boot-is-tampered
true

runtime-is-tampered
false

boot-tamper-details
Mismatched pcrs: 8

runtime-tamper-details
none

net-ipv4-enp1s0
10.84.147.192

net-ipv4-lo
127.0.0.1

net-ipv6-enp1s0
fe80::aa90:dfb3:c274:7d25%enp1s0

net-ipv6-lo
::1

net-mac-eno1
00:01:c0:24:b8:43

net-mac-enp1s0
00:01:c0:24:b8:3b

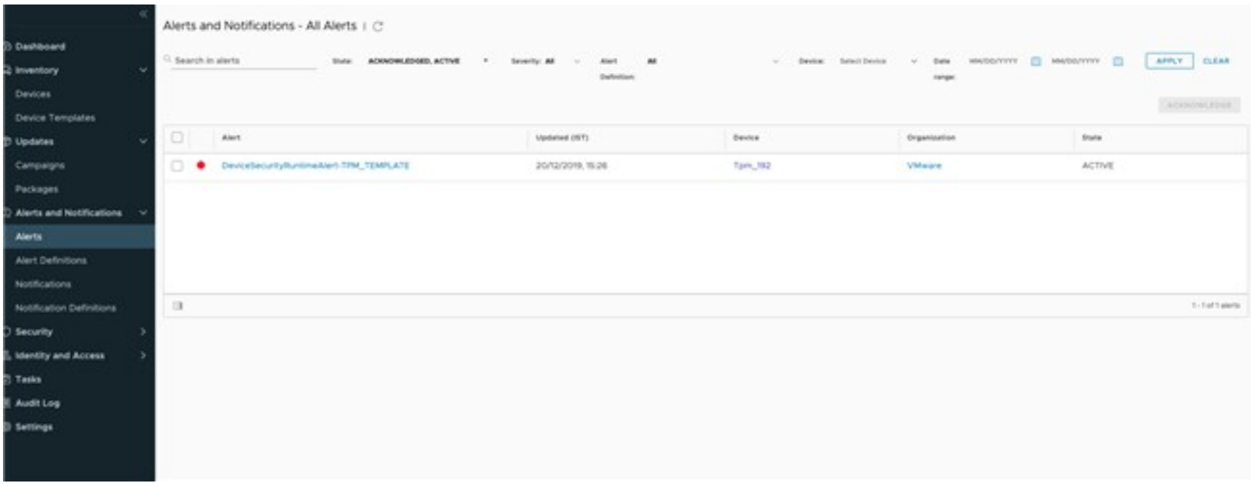
net-mac-lo
00:00:00:00:00:00

Custom Properties

security-profile-id
cb0646f7-c9c3-4028-a3da-208bc85da1b2

What is Runtime Attestation

Runtime attestation is a secure mechanism to verify the integrity of an IoT gateway during run time. The runtime attestation check occurs every 3600 seconds. When a tampering is detected, SmartHub INFER IoT Center raises an alert.



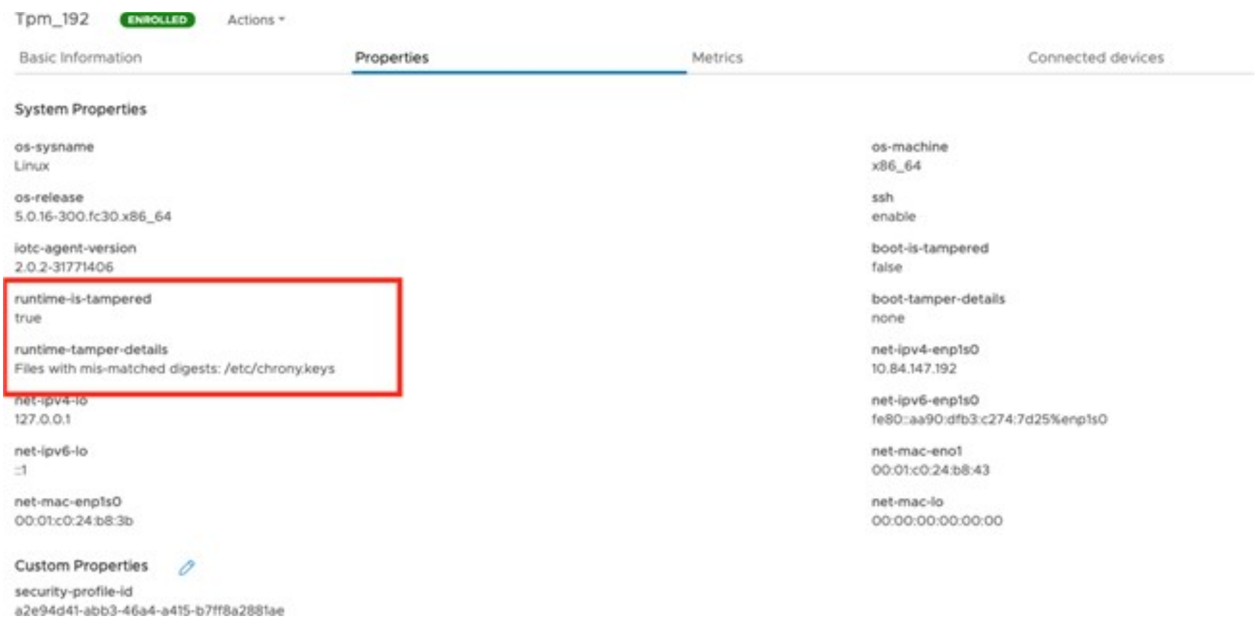
Alerts and Notifications - All Alerts

State: ACKNOWLEDGED, ACTIVE Severity: All Alert Definition: All Device: Select Device Date range: MM/DD/YYYY MM/DD/YYYY

□		Alert	Updated (IST)	Device	Organization	State
□	●	DeviceSecurityRuntimeAlert-TPM_TEMPLATE	20/12/2019, 15:26	Tpm_192	VMware	ACTIVE

1 - 1 of 1 alerts

The cause of failure is updated in the device properties.



System Properties

os-sysname
Linux

os-release
5.0.16-300.fc30.x86_64

iotc-agent-version
2.0.2-31771406

runtime-is-tampered
true

runtime-tamper-details
Files with mis-matched digests: /etc/chrony.keys

net-ipv4-lo
127.0.0.1

net-ipv6-lo
::1

net-mac-enp1s0
00:01:c0:24:b8:3b

os-machine
x86_64

ssh
enable

boot-is-tampered
false

boot-tamper-details
none

net-ipv4-enp1s0
10.84.147.192

net-ipv6-enp1s0
fe80::aa90:dfb3:c274:7d25%enp1s0

net-mac-eno1
00:01:c0:24:b8:43

net-mac-lo
00:00:00:00:00:00

Custom Properties

security-profile-id
a2e94d41-abb3-46a4-a415-b7ff8a2881ae

Note Runtime attestation is supported on Fedora IoT operating systems running on CompuLab IoT gateways.

To configure the runtime attestation, you must enable Integrity Measurement Architecture (IMA) on your gateway.

What Is IMA

Integrity Measurement Architecture (IMA) is an open source trusted computing component. IMA, when anchored in a hardware Trusted Platform Module (TPM), maintains a runtime measurement list with an aggregate integrity value of the list. This ensures that the measurement cannot be tampered without it being detected. Hence, on a trusted-boot system, the IMA can be used to attest to the system's runtime integrity.

This chapter includes the following topics:

- Preparing Your Gateway for Boot Attestation
- Preparing Your Gateway for Runtime Attestation
- Create a Boot Attestation Profile
- Create a Runtime Attestation Profile
- Associate the Attestation Profile With the TPM-Based Template
- Applying a Security Profile on Multiple Gateway Devices Using Campaigns

Preparing Your Gateway for Boot Attestation

To prepare your gateway for boot attestation, you must generate a `fingerprint.json` file.

Procedure

1 To generate a fingerprint, run the following command:

```
/opt/vmware/iotc-agent/bin/fingerprint dev > fp.json
```

A fingerprint file fp.json is generated.

2 Verify the fingerprint file:

```
cat fp.json
```

What to do next

Using the fp.json file, you can now create a boot attestation profile from the SmartHub INFER IoT Center console.

Preparing Your Gateway for Runtime Attestation

For SmartHub INFER IoT Center to detect tampering, you must configure the following IMA settings on your gateway.

Configure the Kernel

Append the Kernel command line with the following flag:

```
rootflags=i_version ima_policy=tcb ima_hash=sha256
```

For example, on a Fedora Workstation:

1 Edit /etc/default/grub with:

```
GRUB_CMDLINE_LINUX="rootflags=i_version ima_policy=tcb ima_policy=secure_boot ima_hash=sha256"
```

2 sudo grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg.

Note These steps might not work with a Fedora IoT image. Perform the following steps if the preceding steps do not work:

1 Open the /boot/loader/entries/ostree-1-fedora-iot.conf file and add the following flag to the line that starts with options:

```
"rootflags=i_version ima_policy=tcb ima_hash=sha256"
```

2 Restart the gateway for the settings to take effect.

Modify the /etc/fstab File

Add the following line to the fstab file:

```
UUID=d4bbe97d-a719-43af-a89a-19a9455cec5b /                    ext4    noatime,iversion        1 1
```

Obtain the Golden Values for Runtime Attestation

For the runtime attestation to work, you must record the current good state of your gateway. For this, you must run the ima_snapshot tool on the gateway.

1 To generate ima-policy, run the following script:

```
/opt/vmware/iotc-agent/script/install-ima-policy.sh
```

The ima-policy is generated and is placed in /etc/ima/ima-policy.

2 To generate ima-snapshot file, run the following script:

```
/opt/vmware/iotc-agent/bin/ima_snapshot -o <<ima.json path>>
```

Note

- Add all the paths to be excluded in a file and provide the file path to -e option in the final ima.json file.
- Edit the ima.json file and delete all the /usr/lib and /usr/lib64 file paths.

3 Verify the file by running the following command:

```
cat ima.json
```

Note The ima.json file must contain all the hashes.

4 Generate a fingerprint file. Run the following command:

```
/opt/vmware/iotc-agent/bin/fingerprint dev > fp.json
```

5 Verify the fingerprint file:

```
cat fp.json
```

Using the ima.json and fp.json files, you can now create a runtime attestation profile from the SmartHub INFER IoT Center console.

Create a Boot Attestation Profile

Use the fp.json file to create a boot attestation profile in the SmartHub INFER IoT Center console.

Prerequisites

You must have created a TPM-based device template. The TPM-based template contains the following system properties:

- **runtime-tamper-details**
- **boot-tamper-details**
- **runtime-is-tampered**

- **boot-is-tampered**

The template also contains the following custom property:

- **security-profile-id**

Procedure

1 From the SmartHub INFER IoT Center console, go to **Security** > **Profiles**.

2 Click **ADD PROFILE**.

The Add Profile wizard is displayed.

3 In the **Details** step:

a Enter a profile name.

b In the **Device Template** drop-down menu, select the TPM-based device template that you have created.

c Under **Notes**, enter information about theprofile.

d Click **NEXT**.

4 In the **Boot Attestation** step, click **UPLOAD** and upload the `fp.json` file. Click **NEXT**.

5 In the **Runtime Attestation** step, click **NEXT**.

6 In the **Review** step, review the updates and click **SAVE**.

Results

You have successfully created a boot attestation profile. You can view your profile under **Security** > **Profiles**. An alert definition is created for this profile. The alert definition is used for generating alerts when your gateway is tampered. To view the alert definition, go to **Alerts and Notifications** > **Alert Definitions**.

What to do next

Associate this profile with the TPM-based template.

Create a Runtime Attestation Profile

Use the `ima.json` and `fp.json` files to create a runtime attestation profile in the SmartHub INFER IoT Center console.

Prerequisites

You must have created a TPM-based device template. The TPM-based template contains the following system properties:

- **runtime-tamper-details**
- **boot-tamper-details**
- **runtime-is-tampered**

- **boot-is-tampered**

The template also contains the following custom property:

- **security-profile-id**

Procedure

1. From the SmartHub INFER IoT Center console, go to **Security** > **Profiles**.
2. Click **ADD PROFILE**.

   The Add Profile wizard is displayed.
3. In the **Details** step:
   a. Enter a profile name.
   b. In the **Device Template** drop-down menu, select the TPM-based device template that you have created.
   c. Under **Notes**, enter information about theprofile.
   d. Click **NEXT**.
4. In the **Boot Attestation** step, click **UPLOAD** and upload the `fp.json` file. Click **NEXT**.
5. In the **Runtime Attestation** step, click **UPLOAD** and upload the `ima.json` file. Click **NEXT**.

   Note The maximum size allowed for uploading is 20 MB.
6. In the **Review** step, review the updates and click **SAVE**.

Results

You have successfully created a runtime attestation profile. You can view your profile under **Security** > **Profiles**. An alert definition is created for this profile. The alert definition is used for generating alerts when your gateway is tampered. To view the alert definition, go to **Alerts and Notifications** > **Alert Definitions**.

What to do next

Associate this profile with the TPM-based template.

Associate the Attestation Profile With the TPM-Based Template

After creating an attestation profile, you must associate it with the TPM-based template. This ensures that the gateways you enroll using the TPM-based template are tamper-detectable.

Prerequisites

You must have a valid attestation profile.

Procedure

1. From the SmartHub INFER IoT Center console, go to **Security** > **Profiles**.
2. Copy the Profile ID of the attestation profile that you have created.
3. Go to **Device Templates** and click the TPM-based template that you have created.
4. Scroll down to the **Custom Properties** section and click the edit icon.
5. In the **Edit Custom Property** window, click the edit icon against the security-profile-id.
6. Paste the Profile ID under the **Default Value** text box. Click **DONE**.
7. Click **SAVE** to save the changes.
8. Next, configure the TPM attestation level in your gateway. Run the following command to open the `iotc.cfg` file:

   ```
   vi /opt/vmware/iotc-agent/conf/iotc-agent.cfg
   ```

9. Set the TPM attestation level to full:

   ```
   tpmAttestationLevel = full
   ```

   Note If you want to enable only boot attestation, then set `tpmAttestationLevel = boot`. If you want to enable both runtime and boot attestation, set `tpmAttestationLevel =full`.

Results

You have successfully associated the attestation profile to your TPM-based device template. You can now onboard your gateway using the TPM-based template.

What to do next

Onboard a gateway using the TPM-based authentication method. For more information, see Onboard a Gateway Using TPM-Based Authentication. After on-boarding your gateway, go to **Audit Log** in the SmartHub INFER IoT Center console and verify that the following audit types are displayed:

- **TPM Boot Attestation Succeeded**
- **Runtime Boot Attestation Succeeded**

If there is an attestation failure, verify the following:

- Verify the **Alerts** tab for any alerts corresponding to the boot or runtime attestation.
- Verify the **Properties** tab of the device.

  If there is a boot failure, the cause of the error is displayed. For example:

  ```
  boot-is-tampered
  true
  boot-tamper-details:
  "PCR8 mismatched."
  ```

For a run-time failure, the cause of the error is displayed. For example:

```
runtime-is-tampered
true
runtime-tamper-details:
"Files with mis-matched digests: /etc/chrony.conf"
```

Applying a Security Profile on Multiple Gateway Devices Using Campaigns

When you upgrade the firmware or apply security patches to your gateway, the golden profile of your gateway changes, but this action does not qualify as file tampering. To avoid attestation failures for such instances, create a security profile corresponding to the change and apply it to all the gateway devices that require an upgrade using campaigns.

Procedure

1 Create a security profile corresponding to the file change.

2 Create an IoTC Package with a label added to the package-spec.yml file. For example:

```
# This is a simple array of strings which are just that - labels by
             # which you could search a package in SmartHub INFER IoT
            Center labels:
                  - security-profile-id:eaa7f966-2915-480f-bf73-2524e019a96d
```

Note The security-profile-id: <label> must match the security profile ID that you create in step 1.

3 Upload the package to SmartHub INFER IoT Center. For information about uploading IoTC Packages, see #unique_134.

4 To apply updates to all gateway devices that require an update, create a campaign with the appropriate query. For information about creating campaigns, see Create a Campaign.

5 Run the campaign.

6 After the campaign runs successfully, click an updated gateway device and verify that the security-profile-id is updated under **Custom Properties**.

Note For the attestation configuration to take effect, you must reboot the upgraded devices by running the following script when the campaign is in the ACTIVATE phase:

```
now="date"
echo $now >> /tmp/activtion.log 2>&1
echo "Starting Activation for Updating Security Profile" >> /tmp/activtion.log 2>&1

sudo shutdown -r +1 >> /tmp/activtion.log 2>&1

echo 0
```

Results

You have successfully applied a security profile across multiple gateway devices using campaigns.

Working with Liota

18

Little IoT Agent or Liota is an open source SDK for building IoT gateway applications.

Liota simplifies the interaction between any device and any data center component, through any gateway and over any transport protocol. With Liota, you can enroll your gateway and connected Thing devices using the Basic, Token-Based, or Property-Based enrollment method.

For more information about Liota, see VMware's Liota documentation at https://github.com/vmware/liota.

To view the Liota Developer Guide, go to https://github.com/vmware/liota/wiki/Liota-Developer-Guide.

This chapter includes the following topics:

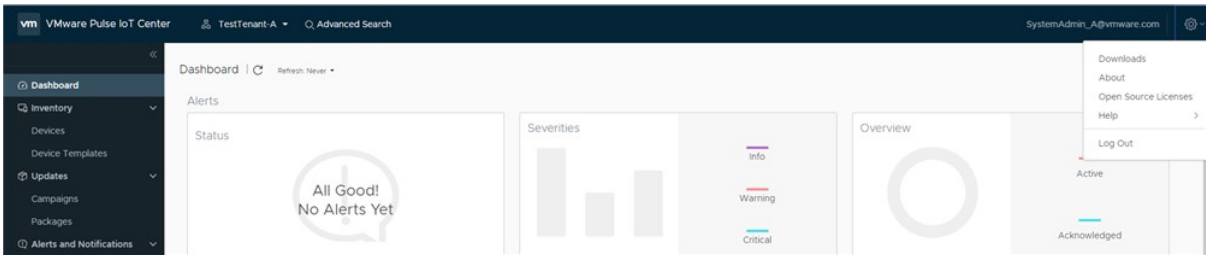
Download Liota v2

This section lists the steps to download the Liota v2 package from the SmartHub INFER IoT Center console.

Procedure

1 Log in to the SmartHub INFER IoT Center console.

2 Click the **Settings** icon at the top right corner and click **Downloads**.



The Downloads window lists the items that can be downloaded from the SmartHub INFER IoT Center console.

3 Download the Liota package and the Pulse Agent package from the list and save them to your local drive.

What to do next

Install the Liota package on your gateway.

Installing Liota

You can install Liota on a gateway running on the Windows or Linux operating system.

Install Liota on Linux

To install the Liota v2 package on your Linux gateway and run as a root user, perform the steps listed in this section.

Procedure

1 Copy the Liota v2 package to the gateway where Liota must be installed.

2 Navigate to the folder containing the Liota package and run the following commands:

```
tar xvzf liota.2.0.0.tar.gz
cd liota
python3 setup.py install
```

Results

The setup.py script installs Liota as a root user, copies the configuration files from /usr/lib/liota/config/ to /etc/liota, creates the /var/log/liota directory, sets the library path LD_LIBRARY_PATH=/opt/vmware/iotc-agent/lib/ <iotc-agent installation directory>, and starts the Liota daemon.

For the Liota service to start automatically on reboot, copy the liota.service script from the scripts folder to the /etc/init.d/ location and enable the Liota service according to your operating system (Debian, Ubuntu, RHEL, or CentOS).

What to do next

Onboard the gateway using Liota.

Install Liota on Windows

To install the Liota v2 package on a Windows 10 IOT Enterprise operating system, perform the steps listed in this section.

Prerequisites

1 Install Python 3:

   a Download Python 3 from: https://www.python.org/ftp/python/3.8.0/python-3.8.0-amd64.exe

   b To start the installation, double-click the .exe file.

   c In the first screen, select the **Add Python 3.8 to PATH** check box.

   d Click **Customize Installation** and in the next screen, click **Next.**

   e In the **Advanced Users** screen, select the **Install for all users** check box and click **Install**.

2 To work with the SmartHub INFER IoTC Agent Default Client, perform the following

steps: a From SmartHub INFER IoT Center, download the iotc-agent file and untar

it.

   b Go to <extracted-agent-folder>\iotc-agent\conf\iotc-agent.cfg and update the value of agentIPCMode to 1.

   c Go to <extracted-agent-folder>\iotc-agent\conf\iotc-defclient.cfg and update the value of defClientIPCMode to 1.

   d Open the Windows PowerShell ISE as administrator and run the Set-ExecutionPolicy RemoteSigned command. Click **Yes to All**.

   e From the Windows PowerShell ISE, run the install.ps1 file.

Procedure

1 Add the C:\Program Files\VMware\iotc-agent\bin\ path to the System Path variable.

2 Open a command prompt as administrator.

3 Navigate to the extracted Liota folder path and run the python setup.py install command.

4 Copy the required packages from the <extracted-liota-folder-path>\examples\pulseV2\ folder to the C:\Program Files\VMware\liota\data\packages folder. Specifically, the reg.py and temperature_sensor.py packages. Replace some packages if needed.

5 To access liota-cli, CD to C:\Program Files\VMware\liota\data\packages\ and run python liotad\liotapkg.py --help.

Results

You have installed Liota on Windows successfully.

What to do next

- You can either copy the configuration files from /usr/lib/liota/config/ to /etc/liota manually and create a /var/log/liota directory, or use the helper script post-install-setup.sh to copy the configuration files.

  a If you require Liota to be installed as a non-root user that is different from the one that exists on the system, run:

  ~~~
  ```bash
    $ cd /usr/lib/liota
    $ LIOTA_USER="non-root user" ./post-install-setup.sh
  ```
  ~~~

  b If you require Liota to be installed as a root user (not preferred), run:

  ~~~
  ```bash
    $ cd /usr/lib/liota
    $ LIOTA_USER="root" ./post-install-setup.sh
  ```
  ~~~

- After enrolling a gateway, run the python reg.py <packagename> command. The package name must not contain the .py extension. For example, python reg.py temperature_sensor.
- You can view the logs from the C:\Program Files\VMware\liota\log\liota.log file.

Onboard Your Linux Gateway Using Liota

This section lists the steps to onboard your Linux gateway to SmartHub INFER IoT Center using Liota.

Prerequisites

Note Ensure that the properties that you provide when enrolling your gateway are secure.

Procedure

1 Copy the general_edge_system.py and iotcc_v2_gateway.py packages from the examples/pulsev2 folder in the Liota installation folder to the /usr/lib/liota/packages/ directory.

2 Modify the gateway name in the general_edge_system.py package:

```
class PackageClass(LiotaPackage):
     def run(self, registry):
        from liota.entities.edge_systems.general_edge_system import GeneralEdgeSystem
        import socket

           # Set the name of the gateway
           edge_system = GeneralEdgeSystem("Gateway001")
           registry.register("edge_system", edge_system)
    def clean_up(self)
      pass
```

3 Load the packages using the following command:

```
cd /usr/lib/liota/packages
sha1sum iotcc_v2_gateway.py (# Use this command to get the SHA1SUM.)
python3 liotad/liotapkg.py load iotcc_v2_gateway "SHA1SUM" (# Use SHA1SUM from the previous
command.)
```

Results

The gateway is onboarded to SmartHub INFER IoT Center. If the gateway is already onboarded, it loads the instance to Liota.

What to do next

Collect metrics: A sample package `temperature_sensor.py` is available in the `examples/pulsev2` folder. Update the callback function to collect the metrics.

Onboard Your Windows Gateway Using Liota

To onboard your Windows gateway, perform the steps listed in this section.

In this example, we enroll the gateway using the Token-Based Authentication method.

Prerequisites

Ensure that you have registered your Windows gateway template on SmartHub INFER IoT Center.

Procedure

1 Download the Liota package from SmartHub INFER IoT Center.

2 Go to the downloaded folder and navigate to **liota** > **examples** > **pulsev2**.

3 Copy all the files in the **pulsev2** folder.

4 Navigate to **Program Files** > **VMware** > **liota** > **data** > **packages** and paste the copied files in the packages folder. Replace packages if needed.

5 From the SmartHub INFER IoT Center console, select your registered Windows gateway device template and click **Actions** > **Create Gateway Credentials** and copy the credentials.

6 From the command prompt of your Windows gateway, run the following command:

```
c:\Program Files\VMware\liota\data\packages>pythond liotad\liotapkg.py enroll token <credentials>
```

Results

You have successfully enrolled your Windows gateway on SmartHub INFER IoT Center.

Enroll a Liota Gateway Using the Token-Based Authentication Method

You can enroll your Liota gateway and its connected Things such as light bulbs and sensors using the Token-based enrollment method.

Prerequisites

- Ensure that you have created a Liota gateway template with Token-Based as the enrollment provider type, and have connected Thing templates to it.
- Ensure that you install Liota and SmartHub INFER IoTC Agent in the /opt/vmware/ folder of your gateway.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** >

Devices. The Devices - All Devices page is displayed.

2 Click **REGISTER**.

The Register Gateway wizard is displayed.

3 Enter your gateway name and select the Liota device template that has Token-Based Authentication enabled to associate with your gateway.

4 Click **REGISTER**.

Your gateway is registered and is listed in the Devices - All Devices page.

5 Create a credential to enroll your gateway. From the Devices -All Devices page, click the gateway that you have registered.

6 Click the **Actions** drop-down menu and select **Create Gateway Credentials**. Click **CREATE**.

7 Copy the token to the clipboard.

The token expiry time that you set when creating the template is displayed.

8 Log in to your gateway and change the directory to /opt/vmware/liota/data/packages.

9 Open the pulse_edge_system.py file and edit the property-based enrollment properties. Run the command: vi pulse_edge_system.py.

10 Under #token, enter the gateway template name, Thing device template name, and paste the token value.

11 To enroll your Liota gateway, run:

```
./liotad/liotapkg.py enroll <token>
```

12. You must receive the following output:

    ```
    Device ID: <device-ID>
    Device enrolled successfully.
    ```

Results

Your Liota gateway is enrolled successfully. Verify that the status of your gateway is displayed as **ENROLLED** in the SmartHub INFER IoT Center console. Also verify that the properties and metrics are populated.

What to do next

Enroll a Thing device to your Liota gateway.

Enroll a Liota Gateway Using the Property-Based Authentication Method

You can enroll a Liota gateway and its connected Thing devices using the property-based authentication method.

Prerequisites

- Ensure that you have created a Liota gateway template with Property-Based as the enrollment provider type, and have connected Thing templates to it.
- Ensure that you install Liota and SmartHub INFER IoTC Agent in the `/opt/vmware/` folder of your gateway.

Procedure

1. From the SmartHub INFER IoT Center UI, go to **Inventory** >

   Devices. The Devices - All Devices page is displayed.

2. Click **REGISTER**.

   The Register Gateway wizard is displayed.

3. Enter your gateway name and select the device template that has Property-Based Authentication enabled to associate with your gateway.

4. Click **REGISTER**.

   Your gateway is registered and is listed in the Devices - All Devices page.

5. Create a credential to enroll your gateway. From the Devices -All Devices page, click the gateway that you registered.

6. Click the **Actions** drop-down menu and select **Create Gateway Credentials**. Click **CREATE**.

7. Enter a value for the keys that you defined when you create the device template. The key and value pair must be unique for all the devices that you have configured under your Organization. The device must send the same key and value pair to the server.
8. Log in to your gateway and change the directory to /opt/vmware/liota/data/packages.
9. Open the pulse_edge_system.py file and edit the property-based enrollment properties. Run the command: vi pulse_edge_system.py.
10. Under #property, enter the gateway template name, Thing device template name, enrollment key, and enrollment value.
11. To enroll your Liota gateway, run:

    ```
    ./liotad/liotapkg.py enroll
    ```

12. You must receive the following output:

    ```
    Device ID: <device-ID>
    Device enrolled successfully.
    ```

Results

Your Liota gateway is enrolled successfully. Verify that the status of your gateway is displayed as **ENROLLED** in the SmartHub INFER IoT Center console. Also verify that the properties and metrics are populated.

Enroll a Thing Device to Your Liota Linux Gateway

To enroll a Thing Device to your Linux gateway, perform the following steps:

In this example, we enroll a temperature sensor device.

Prerequisites

You must have created a Thing device template.

Procedure

1. Log in to your gateway and change the directory to /opt/vmware/liota/data/packages.
2. To enroll the Thing device, run:

    ```
    python3 liotad/liotapkg.py enroll-device <ThingTemplateName> <NameOfThingDevice>
    ```

    ```
    python3 liotad/liotapkg.py enroll-device LightBulbTemplate LightBulb001
    ```

3 Alternatively, you can enroll a Thing device using the reg.py file:

   a Navigate to the **Packages** folder and open the device package in a text editor.

   b Replace the deviceName with your device name.

   ```
   deviceName = "TemperatureSensor001"
   ```

   c Replace TemperatureSensorTemplate with your device template name.

   ```
   d = PulseConnectedDevice(device_name, "TemperatureSensor",
   self.iotcc_edge_system.reg_entity_id)
   ```

   d Save and close the file.

   e Navigate to cd /opt/vmware/liota/data/package and run:

   ```
   python3 reg.py temperature_sensor
   ```

Results

The Thing device is enrolled and connected to your Liota Linux gateway.

What to do next

Run commands on your enrolled Thing device.

Enroll a Thing Device to Your Liota Windows Gateway

To enroll a Thing Device to your Windows gateway, perform the following steps:

In this example, we enroll a temperature sensor device.

Prerequisites

You must have created a Thing device template.

Procedure

1 From the Windows command prompt, run c:\Program Files\VMware\liota\data\packages>pythod liotad\liotapkg.py enroll-device <DeviceTemplateName> <DeviceName>. For example:

```
c:\Program Files\VMware\liota\data\packages>pythod liotad\liotapkg.py enroll-device LightBulb
LightBulb001
```

2 Alternatively, you can enroll a Thing device using the reg.py file:

   a Navigate to the **Packages** folder and open the device package in a text editor.

   b Replace the deviceName with your device name.

   ```
   deviceName = "TemperatureSensor001"
   ```

c Replace TemperatureSensorTemplate with your device template name.

```
d = PulseConnectedDevice(device_name, "TemperatureSensor",
self.iotcc_edge_system.reg_entity_id)
```

d Save and close the file.

e From the command prompt, run c:\Program Files\VMware\liota\data\packages>python reg.py <device_package_name>

```
c:\Program Files\VMware\liota\data\packages>python reg.py temperature_sensor
```

Results

The Thing device is enrolled and connected to your Liota Windows gateway.

Run Commands on Your Thing Device

You can run commands on your enrolled Thing device.

The Liota package contains scripts for running commands on Thing devices such as light bulbs and sensors. When creating a Thing device template, provide the command name and parameters of the client commands and ensure that they match with the definition name and parameters in the script that is available in the Liota package. Next, register the script with SmartHub INFER IoT Center by running the following command:

```
python3 reg.py <Thingdevice_action_script>
```

For example:

```
python3 reg.py light_actions
```

When the gateway receives a client command from SmartHub INFER IoT Center, it runs the definition that is available in the Liota package.

For example, when you send a client command from SmartHub INFER IoT Center for the connected light bulb to turn on, the turnOn definition is run in the Liota package of the gateway and the connected light bulb turns on.

Enrolling BACnet and Modbus Thing Devices Using Liota

BACnet and Modbus are communication protocols used for connecting industrial electronic devices and building automation and control systems.

Note This feature is available as a preview.

BACnet uses the User Datagram Protocol (UDP) and its default port is 47808. Modbus uses the Transmission Control Protocol (TCP) and its default port is 502. You can customize these ports on the Thing device.

The BACnet protocol includes broadcast exchange services such as Who-Is and I-Am, which are used for discovering devices. The gateway sends the Who-Is command to all the IP addresses on the port and the BACnet devices respond with I-Am. The Modbus protocol does not have an in-built discovery support. Liota scans for all Modbus devices on a defined range of IP addresses, on a specified port.

The BACnet protocol uses object type and instance number to get or set properties on a BACnet device. The following table lists the supported BACnet object types:

Object	Data Type	Read/Write
analogInput	Float	Read
analogOutput/analogValue	Float	Read/Write
multiStateInput	Integer	Read
multistateOutput/multiStateValue	Integer	Read/Write
binaryInput	Boolean	Read
binaryOutput/binaryValue	Boolean	Read/Write

The Modbus protocol uses entity type and address to get or set properties on a Modbus device. The following table specifies the entities:

Entity	Data Type	Read/Write
DiscreteInput	Boolean	Read
Coil	Boolean	Read/Write
InputRegister	■ Integer 16/32/64 ■ Float 32/64 ■ String	Read
HoldingRegister	■ Integer 16/32/64 ■ Float 32/64 ■ String	Read/Write

Liota contains example package files to enroll Thing devices that run on the BACnet and Modbus communication protocols.

Create a BACnet Thing Template

Configure the BACnet Thing template with metric types that you want to collect, and add custom commands to send to your BACnet Thing device.

Prerequisites

Ensure that you have:

- Created a Liota gateway template on SmartHub INFER IoT Center.
- Enrolled a gateway using the Liota gateway template.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Device Templates**.

2 In the Inventory > Device Templates page, click **CREATE**.

The Create Device Template wizard is displayed.

3 In the **Details** step, enter the following information:

a Under **Template Name**, enter the name of your Thing template.

b In the **Select Device Type** drop-down menu, select **Thing**.

4 Click **NEXT**.

5 In the **Properties** step, click **NEXT**.

6 In the **Metrics** step, click **+Add** to add a metric and enter the following information. In this example, we configure the metrics for a BACnet thermostat device:

Note Ensure that the metric name and value type that you enter in the BACnet device package matches with the information that you enter here.

a Under **Display Name**, enter the display name of the metric to collect. For example, **Temperature**.

b In the **Value Type** drop-down menu, select the metric value type. For example, **Double**.

c Under **Unit**, enter the unit of the collected metric. For example, **Celsius** or **C**.

7 Click **NEXT**.

8 In the **Connected Device Templates** step, click **NEXT**.

9 In the **Commands** step, click **+Add** to add a command. In this example, we add a command to set the temperature on your thermostat device. Enter the following information:

a Under the **Type** drop-down menu, select **Client Command**.

b Under **Name**, enter a name for your command. For example, **set-temperature**.

c Under **Client Identifier**, enter the client domain. For example, **com.liota.bacnet**.

d Under **Arguments**, enter the following information:

1 **name** - Name of the argument. For example, **Temperature**.

2 **addr** - The unique object type and value of the instance. For example, **{"object":"analogOutput", "instance":"0"}**.

3 **value** - Enter the temperature value to be updated on the device.

4 Click **DONE**.

10 Click **NEXT**.

11 In the **Review** step, review the information that you have entered.

12 To save the Thing template, click **SAVE**.

Results

You have successfully created a BACnet Thing template to collect temperature metrics and have added a custom command to configure the temperature.

What to do next

Add this Thing Template to the Liota gateway template that you have created.

Create a Modbus Thing Template

Configure the Modbus Thing template with metric types that you want to collect, and add custom commands to send to your Modbus Thing device.

Prerequisites

Ensure that you have:

- Created a Liota gateway template on SmartHub INFER IoT Center.
- Enrolled a gateway using the Liota gateway template.

Procedure

1 From the SmartHub INFER IoT Center UI, go to **Inventory** > **Device Templates**.

2 In the Inventory > Device Templates page, click **CREATE**.

The Create Device Template wizard is displayed.

3 In the **Details** step, enter the following information:

a Under **Template Name**, enter the name of your Thing template.

b In the **Select Device Type** drop-down menu, select **Thing**.

4 Click **NEXT**.

5 In the **Properties** step, click **NEXT**.

6 In the **Metrics** step, click **+Add** to add a metric and enter the following information. In this example, we configure the metrics for a Modbus humidity sensor device:

Note Ensure that the metric name and value type that you enter in the Modbus device package matches with the information that you enter here.

a Under **Display Name**, enter the display name of the metric to collect. For example, **Humidity**.

b In the **Value Type** drop-down menu, select the metric value type. For example, **Double**.

c Under **Unit**, enter the unit of the collected metric. For example, **Water vapor**.

7 Click **NEXT**.

8 In the **Connected Device Templates** step, click **NEXT**.

9 In the **Commands** step, click **+Add** to add a command. In this example, we add a command to set the humidity on your humidity sensor device. Enter the following information:

a Under the **Type** drop-down menu, select **Client Command**.

b Under **Name**, enter a name for your command. For example, **set-humidity**.

c Under **Client Identifier**, enter the client domain. For example, **com.liota.modbus**.

d Under **Arguments**, enter the following information:

1 **name** - Name of the argument. For example, **Humidity**.

2 **addr** - The unique entity, address, and type values. For example, **{"entity": "HoldingRegister", "address": 1, "type": "float32"}**.

3 **value** - Enter the humidity value to be updated on the device.

4 Click **DONE**.

Results

You have successfully created a Modbus Thing template to collect humidity metrics and have added a custom command to configure the humidity level.

What to do next

Add this Thing Template to the Liota gateway template that you have created.

Enroll a BACnet Thing Device

Configure the BACnet package in Liota to enroll a BACnet Thing device and collect metrics.

Prerequisites

Ensure that you have:

- Created a Liota gateway template on SmartHub INFER IoT Center.
- Enrolled a gateway using the Liota gateway template.
- Created a Thing template to enroll the BACnet Thing device.
- Added a firewall rule to open the BACnet UDP Port.

Procedure

1 Copy the Liota packages from the installer location to the packages folder in your gateway. Run the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# cp ~/mirrors_github_liota/example/
pulsev2/* .
```

The Liota packages are copied to the packages folder in your gateway.

2 Edit the BACnet device package that you want to configure. In this example, we configure the BACnet Thermostat package to collect metrics. Run the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# vi bacnet_thermostat.py
```

3 The bacnet.thermostat.py package contains definitions for collecting metrics and for running commands.

a Configure the following parameters:

Parameter	Description
device_props	Enter the IP address of your BACnet Thing device. For example, device_props = {"ip":"192.168.101.2"}
comms	Enter the Ethernet name and port number. For example, comms = BacnetDeviceComms({"iface":"eth3", "port":"47808"})
device_name	Enter the device name that is registered on SmartHub INFER IoT Center. For example, device_name = "BacnetTemperatureSensor"
d	Enter the device template name that you have created on SmartHub INFER IoT Center. For example, d = PulseConnectedDevice(device_name, "TemperatureSensorTemplate", self.iotcc_edge_system.reg_entity_id, device_context)
metricName	The name of the metric that is collected in SmartHub INFER IoT Center. Ensure that you provide the same metric name and value type when creating a Thing template. For example, metricName = "Temperature"
args	To configure your thermostat device to send metrics to SmartHub INFER IoT Center, update the object type to analogInput. The following object types are available: ■ analogInput ■ analogOutput ■ analogValue ■ binaryInput ■ binaryOutput ■ binaryValue ■ multiStateInput ■ multiStateOutput ■ multiStateValue

Parameter	Description

	For example,
	`args = [self.reg_device.reg_entity_id,`
	`"Temperature_sensor", {"object":"analogInput", "instance":"0"}]`

d_metric	Define the type of metric that is sent to SmartHub INFER IoT Center. For example:
	`d_metric = Metric(`
	`name=metricName,`
	`mtype='double',`
	`unit=None,`
	`interval=10,`
	`sampling_function=collect_metrics,`
	`sampling_args=args`
	`)`

client_id	The ID for sending the commands. For example, `client_id = "com.liota.bacnet"`
interval	The time interval in seconds to check for the commands. For example, `interval=10`

4 After configuring the package, register it by running the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# python3 reg.py bacnet_thermostat
```

5 Alternatively, you can register

Results

The BACnet thermostat device is enrolled to SmartHub INFER IoT Center. The **Metrics** tab of the device publishes the metrics every 10 seconds.

What to do next

Send a command to the enrolled thermostat device:

1 In the SmartHub INFER IoT Center console, go to **Inventory** > **Devices**.

2 Click the thermostat device that you have enrolled, click the **…** drop-down menu, andclick Commands.

3 Click **SEND COMMAND**.

4 In the Send Command window:

   a Under **Select Command**, select the **set-temperature** command.

   b Under **Arguments**, make sure that the argument name is the same as in the configuration package.

c Under **Address**, enter the object type and instance number. This address is to identify the device property to be set. For example,

```
{"object":"analogOutput", "instance":"0"}
```

d Under **Value**, enter the temperature value.

5 Click **SEND COMMAND**.

6 Refresh the Command History table. The status of the command must change to **EXECUTED**.

7 Verify the update from the **Metrics** tab.

Enroll BACnet Devices Using Discovery

You can enroll multiple BACnet devices to SmartHub INFER IoT Center using BACnet device discovery.

Prerequisites

Ensure that you have:

- Created a Liota gateway template on SmartHub INFER IoT Center.
- Enrolled a gateway using the Liota gateway template.
- Created a Thing template to enroll the BACnet Thing devices.
- Added a firewall rule to open the BACnet UDP Port.

Procedure

1 Copy the Liota packages from the installer location to the packages folder in your gateway. Run the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# cp ~/mirrors_github_liota/example/
pulsev2/* .
```

The Liota packages are copied to the packages folder in your gateway.

2 Edit the BACnet discovery package:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# vi bacnet_discovery.py
```

3 The bacnet.discovery.py package contains definitions for collecting metrics and for running commands.

a Configure the following parameters:

Parameter	Description
comms	Enter the Ethernet name and port number. For example, `comms = BacnetDeviceComms({"iface":"eth3", "port":"47808"})` The broadcast exchange service scans all the IP addresses in the specified Ethernet and port for BACnet devices.

d	Enter the device template name that you have created on SmartHub INFER IoT Center. For example,
	`d = PulseConnectedDevice(device_name, "TemperatureSensorTemplate",`
	`self.iotcc_edge_system.reg_entity_id,`
	`device_context)`

metricName	The name of the metric that is collected in SmartHub INFER IoT Center. Ensure that you provide the same metric name and value type when creating a Thing template. For example, `metricName = "Temperature"`

args	To configure your thermostat device to send metrics to SmartHub INFER IoT Center, update the object type to analogInput. The following object types are available:
	■ analogInput
	■ analogOutput
	■ analogValue
	■ binaryInput
	■ binaryOutput
	■ binaryValue
	■ multiStateInput
	■ multiStateOutput
	■ multiStateValue
	For example,
	`args = [self.reg_device.reg_entity_id,`
	`"Temperature_sensor", {"object":"analogInput", "instance":"0"}]`

d_metric	Define the type of metric that is sent to SmartHub INFER IoT Center. For example:
	`d_metric = Metric(`
	`name=metricName,`

Parameter	Description

	mtype='double',
	unit=None,
	interval=10,
	sampling_function=collect_metrics,
	sampling_args=args
	)

client_id	The ID for sending the commands. For example, client_id = "com.liota.bacnet"
interval	The time interval in seconds to check for the commands. For example, interval=10

4 After configuring the package, register it by running the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# python3 reg.py bacnet_discovery
```

Results

All BACnet devices that are available on the specified Ethernet and port are enrolled to SmartHub INFER IoT Center and are listed in the **Inventory** > **Devices** page. The **Metrics** tab of each device publishes the metrics every 10 seconds.

Enroll a Modbus Thing Device

Configure the Modbus package in Liota to enroll a Modbus Thing device and collect metrics.

Prerequisites

Ensure that you have:

- Created a Liota gateway template on SmartHub INFER IoT Center.
- Enrolled a gateway using the Liota gateway template.
- Created a Thing template to enroll the Modbus Thing device.

Procedure

1 Copy the Liota packages from the installer location to the packages folder in your gateway. Run the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# cp ~/mirrors_github_liota/example/
pulsev2/* .
```

The Liota packages are copied to the packages folder in your gateway.

2 Edit the Modbus device package that you want to configure. In this example, we configure the Modbus Humidifier package to collect metrics. Run the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# vi modbus_humidifier.py
```

3 The modbus.humidifier.py package contains definitions for collecting metrics and for running commands.

a Configure the following parameters:

Parameter	Description
device_props	Enter the IP address of your Modbus Thing device. For example, device_props = {"ip":"10.197.99.43"}
comms	Enter the Ethernet name and port number. For example, comms = ModbusDeviceComms({"iface":"eth3", "port":"47808"})
device_name	Enter the device name that is registered on SmartHub INFER IoT Center. For example, device_name = "ModbusHumiditySensor"
d	Enter the device template name that you have created on SmartHub INFER IoT Center. For example, d = PulseConnectedDevice(device_name, "HumiditySensorTemplate", self.iotcc_edge_system.reg_entity_id, device_context)
metricName	The name of the metric that is collected in SmartHub INFER IoT Center. Ensure that you provide the same metric name and value type when creating a Thing template. For example, metricName = "Humidity"
args	To configure your humidity sensor device to send metrics to SmartHub INFER IoT Center, update the entity type to InputRegister and specify the value type as float 32. The following entity types are available: ■ Coil ■ DiscreteInput ■ HoldingRegister ■ InputRegister For example, args = [self.reg_device.reg_entity_id, "humidity_sensor", {"entity": "InputRegister", "address": 1, "type": "float32"}]

Parameter	Description

d_metric	Define the type of metric that is sent to SmartHub INFER IoT Center. For example:
	d_metric = Metric(
	name=metric_name,
	mtype='double',
	unit=None,
	interval=10,
	sampling_function=collect_metrics,
	sampling_args=args
	)

client_id	The ID for sending the commands. For example, client_id = "com.liota.modbus"
interval	The time interval in seconds to check for the commands. For example, interval=10

4 After configuring the package, register it by running the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# python3 reg.py modbus_humidifier
```

Results

The Modbus humidifier device is enrolled to SmartHub INFER IoT Center. The **Metrics** tab of the device publishes the metrics every 10 seconds.

What to do next

Send a command to the enrolled humidifier device:

1. In the SmartHub INFER IoT Center console, go to **Inventory** > **Devices**.
2. Click the thermostat device that you have enrolled, click the **...** drop-down menu, andclick Commands.
3. Click **SEND COMMAND**.
4. In the Send Command window:
   a. Under **Select Command**, select the **set-temperature** command.
   b. Under **Arguments**, make sure that the argument name is the same as in the configuration package.
   c. Under **Address**, enter the entity type, address, and value type. This address is to identify the device register to be set. For example,

   ```
   {"entity": "HoldingRegister", "address": 1, "type": "float32"}
   ```

   d. Under **Value**, enter the temperature value.

5. Click **SEND COMMAND**.
6. Refresh the Command History table. The status of the command must change to **EXECUTED**.
7. Verify the update from the **Metrics** tab.

Enroll Modbus Devices Using Discovery

You can enroll multiple Modbus devices by specifying the IP address range to scan for Modbus devices.

Prerequisites

Ensure that you have:

- Created a Liota gateway template on SmartHub INFER IoT Center.
- Enrolled a gateway using the Liota gateway template.
- Created a Thing template to enroll the Modbus Thing devices.

Procedure

1. Copy the Liota packages from the installer location to the packages folder in your gateway. Run the following command:

   ```
   root@photon-machine [ /opt/vmware/liota/data/packages ]# cp ~/mirrors_github_liota/example/
   pulsev2/* .
   ```

   The Liota packages are copied to the packages folder in your gateway.

2. Edit the `modbus_discovery.py` package:

   ```
   root@photon-machine [ /opt/vmware/liota/data/packages ]# vi modbus_discovery.py
   ```

3 The modbus.discovery.py package contains definitions for collecting metrics and for running commands.

a Configure the following parameters:

Parameter	Description
device_props	Specify the IP address range in a particular port for Modbus to scan for devices. For example, device_props = {"start-ip-range": "10.197.99.6", "end-ip-range": "10.197.99.8", "port": "502"}
d	Enter the device template name that you have created on SmartHub INFER IoT Center. For example, d = PulseConnectedDevice(device_name, "HumiditySensorTemplate", self.iotcc_edge_system.reg_entity_id, device_context)
metricName	The name of the metric that is collected in SmartHub INFER IoT Center. Ensure that you provide the same metric name and value type when creating a Thing template. For example, metricName = "Humidity"
args	To configure your humidity sensor device to send metrics to SmartHub INFER IoT Center, update the entity type to InputRegister and specify the value type as float 32. The following entity types are available: ■ Coil ■ DiscreteInput ■ HoldingRegister ■ InputRegister For example, args = [self.reg_device.reg_entity_id, "humidity_sensor", {"entity": "InputRegister", "address": 1, "type": "float32"}]

d_metric	Define the type of metric that is sent to SmartHub INFER IoT Center. For example:
	d_metric = Metric(
	name=metric_name,
	mtype='double',
	unit=None,
	interval=10,

Parameter	Description

	`sampling_function=collect_metrics,`
	`                    sampling_args=args`
	`                )`

`client_id`	The ID for sending the commands. For example, `client_id = "com.liota.modbus"`
`interval`	The time interval in seconds to check for the commands. For example, `interval=10`

4 After configuring the package, register it by running the following command:

```
root@photon-machine [ /opt/vmware/liota/data/packages ]# python3 reg.py modbus_discovery
```

Results

All Modbus devices that are available within the specified IP address range are enrolled to SmartHub INFER IoT Center and are listed in the **Inventory** > **Devices** page. The **Metrics** tab of each device publishes the metrics every 10 seconds.

Configuring Logs In the Liota Package

You can configure the log levels for viewing Liota logs in your gateway.

Log levels are DEBUG, INFO, WARNING, ERROR, and CRITICAL.

Procedure

◆ To change the log level, run the following command from `/opt/vmware/liota/data/packages`:

```
./liotad/liotapkg.py log-level <LEVEL>
```

Results

Logs for the level that you have changed and its succeeding levels are displayed.

Unenroll a Thing Device or a Gateway Using Liota

You can unenroll a Thing device or a Liota gateway using the Liota unenroll command.

Procedure

1 To unenroll a Thing or a Gateway device, go to the /opt/vmware/liota/data/packages directory and run the following command:

```
python3 liotad/liotapkg.py unenroll-device <device-ID>
```

The logs display the status of the device as unenrolled.

2 To unenroll a Liota gateway, run the following command:

```
python3 liotad/liotapkg.py unenroll
```

All the Thing devices that are attached to the gateway are automatically unenrolled.

3 Go to the SmartHub INFER IoT Center console and click the **Devices** tab. Verify that the status of the device is **UNENROLLED**.

Note You cannot unenroll a gateway or a Thing device from the SmartHub INFER IoT Center console.

Uninstall Liota

Uninstall Liota using the steps listed in this section.

Procedure

- To uninstall Liota, navigate to the /usr/lib/liota/ directory and run the following command:

```
python3 setup.py uninstall
```

Results

This command removes the /usr/lib/liota/, /etc/liota/, and /var/log/liota/ directories and disables the auto start service.

Integrating with ServiceNow 19

You can integrate SmartHub INFER IoT Center with a ServiceNow instance.

SmartHub INFER IoT Center provides an integration mechanism with ServiceNow to manage your device alerts. When an alert is generated for a device, it creates an incident in the ServiceNow instance.

This chapter includes the following topics:

- Steps to Integrate with ServiceNow

Steps to Integrate with ServiceNow

This section lists the prerequisites and steps to integrate your SmartHub INFER IoT Center instance with ServiceNow.

Prerequisites

You must have access to the following instances:

- SmartHub INFER IoT Center console
- ServiceNow

Procedure

1 Create a REST-based Notification Destination:

   a Log in to the SmartHub INFER IoT Center console and navigate to **Settings** > **Notification Destinations**.

   b To create a notification destination, click **CREATE**.

   The Create Destination wizard is displayed.

   c In the **Details** step, enter the name of your notification destination, an optional description, and click **NEXT**.

   d In the **Settings** step, enable **Secure Protocol**. Enter the Host URL of the ServiceNow instance. For example, **dev79872.service-now.com**. Enter the port number as **443**.

   e Copy the security certificate from the ServiceNow browser and paste it in the **Certificate** text box.

f Select the **Authentication Type** as **Basic** and enter your ServiceNow credentials in the **Username** and **Password** text boxes. Click **NEXT**.

g In the **Review** step, review the details and click **SAVE**.

2 Create a Notification Definition:

a Navigate to **Alerts and Notifications** > **Notification Definitions**.

b In the Alerts and Notifications - Notification Definitions page, click **CREATE**.

The Create Definition wizard is displayed.

c In the **Details** step, enter a name for the notification definition, enter an optional description, and click **NEXT**.

d In the **Settings** step, select the **Type** as **REST Notification**, and select that destination that you created in step 1.

Note ServiceNow provides multiple tables to which you can insert or create a record. In this example, we use the **Incident** table to create a record. To view the full list of tables in ServiceNow, go to the ServiceNow instance and navigate to REST API Explorer.

e In the URL field, append the URL with the path `/api/now/table/incident`.

f Under **Advanced Settings**, add a new header with the header name as **Content-Type**. Enter the header value as **application/json**. Click **DONE**.

g In the **Body Template** text box, enter the keys to be populated in ServiceNow. You can derive the keys from the ServiceNow instance. The following example illustrates a sample body template:

```
{
    "caller_id": "Test User",
    "short_description": "Notification for Alert  ${alertState}",
    "description": "This is an automated notification from SmartHub INFER IoT Center.\n\n
Device Id : ${deviceId}, \n deviceTemplateId: ${deviceTemplateId}, \n Alert Name : $
{alertTemplate}, \n Alert State : ${alertState}, \n Severity  :  ${alertSeverity},  \n
Recommendation : ${recommendation}, \n Alert Definition ID : ${alertDefinitionId}, \n Metric
Value : ${lambda}. \n\n To view additional details, go to the SmartHub INFER IoT Center
Server."
}
```

h Click **NEXT**.

3 Review the details that you have entered and click **SAVE**.

Results

You have successfully integrated SmartHub INFER IoT Center with ServiceNow. When you associate an alert definition with this ServiceNow notification definition, ServiceNow files an incident whenever an alert is triggered.

Integrating Third-Party CMS with SmartHub INFER IoT Center 20

To perform over-the-air software updates, operating system updates, and firmware updates to the gateways and devices managed by SmartHub INFER IoT Center using a third-party content management system (CMS), integrate the third-party CMS with SmartHub INFER IoT Center.

If you use an external CMS to store software, firmware, or operating system updates for your gateways and devices, use the `uploadProgram` API to integrate the CMS with SmartHub INFER IoT Center.

For more information, see the *SmartHub INFER IoT Center API Reference Guide*.

Prerequisites

You must be a SmartHub INFER IoT Center administrator to perform this operation.

Troubleshooting 21

If you encounter problems while using SmartHub INFER IoT Center, you can use a troubleshooting topic to understand and solve the problem, if there is a workaround.

This chapter includes the following topics:

- Troubleshooting Campaign Management
- SmartHub INFER IoTC Agent Connectivity to the SmartHub INFER IoT Server

Troubleshooting Campaign Management

You can find troubleshooting steps for common campaign management problems in this section.

Prerequisites

To monitor the progress of a campaign on the gateway, set the agentLogLevel to 6 in the iotc-agent.cfg file. You can then monitor the system logs to view the progress of the campaign using tools such as journalctl -u or iotc-agent -f.

The SmartHub INFER IoTC Agent Fails to Run with the "Exec Format Error" Message

Workaround:

- Prefix the script with a shebang (#!).
- If you are running an executable, run it in a standalone staging environment without the SmartHub INFER IoTC Agent. If the executable fails, fix the executable and try again. If the executable runs successfully, contact SmartHub Support.

Package Manifest File - the Packages Are Not Downloaded, Activated, or Executed

Workaround:

- Ensure that your package is being executed in the headless mode.
- Ensure that the manifestExecution property is enabled in the iotc-agent.cfg file.

Agent SDK - the Packages Are Not Downloaded, Activated, or Executed

Workaround:

- Ensure that the respective lifecycle phase is scheduled with the defaultClient or the API.
- Ensure that the manifestExecution property is disabled in the iotc-agent.cfg file.

SmartHub INFER IoTC Agent Connectivity to the SmartHub INFER IoT Server

When you onboard a gateway, at times, the devices are unable to connect to SmartHub INFER IoTC Center and an error message is displayed. The syslog messages indicate that there is a connectivity problem.

You must perform the following checks:

1. Verify if the SmartHub INFER IoT instance is reachable from the gateway.
   a. Verify the following log location: /var/log/syslog or /var/log/messages or journalctl.If you see the following error in the log file on the gateway device:

      Curl_easy_perform() failed : Could not connect to server.

   b. Verify the SmartHub INFER IoT Center server. Run the following command: Curl -v https://<pulse-server> Ping <pulse-server>
2. Verify the SmartHub INFER IoTC agent logs for connection errors.
   a. Verify the log location on the gateway device: /var/log/syslog or /var/log/messages or journalctl
3. After enrolling, if there is no communication between the SmartHub INFER IoTC agent and the SmartHub INFER IoT Center server, verify the SmartHub INFER IoTC agent logs for token errors in the
   location : /var/log/syslog or /var/log/messages or journalctl

   The following error message is displayed in the log file on the gateway device: ERROR: GetCommand: HTTP GetCommand Request failed: ["Invalid Device token"]
4. If the preceding step fails, contact SmartHub Support.

   Note If historical data associated with the gateway is not important, then you can try re-enrolling the gateway device. See **Onboarding a Gateway to SmartHub INFER IoT Center** in *SmartHub INFER IoT Center User Guide*.