Oracle® Cloud

Developing Applications with Oracle Internet of Things Cloud Service

23.3.1
E53314-52
July 2023

ORACLE

Oracle Cloud Developing Applications with Oracle Internet of Things Cloud Service, 23.3.1

E53314-52

Primary Author: Oracle Corporation

Contents

Preface

Audience vii
Documentation Accessibility vii
Diversity and Inclusion viii
Related Documents viii
Conventions viii

1 Get Started with Oracle Internet of Things Cloud Service

About Oracle IoT Cloud Service 1-1
Supported Browsers 1-2
Access Oracle IoT Cloud Service 1-2
How to Get Support 1-3
Locate Diagnostic Information for Oracle Support 1-3

2 Create and Manage Device Models

Device Virtualization 2-1
Create a New Device Model 2-2
Using the URI Data Type in Device Models 2-3
Duplicate a Device Model 2-4
Edit a Device Model 2-4
Assign a Device Model to a Cloud Service 2-5
View the Devices Associated with a Device Model 2-5
Print Device Model Settings 2-5
Export Device Model Settings 2-6
Delete a Device Model 2-6

3 Register and Activate Devices

Register a Single Device 3-1
Register a Batch of Devices 3-2
About CSV Batch Registration File Properties 3-3

Activate a Device 3-4
Activating a Batch of Registered Devices 3-5
Activate a Device Connected to a Gateway Device 3-6
Pair a Bluetooth Device to a Gateway Device 3-6

4 Create and Manage Device Policies

Before You Create and Use Device Policies 4-2
Create a Device Policy 4-2
Assign a Policy to a Device 4-5
Add a Policy Function 4-5
Add Attributes to a Policy 4-6
Duplicate a Policy 4-7
Edit Policy Details 4-8
Edit Policy Function Settings 4-8
Increase or Decrease Device Policy Priority 4-9
Delete a Policy Function 4-9
Delete a Policy 4-10
Enable Device Policies 4-10
Use the Formula Editor 4-11

5 Manage Devices

Cloud Service Device Types 5-1
Cloud Service Device Life Cycle 5-2
Locate a Device 5-3
Edit Device Details 5-3
Add and Edit Device Metadata 5-3
Add and Edit Device GPS Coordinates 5-4
View Device Status and Diagnostic Information 5-4
Send a Test Message to a Device 5-4
View the Device Models Assigned to a Device 5-5
View Device Messages 5-5
Manage Alert Messages 5-5
Manage Device Data Messages 5-6
Manage the Device Selection Used in an IoT Application 5-6

6 Learn About Oracle IoT Digital Twin

About the IoT Digital Twin Framework 6-1

About the Oracle IoT Digital Twin Implementation 6-2

7 Use the Oracle IoT Digital Twin Simulator

Typical Workflow for Simulating Devices 7-1
Access the IoT Digital Twin Simulator Dashboard 7-2
Navigate Back to the Simulations Page from Any Page 7-3
Create Simulation Models 7-3
Add a Predefined Simulation Model in Oracle IoT Digital Twin Simulator 7-3
Create a Custom Simulation Model 7-4
Import a Simulation Model 7-6
Create Simulated Devices 7-7
Create a Simulated Device in IoT Device Simulator 7-7
Create a Simulated Device for a Registered Device 7-8
Create a Simulated Gateway Device 7-8
Use Simulated Devices 7-9
Manage Simulated Devices 7-9
View a Simulated Device 7-9
Delete a Simulated Device 7-10
Work with Simulation Models 7-10
Edit a Simulation Model 7-10
Use Expressions and Functions in Simulation Models 7-11
Export a Simulation Model 7-12
Delete a Simulation Model 7-12

8 Create and Manage Connectors

About Connectors 8-1
Use Cases for the Connectors 8-3
Typical Workflow for Connecting Non-IP Devices 8-4
Create and Configure Connectors 8-5
Create an HTTP Server Connector 8-6
Create an MQTT Server Connector 8-7
Create an MQTT Client Connector 8-9
Create an OrangeLora Connector 8-11
Create a ProximusLora Connector 8-12
About Interpreters 8-12
Configure Interpreters 8-13
Configure Interpreters at Runtime 8-14
Learn About Adding Grammars 8-15

9 Create and Manage Explorations

About Oracle IoT Cloud Service Stream Exploration 9-1
Add a Message Exploration to an Application 9-3
Add Stream Explorer References to Static Data 9-4
Create a New Map 9-5
Use Maps to Create Explorations 9-7
Edit a Map 9-9
Edit Explorations 9-10

10 Integrate with External Services

Integrate Enterprise Applications with Oracle IoT Cloud Service 10-1
Integrate Oracle Business Intelligence Cloud Service with Oracle Internet of Things Cloud Service 10-3
Integrate Oracle Analytics Cloud with Oracle Internet of Things Cloud Service 10-5
Integrate Oracle Mobile Cloud Service with Oracle IoT Cloud Service 10-7
Integrate Oracle IoT Cloud Service with JD Edwards EnterpriseOne IoT Orchestrator 10-14
Integrate Oracle Storage Cloud Service with Oracle IoT Cloud Service 10-22
Integrate Oracle Big Data Cloud Service with Oracle Internet of Things Cloud Service 10-25
Integrate Oracle IoT Cloud Service with Oracle Integration Cloud Service 10-32
Add a Custom Certificate for an Integration 10-34
Verify Integration Connectivity 10-35

11 Develop Enterprise Applications That Use Oracle IoT Cloud Service

Typical Workflow for Developing Enterprise Applications That Use Oracle IoT Cloud Service 11-1
Receive Incoming Data From Oracle IoT Cloud Service 11-2
About the Oracle Internet of Things Cloud Service REST API 11-5

12 Troubleshoot

Troubleshoot Integrations 12-1
Troubleshoot Management Console Issues 12-2
Troubleshoot REST API Issues 12-4
Troubleshoot Stream Explorer Issues 12-5

13 Glossary

Preface

Using Oracle Internet of Things Cloud Service describes how to use Oracle Internet of Things (IoT) Cloud Service to connect your devices to the cloud, analyze the data your devices send, and integrate that data with your enterprise applications and other services.

Topics:

- Audience
- Documentation Accessibility
- Diversity and Inclusion
- Related Documents
- Conventions

Audience

Using Oracle Internet of Things Cloud Service is intended for anyone who plans to use Oracle Internet of Things Cloud Service. This audience includes the following:

- Administrators who are responsible for the overall Oracle IoT Cloud Service instance configuration.
- Operators who are responsible for the day-to-day operations in an Oracle IoT Cloud Service instance, such as management of devices lifecycle and monitoring of messages.
- Software developers who create device client software using APIs for the Oracle IoT Cloud Service Client Software Libraries and the Oracle IoT Cloud Service Gateway.
- Customers developing device drivers for their native devices that will be integrated with the Oracle IoT Cloud Service.
- Enterprise application developers who create software applications that utilize the Oracle IoT Cloud Service Client Software Enterprise Libraries and REST APIs to communicate with their devices using the Oracle IoT Cloud Service.

This document assumes you are familiar with web technologies and have a good understanding of the Java programming language and development environments.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

For more information, see these Oracle resources:

- Oracle Cloud at http://cloud.oracle.com
- *Getting Started with Oracle Cloud*
- *Getting Started with Oracle Stream Explorer*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Get Started with Oracle Internet of Things Cloud Service

Need help setting up your Oracle Internet of Things Cloud Service instance? Forgotten how to access the management console? Use these topics to learn more about getting started with Oracle Internet of Things Cloud Service.

Topics

- About Oracle IoT Cloud Service
- Supported Browsers
- Access Oracle IoT Cloud Service
- How to Get Support
- Locate Diagnostic Information for Oracle Support

About Oracle IoT Cloud Service

The Internet of Things integrates different technologies such as mobile, cloud, big data, and analytics, and connects everyday objects to each other and to the Internet.

Oracle Internet of Things Cloud Service simplifies IoT so you can rapidly assimilate IoT into your digital strategy and create innovative services with less risk. The integration with your existing applications and processes is key to a successful IoT strategy. Oracle Internet of Things Cloud Service makes this integration easy and efficient.

Managing and analyzing the enormous amount of real-time data generated by all the IoT-connected devices demands a multi-faceted, yet robust IoT solution. Oracle Internet of Things Cloud Service provides you real-time analysis tools that let you correlate, aggregate, and filter incoming data streams. It also provides you built in integrations that allow the automatic synchronization of data streams with Oracle Business Intelligence Cloud Service.

Oracle Internet of Things Cloud Service enables secure and reliable bidirectional communication between IoT devices and the cloud. The devices can connect to the cloud directly, or indirectly through a gateway. Oracle Internet of Things Cloud Service assigns a unique digital identity to each device to establish trust relationships among devices and applications. It also enforces authentication and authorization for end-to-end communication security and to ensure proof of origin of data. It uses a cross-protocol functionality that lets you directly address any device connected to the cloud, regardless of network protocol and firewall restrictions. It is also provides reliable communication between the cloud and your devices, even over unreliable networks or with devices that connect intermittently.

You can access Oracle Internet of Things Cloud Service data and functions from your enterprise applications by using REST APIs. You can also use REST APIs to securely integrate your connected devices with your enterprise applications.

Supported Browsers

Supported Browsers

Oracle Internet of Things Cloud Service supports the following web browsers:

- Google Chrome
- Microsoft Edge
- Mozilla Firefox
- Apple Safari

See Oracle Software Web Browser Support Policy for more information

Access Oracle IoT Cloud Service

Use REST API calls or the management console to access and manage your Oracle Internet of Things Cloud Service instance.

This procedure assumes that you have created your Oracle Internet of Things Cloud Service instance.

1. Sign in to Oracle Cloud:
   a. Open a web browser and browse to http://cloud.oracle.com.
   b. Click **Sign In**.
   c. Select **Traditional Cloud Account** in the first list unless you've been told by email or by an Oracle Sales Representative that you are using a Cloud Account with Oracle Identity Cloud Service (IDCS).
   d. Select the data center where your services are located in the **Select Data Center** list.
   e. Click **My Services**.
   f. Enter the identity domain provided in your welcome email and then click **Go**.
   g. Enter your user name and your password and then click **Sign In**.
2. In the **Cloud Services** list, click the **Menu** ( ) icon for **IoT Enterprise** and then select **Open Service Console**. If **IoT Enterprise** is not listed, click **Customize Dashboard**, scroll to **IoT Enterprise,** click **Show**, and then close the dialog.
3. Click **Go to Console** if this is your first time accessing Oracle Cloud.
4. Select one of these options:
   a. Click **Create Service** if you have not previously created an Oracle Internet of Things Cloud Service instance or you want to create a new instance.
   b. Click the **Menu** ( ) icon of an existing Oracle Internet of Things Cloud Service instance and then select **Management Console**.

How to Get Support

Use these resources to resolve problems:

- Visit the Oracle Help Center at http://docs.oracle.com/en/.
- If you're an Oracle Premier Support Customer, visit My Oracle Support.
- Contact Oracle Technical Support. See Contacting Oracle Support in *Getting Started with Oracle Cloud*.

Locate Diagnostic Information for Oracle Support

To resolve support issues quickly, send the Oracle support team the diagnostic information listed in the table.

To locate and view diagnostic information, see Find Diagnostic Information to Help with Troubleshooting.

Diagnostic Information	Description	Example
Server Version Number	Record the version number of the Oracle Cloud instance.	17.1.5.0
Client Platform Version	Record the client platform version.	Client library information for each platform.
Java or Android Compiler Version	Record the version number of the device Java or Android compiler.	Android 7.0
Error Message	Record the error message or attach a screen image.	*Operation failed. Cause: {"type":"https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html","title":"Project was not found: ProjectId{iotAppId='0-AD', serviceId='0-AD'}","status":"404"}*
Oracle Internet of Things Cloud Service URL	Record the URL of the Oracle Internet of Things Cloud Service instance.	`https:/targetcloudinstance.com`
Error Message Time Stamp	Record the time and date the error occurred.	Tue, Feb 7, 2017 1:07:42 PM
Log File	Provide a log file that documents the issue.	*log.txt*
Screen Images	Attach relevant screen images of the error message or the page where the error occurred.	Include a screen image with annotations.
Navigation Path	Provide the navigation path to the page where the error occurred.	*Device Model > Device Model Details*

2

Create and Manage Device Models

A device model is an interface that lets any device communicate with Oracle Internet of Things Cloud Service regardless of its manufacturer or operating system. A device model can represent a device, gateway, device adapter, or a device application.

Note that Oracle Internet of Things (IoT) Intelligent Applications provide a Direct Ingestion option to ingest telemetry data directly into your IoT applications, eliminating the need to use client libraries, define device models, and register devices.

Direct ingestion allows devices to directly communicate with the digital twin using standard protocols such as HTTP and MQTT, standard authentication mechanism, and common payload formats. This feature is currently available only for Oracle IoT Asset Monitoring and Oracle IoT Production Monitoring. See *Oracle® Cloud Using Oracle Internet of Things Asset Monitoring Cloud Service* and *Oracle® Cloud Using Oracle Internet of Things Production Monitoring Cloud Service* for more details on direct data ingestion.

Topics

- Device Virtualization
- Create a New Device Model
- Duplicate a Device Model
- Edit a Device Model
- Assign a Device Model to a Cloud Service
- View the Devices Associated with a Device Model
- Print Device Model Settings
- Export Device Model Settings
- Delete a Device Model

Device Virtualization

Virtualization is the act of creating a virtual (rather than physical) version of a computer application, storage device, or a computer network resource. Virtualization makes the logical server, storage, and network independent of the deployed physical infrastructure resources.

Oracle Internet of Things Cloud Service uses virtualization to make it easier to integrate external devices and data with Oracle Internet of Things Cloud Service. Oracle Internet of Things Cloud Service exposes every connected device as a set of resources called a device model. The use of device virtualization abstracts any complexity associated with device connectivity and standardizes device integration with the enterprise. With it, enterprise applications can directly address any device from Oracle Internet of Things Cloud Service regardless of network protocol and firewall restrictions.

A device model is a single entity that represents the interface through which Oracle Internet of Things Cloud Service can interact with a specific device type, regardless of the vendor, underlying standard, or specification that defined that device model. It can represent any

object on the device side that can send messages or respond to REST requests. This object type includes devices, gateways, device adapters, and device applications. Through a device model, Oracle Internet of Things Cloud Service has access to the following:

- metadata associated with a device type
- message formats generated by a device
- exposed web resources that can be used to send commands
- device capabilities, such as device software management

Create a New Device Model

A device model is an interface that lets any device communicate with Oracle Internet of Things Cloud Service regardless of its manufacturer or operating system.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

   `https://hostname/ui`

   Here, *`hostname`* is the host name of your Oracle Internet of Things Cloud Service instance.

2. Click the **Menu** (☰) icon.
3. Select **Devices** and then select **Model**.
4. Select one of these options:
   - If you have not previously created a device model, click **Create Device Model**.
   - If you have previously created a device model, click the **Add** (⊕) icon.
5. Complete these fields:
   a. **Name**: Enter a name for the device model.
   b. **Description**: Enter an optional description for the device model.
   c. **URN**: Enter a unique identifier for the device model. Use this format: `urn:com:<mycompany>:<mydevice>:<what the device model does>`.
6. Select system attributes for the device model.
7. (Optional) Add custom attributes for the device model:
   a. Expand the **Custom Attributes** option list.
   b. Click the **Add** (⊕) icon.
   c. Enter a name for the custom attribute in the **Name** field.
   d. Enter an optional description for the custom attribute in the **Description** field.
   e. Select a data type in the **Type** list.
   f. Select **Writable** if you want to make the custom attribute writable.
   g. Click **OK**.
8. (Optional) Define the actions that can be invoked on the device:

a. Expand the **Actions** option list.

b. Click the **Add** (⊕) icon.

c. Enter a name for the action in the **Name** field.

d. Enter an optional description for the action in the **Description** field.

e. Select the data type for the action in the **Arguments** list.

f. Enter an optional alternate name for the action in the **Alias** field.

g. Click **OK**.

9. (Optional) Create alerts and custom message formats for the device model:

a. Expand the **Alerts and Custom Messages** option list.

b. Click the **Add** (⊕) icon.

c. Enter a name for the alert or custom message in the **Name** field.

d. Enter an optional description for the alert or custom message in the **Description** field.

e. Enter a unique identifier for the alert or custom message in **URN** field. Use this format:`urn:<mycompany>:<department>:<mydevice>:<device model>:<message>`.

f. Select a data type in the **Type** list.

g. Click **OK**.

h. Select the alert message format and then click the **Add** (⊕) icon in the **Fields** column.

i. Enter a name for the message type in the **Name** field.

j. Select a data type in the **Type** list.

k. Select **Optional** to indicate the field value can be missing in the device model message format.

l. Click **OK**.

10. Click **Save**.

Using the URI Data Type in Device Models

The URI (Uniform Resource Identifier) data type helps store pointers or references to external content.

If you have devices that need to support large content in messages, such as camera images and videos from facilities monitoring data, you can use the URI data type in your device model attributes. The URI data type lets you store pointers to your content, which can be stored in an external location, such as Oracle Storage Cloud Service or Oracle Documents Cloud Service.

Use the URI data type for device model attributes, action arguments, and format field types that must point to external content.

You can use the client software libraries to create applications that can automatically upload the external content to Oracle Storage Cloud Service, and put URI pointers to the content in your device messages. See Developing Device Software Using the Client Software Libraries in for more information on client software libraries.

Duplicate a Device Model

Duplicate a device model to quickly copy the settings of an existing device model to a new device model.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

   `https://hostname/ui`

   Here, *hostname* is the host name of your Oracle Internet of Things Cloud Service instance.

2. Click the **Menu** ( ) icon.
3. Select **Devices** and then select **Model**.
4. Click the **Duplicate** ( ) icon.
5. Complete these fields:
   a. **Name**: Enter a new name for the device model.
   b. **Description**: Enter an optional description for the device model.
   c. **URN**: Enter a new unique identifier for the device model. Use this format: .
6. Select system attributes for the device model.
7. (Optional) Add or edit the custom attributes for the device model.
8. (Optional) Add or edit the actions that can be invoked on the device
9. (Optional) Add or edit the alerts and custom message formats for the device model:
10. Click **Save**.

Edit a Device Model

Edit a device model to edit, add, duplicate, or remove device model settings including the device model name, description, and attributes.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

   `https://hostname/ui`

   Here, *hostname* is the host name of your Oracle Internet of Things Cloud Service instance.

2. Click the **Menu** ( ) icon.
3. Select **Devices** and then select **Model**.
4. Click the **Edit** ( ) icon.
5. Edit the device model settings.

6. Click **Save**.

Assign a Device Model to a Cloud Service

To use a device model in a specific cloud service, you must associate it with the cloud service.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

   `https://hostname/ui`

   Here, *hostname* is the host name of your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, and then click **Applications**.
3. Click the entry corresponding to the Oracle IoT Asset Monitoring Cloud Service application.
4. Click **Device Model**.
5. Click the **Choose Device Model** () icon.
6. Select the **Add** checkbox for the device model you want to assign to the cloud service.
7. Click **Done**.

View the Devices Associated with a Device Model

View the devices associated with the device model to determine how many devices are using the device model.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

   `https://hostname/ui`

   Here, *hostname* is the host name of your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** () icon.
3. Select **Devices** and then select **Model**.
4. Click the **Device** () icon.

Print Device Model Settings

Print the device model settings to view a hard copy of the device model settings.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

`https://hostname/ui`

Here, *hostname* is the host name of your Oracle Internet of Things Cloud Service instance.

2. Click the **Menu** ( ) icon.
3. Select **Devices** and then select **Model**.
4. Click the **Print** ( ) icon.
5. Select a printer.
6. Click **OK**.

Export Device Model Settings

Export the device model settings to use the device model settings in another application or to save a copy of the device model settings as a backup in case of a system failure.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

   `https://hostname/ui`

   Here, *hostname* is the host name of your Oracle Internet of Things Cloud Service instance.

2. Click the **Menu** ( ) icon.
3. Select **Devices** and then select **Model**.
4. Click the **Export** ( ) icon.
5. Click **Save File**.
6. Click **OK**.
7. Browse to a location to save the file.
8. Click **Save**.

Delete a Device Model

Delete a device model when it is no longer required.

1. Open the Oracle Internet of Things Cloud Service Management Console.

   You can access the Oracle Internet of Things Cloud Service Management Console from the following URL:

   `https://hostname/ui`

   Here, *hostname* is the host name of your Oracle Internet of Things Cloud Service instance.

2. Click the **Menu** ( ) icon.
3. Select **Devices** and then select **Model**.

4. Click the **Delete** () icon.

   A warning appears if the device model is in use. If you delete the device model, the related message flows, explorations, integrations, and device message links are affected as well.

5. Click **Continue**.

3

Register and Activate Devices

Data cannot be sent or received by a device until it is registered and activated. You can register devices individually or in a batch using CSV files. Devices are indirectly or directly connected to Oracle Internet of Things Cloud Service. Indirectly connected devices connect to a gateway and the gateway forwards data to Oracle Internet of Things Cloud Service. Indirectly connected devices are automatically registered and activated by the gateway. A device adapter for the indirectly connected device's protocol must be installed on the gateway to allow the exchange of data between the gateway, the indirectly connected device, and Oracle Internet of Things Cloud Service.

Topics

- Register a Single Device
- Register a Batch of Devices
- About CSV Batch Registration File Properties
- Activate a Device
- Activating a Batch of Registered Devices
- Activate a Device Connected to a Gateway Device
- Pair a Bluetooth Device to a Gateway Device

Register a Single Device

To communicate with Oracle Internet of Things Cloud Service, every device that is connected to Oracle Internet of Things Cloud Service must be registered and then activated. All devices are registered as a Directly Connected Device (DCD). During activation, the device indicates support for indirect enrollment. A device indicating indirect enrollment capability is automatically changed from DCD to Gateway.

1. Open the Oracle Internet of Things Cloud Service Management Console. The URL format to access the IoT management console is: `<iot instance name>.<domain name>/ui`. For example: `https://myiotcs.mydomain.oraclecloud.com/ui`.
2. Click the **Menu** (☰) icon adjacent to the Oracle Internet of Things Cloud Service title on the Management Console.
3. Click **Devices**.
4. Click **Registration**.
5. Click **Register Single Device**.
6. Complete the optional and mandatory fields.

> **Note:**
>
> If you leave the **Activation Secret** field blank, a value is auto-generated and displayed when the device registration is confirmed. You can enter your own Activation Secret value. Any additional information, such as Name, Description, and Metadata are optional, but can be useful as search criteria when managing your registered devices.

7. Click **Register**.
8. (Optional) Click **Select Devices to Provision** and select one or more indirectly connected devices to provision the gateway with the shared secret credentials of the selected devices.

   Indirectly connected devices that use controlled roaming can connect through a gateway only if the gateway has the shared secret credentials to activate the device.
9. Enter a password in the **File Protection Password** field to encrypt the provisioning file that contains the configuration and credentials to activate your device.
10. Enter the password again in the **Confirm Password** field.
11. Click **Download Provisioning File** to download the provisioning file to your computer.

    The provisioning file is required to run your client.
12. Click **Finish**.
13. Click **Management** .

    The registered device is listed in the Management pane with a State value of `Registered` and Type value of `Unknown`.
    See Cloud Service Device Life Cycle, and Locate a Device for more information about device states, types, and managing your devices.

Register a Batch of Devices

You can register a batch of devices that you want to connect to Oracle Internet of Things Cloud Service using a comma-separated values (CSV) file.

For information about CSV file properties, see About CSV Batch Registration File Properties.

1. Open the Oracle Internet of Things Cloud Service Management Console. The URL format to access the IoT management console is: `<iot instance name>.<domain name>/ui`. For example: `https://myiotcs.mydomain.oraclecloud.com/ui`.
2. Click the **Menu** ( ) icon adjacent to the Oracle Internet of Things Cloud Service title on the Management Console.
3. Click **Devices**.
4. Click **Registration**.
5. Select one of these options:

- Click **Download CSV template** to download a CSV template that you can complete.

> **Note:**
>
> The CSV file contains the mandatory and optional property values for each device. If a value is not provided for the optional properties, insert a comma to indicate that a value is not provided. In the last line of the sample CSV file, a comma indicates that property values are not provided for `ActivationId` and `Activation Secret`

- Click **Batch Registration** to upload an existing CSV file.

6. Click **Browse** and browse to the CSV file that contains the registration information for the devices you are registering.
7. Click **Next** when the CSV registration file is successfully uploaded.

   If the Review page contains a warning () icon, select one of these options:

   - **Update** - Choose this option if you want to update the information for an existing registered device. The registered device has the same manufacturer, model and serial number as one of the devices listed in the CSV registration file.
   - **Ignore** - Choose this option if you do not want to include the device in the current registration process.
8. Click one of these options:
   - **Next** - Click to proceed to register the items in the CSV registration file that have been identified as being viable candidates for registration.
   - **Cancel** - Click to discontinue the current batch registration process.
9. Enter a password to encrypt the provisioning file that contains the configuration and credentials to activate your device.
10. Enter the same password in the **Confirm Password** field.

    If both passwords match the Download Provisioning File button becomes enabled.
11. Click **Download Provisioning File** to download the provisioning file to your computer.

    You will need this file to run your client.
12. Click **Finish**.
13. Click **Management** on the left menu.

    The registered devices are listed in the Management page and have the state of `Registered`.
14. Activate the registered devices to begin a secure communication between the devices and Oracle Internet of Things Cloud Service. See Activating a Batch of Registered Devices.

About CSV Batch Registration File Properties

The following table provides descriptions of the properties that appear in the Comma Separated Values (CSV) file used to register a batch of devices with Oracle Internet of Things Cloud Service. Mandatory and optional values are described in the table and are listed in the order they are expected to appear in the CSV file.

To register a batch of devices with Oracle Internet of Things Cloud Service, see Register a Batch of Devices.

Property	Required / *Optional*	Description
`Name`	*Optional*	The `String` data type assigned to the registered device. This value can be modified after device registration.
`Manufacturer`	**Required**	The manufacturer of the device.
`Model Number`	**Required**	The model number of the device
`Serial Number`	**Required**	The serial number of the device.
`Activation ID`	*Optional*	A Device Unique Identifier (UID) that is required for device activation. If a value is not specified, an auto-generated value is assigned to the device after a successful registration. The value cannot be changed after the device is successfully registered.
`Activation Secret`	*Optional*	The Activation Secret (also known as Shared Secret) value required to activate your device. If a value is not specified, an auto-generated string value is assigned to the device after a successful registration. This value is available after a successful registration. This value can be modified before you modify your device.
`Latitude`	*Optional*	The decimal notation of the latitude of the device's position. For example: -43.5723 [World Geodetic System 1984]. If you specify the latitude, then you must also specify the longitude.
`Longitude`	*Optional*	The decimal notation of the longitude of the device's position. For example: , e.g. -43.5723 [World Geodetic System 1984]. If you specify the longitude, then you must also specify the latitude.
`Altitude`	*Optional*	The decimal notation of the altitude of the device's position, in meters above sea level.
`Accuracy`	*Optional*	The accuracy of the device's position in meters. This must be a positive number or zero. An accuracy value can only be specified if the latitude and longitude are provided.
`Metadata`	*Optional*	Key/value pairs that are listed in successive columns. There must be an even number of columns containing keys and values. If there is an odd number of columns, an error message is returned.

Activate a Device

A device can be activated after it is registered and an application has been created and run on the device. During activation, the device indicates support for indirect

enrollment. A device indicating indirect enrollment capability is automatically changed from DCD to Gateway.

1. Register your directly connected device. See Registering a Single Device.
2. Create an application for the device using the Oracle Internet of Things Cloud Service Client Software Library APIs. See Developing Device Software Using the Client Software Libraries.

   When using the Java Client Library, for example, use the following steps to initialize and activate the device:

   a. Add this statement to the device application code to initialize the device:

   ```
   DirectlyConnectdDevice dcd = new
   DirectlyConnectedDevice(configFilePath, configFilePassword);
   ```

   b. Add this statement to the device application code to activate the device:

   ```
   if (!dcd.isActivated())
   { dcd.activate(deviceModelUrn); }
   ```

3. Verify the device has been activated:
   a. Open the Oracle Internet of Things Cloud Service Management Console.
   b. Click the **Menu** () icon adjacent to the Oracle Internet of Things Cloud Service title on the Management Console.
   c. Click **Devices**.
   d. Click **Management**.
   e. Locate the device in the device table or use the **Property** and **Value** fields at the top of the table to search for a specific device.
   f. Verify `Activated` and not `Registered` is displayed in the **State** column.

Activating a Batch of Registered Devices

After you've registered a batch of devices, you need to activate the devices before they can securely communicate with Oracle Internet of Things Cloud Service.

1. Register the devices and download the provisioning file. See Registering a Batch of Devices.
2. Activate each of the registered devices. See Activate a Device.
3. Verify that each of the registered devices has been activated.
   a. Open the Oracle Internet of Things Cloud Service Management Console.
   b. Click the **Menu** () icon adjacent to the Oracle Internet of Things Cloud Service title on the Management Console.
   c. Click **Devices**.
   d. Click **Management**.
   e. Locate the device in the device table or use the **Property** and **Value** fields at the top of the table to search for a specific device.

f. Verify `Activated` and not `Registered` is displayed in the **State** column.

Activate a Device Connected to a Gateway Device

Activation of a device connected via a gateway device is done by the gateway device that is directly connected to the IoT Cloud Service and that has already been activated successfully.

A device that sends data over a non-HTTPS or TCP/IP protocol or interface, such as Bluetooth, Zigbee, I2C, or GPIO, can only communicate with Oracle IoT Cloud Service via a gateway device that is already connected to Oracle IoT Cloud Service. The gateway device must have the Oracle IoT Cloud Service Gateway already configured in it. It must also have the appropriate device adapter that recognizes the protocol of the device that is trying to connect indirectly to Oracle IoT Cloud Service. The gateway device is responsible for authenticating and trusting the device connected via the gateway device before the gateway device can send a request for the device's activation. The gateway device to which the device is connected must already have a trusted connection with Oracle IoT Cloud Service before the device can be registered and activated. Additional authentication requirements, such as knowing a pre-provisioned shared secret associated with the device connected via the gateway, can be placed on the gateway device that is making the request.

Pair a Bluetooth Device to a Gateway Device

Your Bluetooth device needs to be paired with your gateway device so that the gateway device can send data from your Bluetooth device to Oracle IoT Cloud Service. Your gateway device will automatically register the Bluetooth device by sending the Bluetooth device's endpoint information to the Oracle IoT Cloud Service server.

1. Attach a Bluetooth USB dongle to the gateway device.
2. Login to your gateway device, open a command prompt, and then run this command to confirm that BlueZ software stack version 4.98 or later is installed on your gateway device:

   ```
   hcitool | grep ver
   ```

   The output whould be similar to the following:

   ```
   hcitool - HCI Tool ver 4.101
   ```

3. Run this command to list the Bluetooth devices currently paired with the gateway:

   ```
   bt-device -l
   ```

   > **Note:**
   >
   > If you receive the following message after running that last command, use the next step to modify the `udev` rules to allow non-root users to access Bluetooth services:
   > `bt-device: bluez service is not found Did you forget to run bluetoothd?`

4. (Optional) Modify the `udev` rules to allow non-root users access to Bluetooth.

a. As the root user or using the `sudo` command, create a `/etc/udev/rules.d/bluetooth.rules` file.

```
sudo touch /etc/udev/rules.d/bluetooth.rules
```

b. Determine the path to the Bluetooth adapter in your system's `/sys` folder using the `find` command.

```
find /sys -name bluetooth
```

You should see something like the following

```
/sys/devices/platform/bcm2708_usb/usb1/1-1/1-1.2/1-1.2:1.0/bluetooth
```

c. Add the following line to the `bluetooth.rules` file, using the path found in the previous step for the `DEVPATH` property entry. You also need to append `/hci[0-9]*` to the `DEVPATH` value.

```
ACTION=="add", SUBSYTEM=="bluetooth", KERNEL="hci[0-9]*"
DEVPATH="/sys/devices/platform/bcm2708_usb/
usb1/1-1/1-1.2/1-1.2:1.0/bluetooth/hci[0-9]*, RUN+="/bin/sh -c
'cd /sys%p; chown :bluetooth export unexport; chmod g+w export
unexport'"
```

d. Add the user to the `bluetooth` group, which should be already present in the system.

i. (Optional) Add bluetooth group, if needed.

```
sudo groupadd bluetooth
```

ii. Add user to to the bluetooth group using the following:

```
sudo usermod —aG bluetooth <username>
```

e. Reboot your gateway device.

5. (Optional) If the Bluetooth device is already showing as paired with the device gateway but you are having problems connecting to the device, remove the pairing information.

```
sudo bt-device -r BT_ADDR
```

where *BT_ADDR* is a colon-separated value. For example, for a Nonin Pulse Oximeter device, it would be similar to `00:1C:05:00:`*XX:XX*.

6. Determine the device's Bluetooth address by using the `hcitool scan` command, as illustrated below.

```
hcitool scan
Scanning ...
            40:A6:Z8:T7:795:8C      bt-device0
            40:X6:D9:G8:58:A8       bt-devoce1
            40:R6:E9:N9:19:0G       bt-device2
```

7. To pair the Bluetooth device, turn it off and turn it back on to place it back in discovery mode. For example, for the Nonin Pulse Oximeter, take out its batteries to turn it off and reinsert its batteries in the device to turn it back on and pair it.

8. Set the Bluetooth device aside and type the following command quickly (the pairing period expires in a short period).

```
sudo bt-device -c <BT_ADDR>
```

where *<BT_ADDR>* is a colon separated value. For example, for a Nonin Pulse Oximeter device with a pairing information of `00:1C:05:00:B7:E4`, you would type the following:

```
sudo bt-device —c 00:1C:05:00:B7:E4
```

You would then see something similar to the following:

```
Connecting to: 00:1C:05:00:b7:e34 Agent registered

Device: Nonin_Medical_Inc._577981 (00:1C:05:00:B7:E4)
```

When prompted for the PIN code, locate it on the device or in the device's user manual, and type it at the prompt. In the following example, `567890` is the sample PIN code.:

```
Enter PIN code: 567890

Agent released

Done
```

9. Verify that the Bluetooth device is paired with the gateway device:

```
sudo bt-device —l
```

The output displayed would be similar to the following:

```
Added devices:Nonin_Medical_Inc._577981 (00:1C:05:00:B7:E4)
```

4

Create and Manage Device Policies

Oracle has simplified edge computing implementation on Oracle Internet of Things Cloud Service by letting you create and modify policies externally from the device application. You do not need to write or compile any code. Instead, you create policies on the Management Console that define the data to collect, the data computation(s) to perform, and the frequency of data transmission.

After you create your policy, it is transferred to the client library. Moving policies out of the device application to the client library reduces the size and complexity of the device application and reduces the amount of data sent by the device. With policies, data no longer needs to be sent over the network as it is generated. The policy instructs the edge device to compile the data and send it to the central server at a defined interval. By only sending important data, edge computing significantly reduces network traffic.

What is Edge Computing?

With edge computing, data computations are processed at the edge of the cloud as close to the originating source as possible. Instead of sending raw data to a central server for computation, basic data computations are done at the collection point. Moving basic computations closer to the originating source reduces the amount of data that is sent to a central server, reduces the computing power needed by the central server, and improves the speed of data computations.

Edge computing is an ideal solution if your IoT devices have poor or intermittent connectivity to the cloud, or your data processing is latency-sensitive. With edge computing, devices do not need to be constantly connected to the cloud and latency is reduced or eliminated by performing data processing at the source. More importantly, edge computing lets you respond to local events in near real-time and helps you resolve issues quickly.

A Use Case for Edge Computing

Consider a manufacturing enterprise with factories and machines. Sensors installed on production machinery help a manufacturing facility determine when production line equipment requires maintenance or replacement.

A critical component of the production line is a labelling machine. The labelling machine must have a constant supply of ink and the ink must be kept at a specific temperature. Sensors installed in the labelling machine liquid storage containers and the valves and pumps that move the ink constantly send sensor data to a central server. When the sensor data arrives at the central server, computations are performed on the data to determine if any action is required. A lot of data is constantly communicated back and forward from the labelling machine to the central server. A loss in communication can shut the labelling machine down and disrupt production.

With edge computing, the labelling machine is equipped with a local computing solution that performs all of the computations that are typically performed by the central server. All decision making is performed locally and a package of collated data is sent to the central server daily where it can be monitored and human intervention can occur as required.

Edge computing prevents a preventative shutdown of the labelling machine when a communication error occurs. Manufacturing operations continue for a much longer period and the likelihood of a production disruption is significantly reduced.

Before You Create and Use Device Policies

Useful information before you start creating device policies for your devices.

About Device Policies

Device policies filter and process device data that was earlier done by applications. Take a note of these points before you create and use a device policy:

- Make sure that the device policies when processing data validates that the data adheres to the device model rules, such as checking whether the range of the device data is valid.
- For accurate assigning of a device policy to a device, make sure that you have associated the device with the correct device model.
- A device can be associated with only one device policy at a time, which means that assigning a new policy to a device may remove the previously assigned policy.
- Using device policies may change the data transmitting from the device to the Cloud Service. It may stop and alter the data flow such as in `filter and summary` device policies. It may change the original data such as in `summary` device policies.
- Device policies may affect IoT analytics data by:
  - Changing the timing of the device data coming into the server. For example, the **Eliminate Duplicates** function of a policy will eliminate duplicate device values.
  - Changing the device data completely. For example, the **Mean** function of policy will send a calculated mean value of device values.
  - Preventing device data from coming into the server. For example, the **Filter Condition** function of a policy will filter values, preventing some device values from coming into the server.

Create a Device Policy

Create a device policy to define the data to collect, to filter unimportant data, to perform data computation(s), and to specify the frequency of data transmission.

A device model can contain one or more policies but a device can belong to only one policy at a time. Each policy contains one or more functions that form a policy pipeline. The functions will process the device data in the order of their occurrence in the pipeline. A function can filter and alter some or all of the device data before the subsequent function in the pipeline processes that data.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console:
   a. Open a web browser and browse to the URL of your Oracle Internet of Things Cloud Service instance. Typically, the URL is `https://hostname/ui`.
   b. Enter your user name and password and then click **Sign In**.

2. Click **Menu** () next to the Oracle Internet of Things Cloud Service title on the Management Console
3. Select **Devices** and then select **Model**.
4. Select a user-defined device model in the device model list.

   You cannot add a device policy to a system-defined device model.
5. Click **Device Policies** ().

   The Device policy page appears that can contain one or more policies.
6. Click **Add Policy**.
7. Enter a name for the policy, enter an optional description for the policy, and then click **OK**.
8. Click **Add Policy Pipeline** () in the **Attributes** area.
9. Complete these fields:
   - **Target Attribute**: Select the device attribute to which the policy applies.
   - **Description**: (Optional) Enter a description for the device policy.
   - **Function**: Select one of these options:
     - **Action Condition**: Select this option to invoke an action when the specified condition is true. The device model should have actions defined. in it.

       Select or enter the values for the function
       * **Condition**: Click to insert formula and define an expression in the formula editor. See Use the Formula Editor.
       * **Action**: Select the action to execute when the specified condition is true
       * **Filter**: Enable or disable this option to make the policy behave as a filter policy. Enabling this prevents the attribute from being set to the alerted value, if the condition is met.
     - **Alert Condition**: Select this option to generate an alert when the specified condition is true.

       Select or enter the values for the function
       * **Condition**: Click to insert formula and define an expression in the formula editor. See Use the Formula Editor.
       * **Alert**: Select the specific alert to be raised when the specified condition is true.
       * **Severity**: Select the severity to be assigned to the alert, from low, normal, significant, or critical.
       * **Filter**: Enable or disable this option to make the policy behave as a filter policy. Enabling this prevents the attribute from being set to the alerted value, if the condition is met.
     - **Computed Metric**: Select this option to compute a value from the specified formula.

       Formula: Click to insert formula and define an expression in the formula editor. See Insert Formula for Attribute Functions
     - **Detect Duplicates**: Select this option to generate an alert when a duplicate attribute value occurs within the specified period window.

Select the values for these fields:

* **Window**: Enter a duration in milliseconds for which data is accumulated for applying the function.
* **Alert**: Select the specific alert to be raised when the specified condition is true.
* **Severity**: Select the severity to be assigned to the alert, from low, normal, significant, or critical.

– **Eliminate Duplicates**: Select this option to delete any message containing a duplicate attribute within the specified period.

Window: Enter a duration in milliseconds for which data is accumulated for applying the function.

– **Filter Condition**: Select this option to filter data when the specified condition is true

Condition: Click to insert formula and define an expression in the formula editor. See Use the Formula Editor.

– **Max**, **Mean**, or **Min**: Select one of these functions to capture the maximum, mean, or minimum value for the specified period.

Enter or select values for these fields:

* **Slide**: Enter a value to specify the frequency of evaluating the function
* **Window**: Enter a duration in milliseconds for which data is accumulated for applying the function.

– **Sample Quality**: Select this option to take a data sample at the specified rate.

Condition: Click to insert formula and define an expression in the formula editor. See Use the Formula Editor.

Rate: Enter -1 for random, 0 for none, or a number to specify the rate

– **Standard Deviation**: Select this option to calculate a standard deviation for the specified period.

Enter or select values for these fields:

* **Slide**: Enter a value to specify the frequency of evaluating the function
* **Window**: Enter a duration in milliseconds for which data is accumulated for applying the function.

10. Click **OK**.
11. Click **Device Selection** () and assign the policy to one or more devices.

> **Note:**
>
> You can only assign those devices that have the selected device model. A device can only be assigned one policy at a time. Assigning a device to this policy will remove any other policy that may have been assigned to the device.

12. Click **Save**.

The saved policy is sent to the client side immediately if possible. It depends on how often the clients connect to Oracle IoT Cloud Service. For some clients, the policy is received immediately and for clients that don't connect to the server frequently, may receive the policy later. The clients start processing the updated policy immediately after receiving it.

13. (Optional) Click **Back** to return to the device model list.

Assign a Policy to a Device

Assign a policy to a device to make the device return the data specified in the policy.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** () next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ().
6. Select a device policy.
7. Click **Device Selection Filter** () and then select one of these options:
   - **All**: Assigns the policy to all devices using the selected device model.
   - **Specific**: Assigns the policy to a specific device using the search parameters you specify.

   > **Note:**
   >
   > You can only assign the devices which have the selected device model. A device can only be assigned one policy at a time. Assigning a device to this policy will remove any other policy that may have been assigned to it.

8. Click **OK**.
9. Click **Save**.
10. (Optional) Click **Back** to return to the device model list.

Add a Policy Function

Add a policy function to perform an additional data computation on the device to which the policy is assigned.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** () next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ().

6. Select a device policy.
7. Click **Add Device Function** () in the **Function** area.
8. Select one of these functions in the **Function** list:
   - **Action Condition**: Select this option to invoke an action when the specified condition is true.
   - **Alert Condition**: Select this option to generate an alert when the specified condition is true.
   - **Computed Metric**: Select this option to compute a value from the specified formula.
   - **Detect Duplicates**: Select this option to generate an alert when a duplicate attribute value occurs within the specified period.
   - **Eliminate Duplicates**: Select this option to delete any message containing a duplicate attribute within the specified period.
   - **Filter Condition**: Select this option to filter data when the specified condition is true.
   - **Max**: Select this option to capture the maximum value for the specified period.
   - **Mean**: Select this option to capture the average value for the specified period.
   - **Min**: Select this option to capture the minimum value for the specified period.
   - **Sample Quality**: Select this option to take a data sample at the specified rate.
   - **Standard Deviation**: Select this option to calculate a standard deviation for the specified period.
9. To insert a formula for the **Condition** field of the functions, see Use the Formula Editor.
10. Click **OK**.
11. Click **Save**.
12. (Optional) Click **Back** to return to the device model list.

Add Attributes to a Policy

Add attributes to a policy to request additional data from the device to which the policy is assigned.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** () next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ().
6. Select a device policy.
7. Click **Add Policy Pipeline** () in the **Attributes** area.
8. Complete these fields:

- **Target Attribute**: Select the device attribute to which the policy applies.
- **Description**: (Optional) Enter a description for the device policy.
- **Function**: Select one of these options:
  - **Action Condition**: Select this option to invoke an action when the specified condition is true.
  - **Alert Condition**: Select this option to generate an alert when the specified condition is true.
  - **Computed Metric**: Select this option to compute a value from the specified formula.
  - **Detect Duplicates**: Select this option to generate an alert when a duplicate attribute value occurs within the specified period.
  - **Eliminate Duplicates**: Select this option to delete any message containing a duplicate attribute within the specified period.
  - **Filter Condition**: Select this option to filter data when the specified condition is true.
  - **Max**: Select this option to capture the maximum value for the specified period.
  - **Mean**: Select this option to capture the average value for the specified period.
  - **Min**: Select this option to capture the minimum value for the specified period.
  - **Sample Quality**: Select this option to take a data sample at the specified rate.
  - **Standard Deviation**: Select this option to calculate a standard deviation for the specified period.

To insert a formula for the **Condition** field of the functions, see Use the Formula Editor.

9. Click **OK**.
10. Click **Save**.
11. (Optional) Click **Back** to return to the device model list.

Duplicate a Policy

Duplicate a policy to quickly copy existing policy settings to a new policy.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** ( ) next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ( ).
6. Identify the policy you want to duplicate and click **Duplicate Policy** ( ) for that policy.
7. Enter a name for the policy, enter an optional description for the policy, and then click **OK**.
8. (Optional) Add a policy pipeline.
9. (Optional) Edit the device function for the policy, or add a new device function.
10. Click **Device Selection** ( ) and then select one of these options:

- **All**: Assigns the policy to all devices using the selected device model.
- **Specific**: Assigns the policy to a specific device using the search parameters you specify.

> **Note:**
>
> You can only assign the devices which have the selected device model. A device can only be assigned one policy at a time. Assigning a device to this policy will remove any other policy that may have been assigned to it.

11. Click **Save**.
12. (Optional) Click **Back** to return to the device model list.

Edit Policy Details

Use the **Edit Policy** dialog to change the policy name and description.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** ( ) next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ( ).
6. Select a device policy.
7. Click **Edit** ( ).
8. Enter a name or an optional description for the policy, and then click **OK**.
9. Click **Save**.
10. (Optional) Click **Back** to return to the device model list.

Edit Policy Function Settings

Edit policy function settings to update the settings for the policy. For example, you can change the severity setting or the alert trigger.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** ( ) next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ( ).
6. Select a device policy.
7. Click **Edit** ( ) in the **Function** area.

8. Update the policy function settings.
9. Click **OK**.
10. Click **Save**.
11. (Optional) Click **Back** to return to the device model list.

Increase or Decrease Device Policy Priority

Increase the device policy priority to assign the selected device policy precedence over the device policies that follow it. Decrease a device policy to reduce its precedence over the device policies that precede it.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** () next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ().
6. Select a device policy.
7. Select one of these options:
   - Click **Decrease Priority** () to decrease the priority of the device policy.
   - Click **Increase Priority** () to increase the priority of the device policy.
8. Click **Save**.
9. (Optional) Click **Back** to return to the device model list.

Delete a Policy Function

Delete a policy function when it is no longer required.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** () next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ().
6. Select a device policy.
7. Click **Delete Device Function** () in the **Function** area.
8. Update the policy function settings.
9. Click **Delete**.
10. Click **Save**.
11. (Optional) Click **Back** to return to the device model list.

Delete a Policy

Delete a policy when it is no longer required. When a policy is deleted, the device that used the policy is assigned the next available policy in the order defined on the Device Policies dashboard.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** ( ) next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ( ).
6. Select the policy that you wish to delete and click **Delete** ( ).
7. To confirm, click **Delete**.
8. (Optional) Click **Back** to return to the device model list.

Enable Device Policies

Use the **Disabled** slider button to enable a device policy to which device filters have been added. When required, you can use the same slider button to disable the device policy.

After creating a device policy, you must associate it with one or more devices using the **Device Selection Filter** button. Only after you've applied a device filter to a device policy, you can enable it. Until then, the device policy remains disabled.

1. Sign in to the Oracle Internet of Things Cloud Service Management Console.
2. Click **Menu** ( ) next to Oracle Internet of Things Cloud Service.
3. Select **Devices** and then select **Model**.
4. Select a device model in the device model list.
5. Click **Device Policies** ( ).
6. Select a device policy. Ensure that the policy has a device filter.

   You can apply a device filter by using the **Device Selection Filter** ( ) button.
7. To enable the device policy, click **Disabled** ( ).
8. Click **OK**.

> **Note:**
>
> If two or more device policies are applied and enabled on the same device, then the policy whose device filter includes the particular device at a higher level, takes precedence. If you disable a device policy, the devices that match it's filter may be re-assigned to another policy, which is enabled.

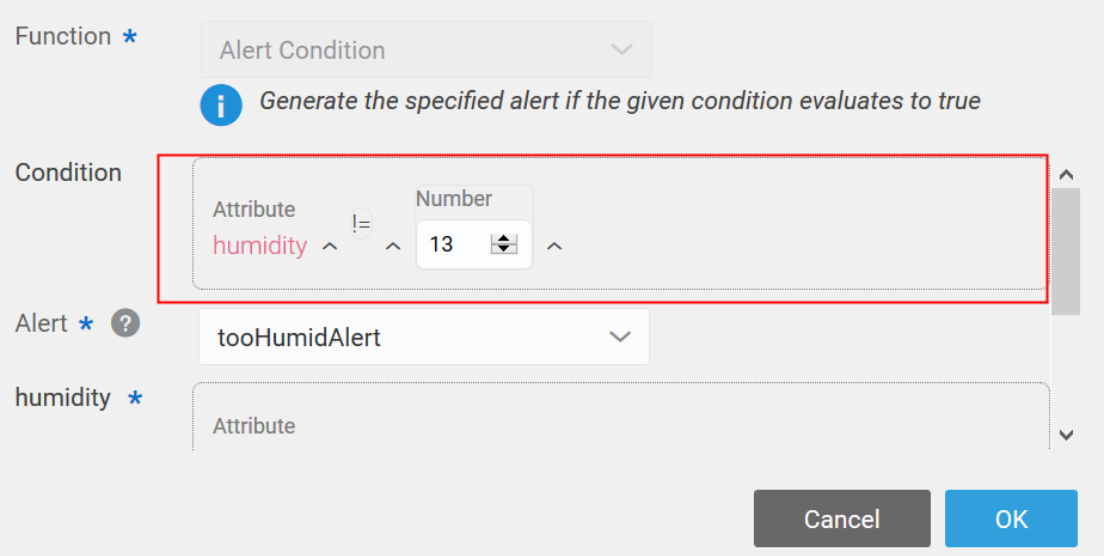
9. Click **Save**.
10. (Optional) Click **Back** to return to the device model list.

Use the Formula Editor

Use the formula editor to define an expression that evaluates a condition for a function of a device attribute. The function becomes a part of the policy pipeline.

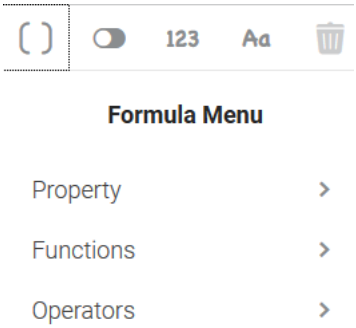
The formula editor appears with a function associated with a device attribute. You can build your expression using the elements of the formula editor.

The following example shows a condition that validates if an attribute value is not equal to a specific number. In this example, the formula editor was used to select humidity as the attribute, != as the operator, and 13 as the numeric value.



Start building your formula by selecting properties, functions or operators from the formula menu.

The image displays the elements of the formula editor explained after the image.



Your expression can contain the following elements:

- **Parenthesis**: Use parenthesis to group operations and indicate precedence.
- **Boolean**: Use boolean to select true or false in the expression.
- **Number**: Use number to enter a numeric value.
- **Text**: Use text to enter alphabetic characters.
- **Property**: A list of attributes that you can use to build your own expression. This list is based on the device model attributes and function that you selected. Click property to select an attribute from a list of attributes.
- **Functions**: Use functions to select the `UPPER` or `LOWER` functions.
- **Operators**: You can use arithmetic (

  `+, -, *, /`

  ), relational (

  `=, <, >, <=, >=, !=`

  ), and logic (

  `AND, OR,`
  `LIKE`

) operators.

5

Manage Devices

After you register a device, you can use the Management Console to locate the device, edit device details, add or remove device metadata, and disable a device.

Topics

- Cloud Service Device Types
- Cloud Service Device Life Cycle
- Locate a Device
- Edit Device Details
- Add and Edit Device Metadata
- Add and Edit Device GPS Coordinates
- View Device Status and Diagnostic Information
- Send a Test Message to a Device
- View the Device Models Assigned to a Device
- View Device Messages
- Manage Alert Messages
- Manage Device Data Messages
- Manage the Device Selection Used in an IoT Application

Cloud Service Device Types

A wide range of devices can be securely connected to Oracle IoT Cloud Service and managed using the Management Console or the REST API.

Each device can be categorized into the three types of devices that can connect to Oracle IoT Cloud Service. The table below describes each type.

Device Type	Description
Gateway Device	A device that communicates with the Oracle IoT Cloud Service on behalf of other devices that are incapable of direct communication with Oracle IoT Cloud Service. It can be a third-party gateway device that has the Oracle IoT Cloud Service Client Software installed on it.
Directly Connected Device	A device that is capable of connecting directly with the Oracle IoT Cloud Service via a running application that is based on Oracle IoT Cloud Service Client Software Libraries (or equivalent third party solution). This device type cannot register any indirectly connected devices. The only supported direct connection protocol is HTTPS over TCP/IP.
Device Connected via Gateway	Device connected to Oracle IoT Cloud Service via a Gateway. It does not require any Oracle IoT Cloud Service software to be installed in it.

Cloud Service Device Life Cycle

Each device in the Oracle IoT Cloud Service network has to go through a life cycle of managed state transitions. The current state a device is in determines its security permissions and the level of participation it's allowed to have in the Oracle IoT Cloud Service network.

You can use either the Oracle IoT Cloud Service Management Console or the Endpoint Repository REST API directly to place a device in a specific state. The following table describes the different states that a device can be in throughout its life cycle.

State	Description	More Info
Registered	This is the initial state of almost all devices. This state indicates that sufficient information about the device has been registered with the Oracle IoT Cloud Service to enable the device to be authenticated when it attempts to register with the service. The existence of the device and certain minimum metadata about the device have been defined in the Endpoint Repository. The device is not yet activated and is not yet allowed to participate in the Oracle IoT Cloud Service network. Devices can be individually registered or registered in batch mode by uploading a CSV file.	Register a Single Device or Register a Batch of Devices
Activated	The device has been authenticated and a trust relationship has been established between the Oracle IoT Cloud Service and the device. An identity for the device and credentials that may be used to authenticate the device have been established and shared between the device and the Oracle IoT Cloud Service. The device can now participate in the Oracle IoT Cloud Service network to the extent allowed by the associated device policy.	Activate a Device or Activating a Batch of Registered Devices
Deactivated	This state allows an administrator of the Oracle IoT Cloud Service to temporarily disable the device from being able to communicate with the Oracle IoT Cloud Service. This state could be used, for example, when a remote patient monitoring device is in transit between one patient and the next. The device can restored to Activated state. Oracle IoT Cloud Service filters out messages received from the disabled devices. A deactivated Gateway stops polling indirectly connected devices and stops forwarding messages to Oracle IoT Cloud Service.	

State	Description	More Info
Decommissioned	This is a permanent state that indicates that the device is no longer trusted and should never be allowed to communicate with the Oracle IoT Cloud Service again. This could be used for a device which is retired or simply because the device is no longer trusted by the administrators of the Oracle IoT Cloud Service. Database records about deleted devices are not physically removed, so that a record of their prior existence, including links to former software configurations, data, alerts, and logs, is preserved. This also ensures that the activation credentials belonging to that device may not be copied by an attacker and used to activate a different device.	

Locate a Device

Use the Device Search on the Management Console to locate a device.

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Select a property in the **Property** list.
4. Enter a value in the **Value** field. Some properties have pre-defined values. To view these values, click inside the **Value** field to view the list of values. Leave the **Value** field blank to return a list of devices that match the selected property.

Edit Device Details

Use the Device Details page to view and edit device information.

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Select a device in the device list and then click the **Edit** (✎) icon.
4. Expand the **Details** tab.
5. (Optional) Edit the **Name**, **Description**, **Manufacturer**, **Serial Number**, and **Model Number** fields.
6. Click **Save**.
7. Click **Back** to return to the device list.

Add and Edit Device Metadata

Add device metadata to make a device easier to locate or to associate it with other devices with similar attributes.

1. Log in to your Oracle Internet of Things Cloud Service instance.

2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Select a device from the table, and click **Edit.**
4. Expand the **Metadata** section.
5. Enter or edit values in the **Classification** and **Location** fields.
6. Click **Add Custom Metadata** to add custom metadata for the device.
7. Enter a **Key** name and **Value** corresponding to the key.

   The **Key** field is pre-populated with key names currently used by other devices or streams. You can choose one of the pre-existing key names, or specify a new key name.

   If you select an existing key name, then the **Type** field is automatically specified. If you are using a new key, then you must choose a data type for **Type.**
8. Click **Save**.

Add and Edit Device GPS Coordinates

Add GPS coordinates to identify the location of a device.

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Expand the **GPS Coordinates** tab.
4. Enter or edit values in the **Latitude**, **Longitude**, and **Altitude** fields.
5. Click **Save**.

View Device Status and Diagnostic Information

View device status and diagnostic information to troubleshoot issues and determine the connection status of a device.

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Expand the **Diagnostics and Remote Configuration** tab.
4. Click **Back** to return to the device list

Send a Test Message to a Device

Send a test message to a device to determine the connectivity status and message delivery latency.

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Expand the **Diagnostics and Remote Configuration** tab.
4. (Optional) Edit the values in the **Count**, **Interval**, and **Size (bytes)** fields.
5. Click **Start** to send a device message.
6. Click **Stop** to stop the message request.

7. Click **Get Status** to get the status of sent messages.
8. Click **Back** to return to the device list

View the Device Models Assigned to a Device

Device model information the device model URN, name, date created, and when it was last modified.

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Expand the **Device Models** tab.
4. (Optional) Edit the values in the **Count**, **Interval**, and **Size (bytes)** fields.
5. Click **Back** to return to the device list

View Device Messages

View device messages to determine the source, format, content, priority, direction, and receipt time of device messages. .

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Management**.
3. Expand the **Messages** tab.
4. (Optional) Select **Auto Refresh Interval** and then select a value to change the auto refresh interval for device messages.
5. (Optional) Select a message format in the messages list to view a list of message data types for the message.
6. Click **Back** to return to the device list

Manage Alert Messages

Alert messages identify issues that require urgent attention, such as error conditions or threshold limits exceeded.

1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Alerts and Messages**.
3. (Optional) Select a date and time interval for the alert messages you want to view and then click **Go**.
4. (Optional) Select **Auto Refresh Interval** and then select a value to change the auto refresh interval for alert messages.
5. (Optional) Click a link in the **Source** column to view details about the device that is the source of the alert.
6. Click **Back** to return to the alerts dashboard.
7. (Optional) Click a link in the **Message Format** column to view the available message data types for the alert message.

8. (Optional) Click a link in the **Description** column to display the details about the alert message.

Manage Device Data Messages

View device messages to determine the source, format, content, priority, direction, and receipt time of device messages.

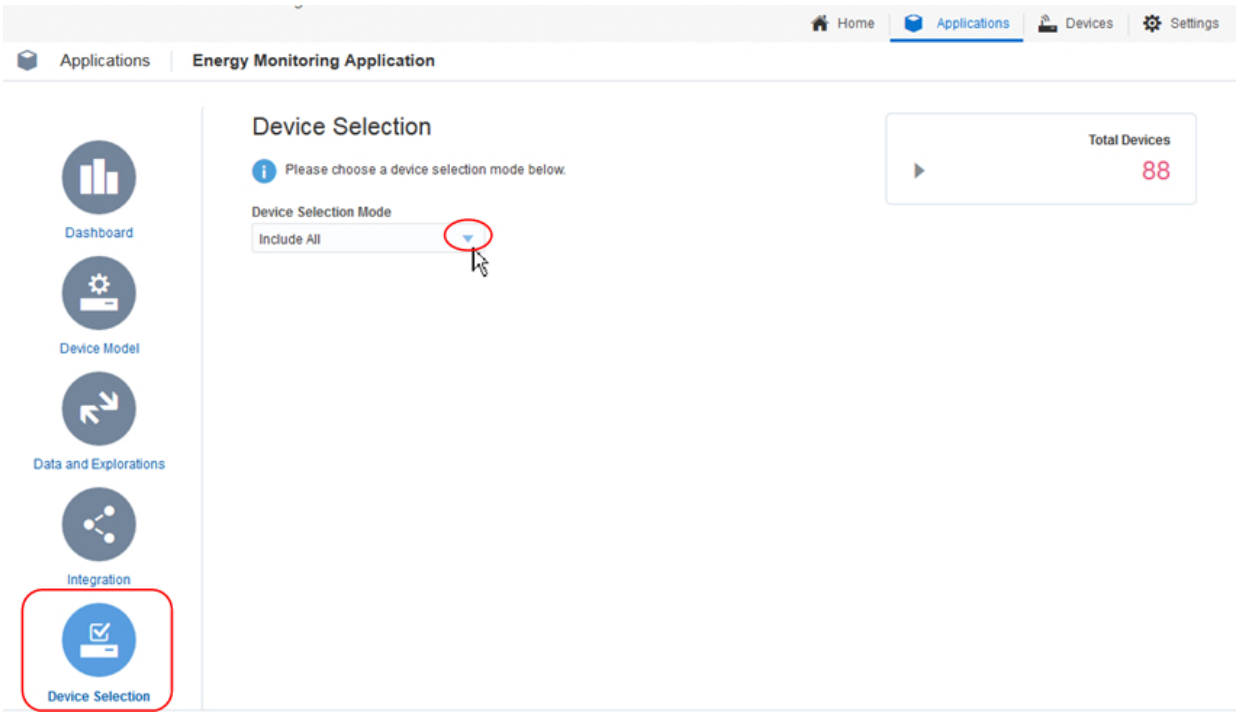
1. Log in to your Oracle Internet of Things Cloud Service instance.
2. Click the **Menu** (☰) icon, click **Devices**, and then click **Alerts and Messages**.
3. Click the **Messages** tab on the right pane.
4. (Optional) Select a date and time interval for the messages you want to view and then click **Go**.
5. (Optional) Click a point on the graph to see the number of messages received at that time.
6. (Optional) Select **Auto Refresh Interval** and then select a value to change the auto refresh interval for messages.
7. (Optional) Click a link in the **Source** column to view details about the device that is the source of the message.
8. Click **Back** to return to the alerts dashboard.
9. (Optional) Click a link in the **Message Format** column to view the available message data types for the message.

Manage the Device Selection Used in an IoT Application

You can filter which devices will be included in the scope of your IoT application by using the Device Selection feature.

By default, all registered devices that support the device models associated with the current IoT Application you are working on are included in the IoT application's scope. You can optionally further refine the device selection used by the IoT application using the following steps:

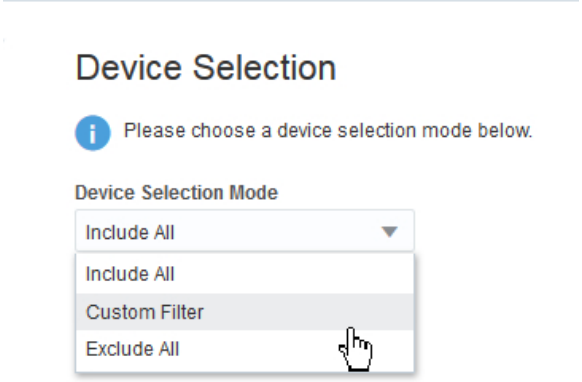
1. From the Management Console, click **Applications**.
2. Select the IoT application you want to work with.
3. Click **Device Selection** on the left navigation area



In the **Device Selection Mode** section, the `Include All` mode is selected by default. On the top right corner of the page, the **Total Devices** value of `88` indicates that there are 88 devices that support the device model or models that's used in the current IoT application you're working with.

4. Click the drop arrow in the **Device Selection Mode** section.

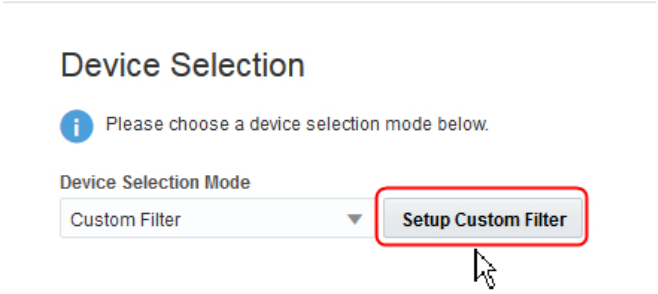
   The image below shows the available modes to choose from:



- **Include All** - This is the default device selection mode. All of the devices that implement at least one of the device models are included in the current IoT application's scope.
- **Custom Filter** - This device selection mode allows you to refine the selection based on
- **Exclude All** - This mode will not include any of the devices that implement the device models included within the scope of the application.

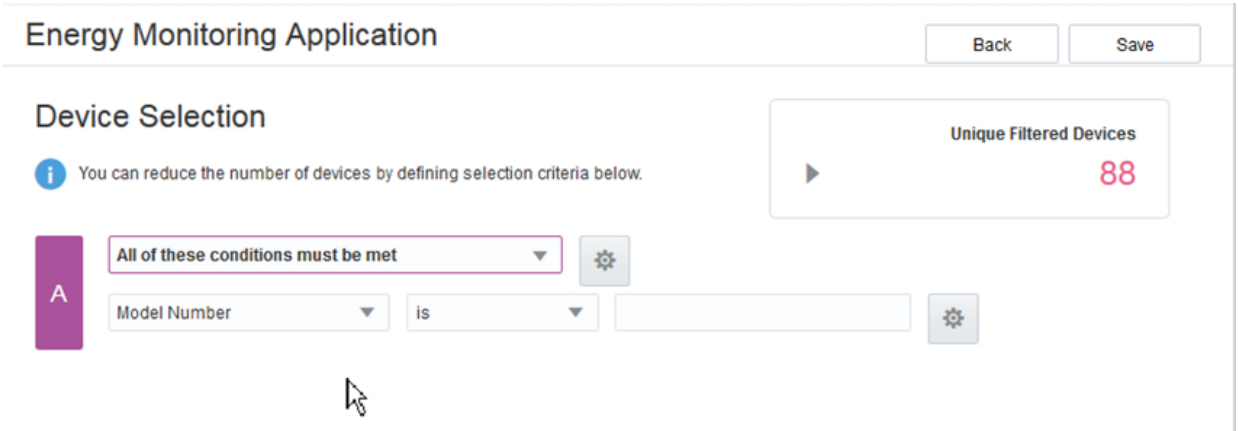
5. From the list of selection modes, choose **Custom Filter**.

   The **Setup Custom Filter** button appears, as shown in the image below.



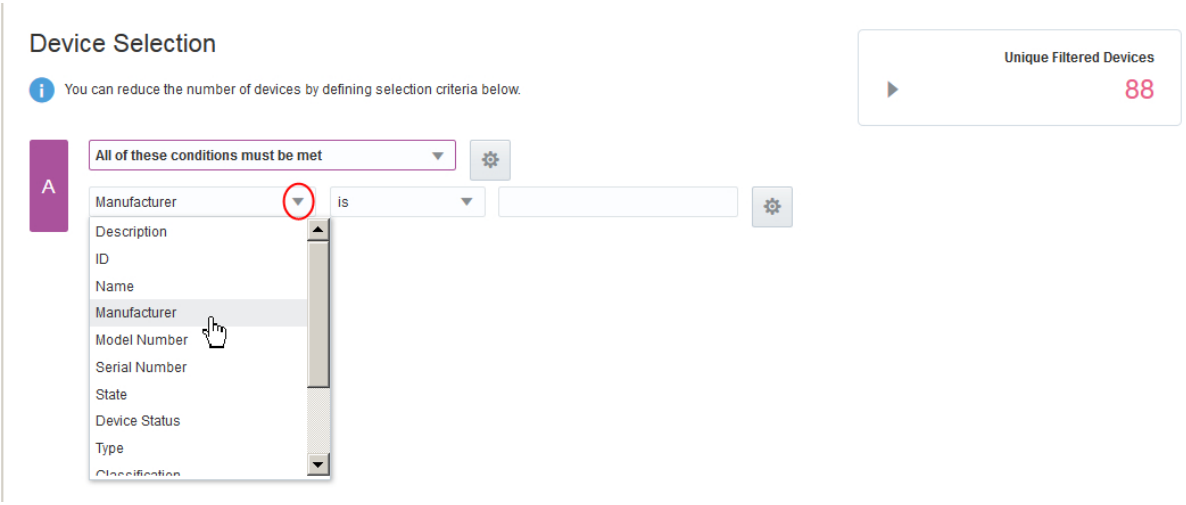
6. Click **Setup Custom Filter**.

   A page similar to the following image is displayed.

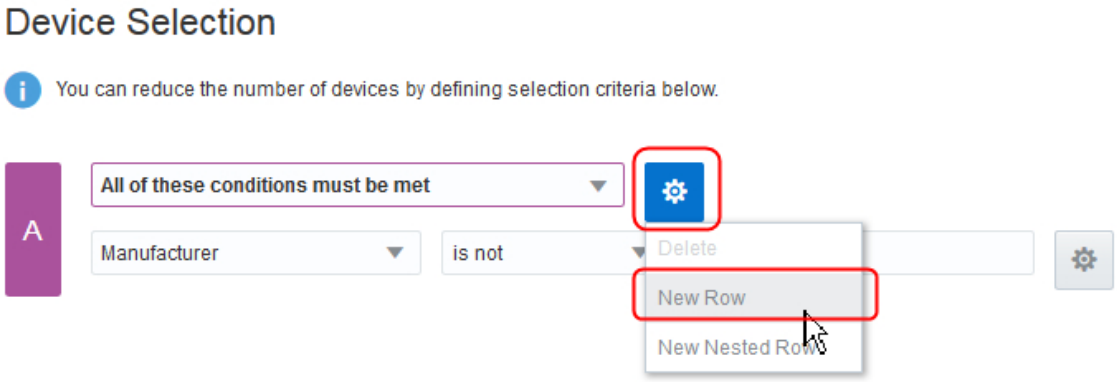


   The initial selection block, labeled A, is shown and the default filter mode is `All of these conditions must be met`. The initial row for the device selection criteria is ready to be configured.

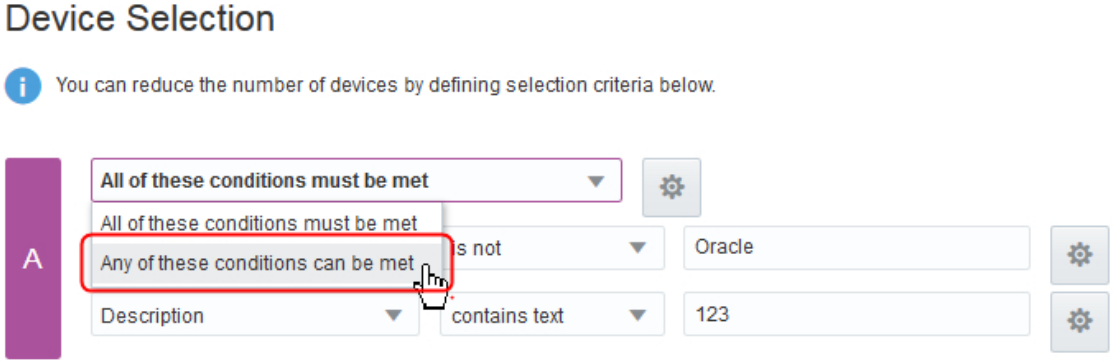
7. Use the drop lists to select the property that you want to use in the device selection criteria, as illustrated by the image below, and enter the value to use for the selection criteria.



8. (Optional) Click the gear icon to add a new row to define an additional criteria to use for device selection.

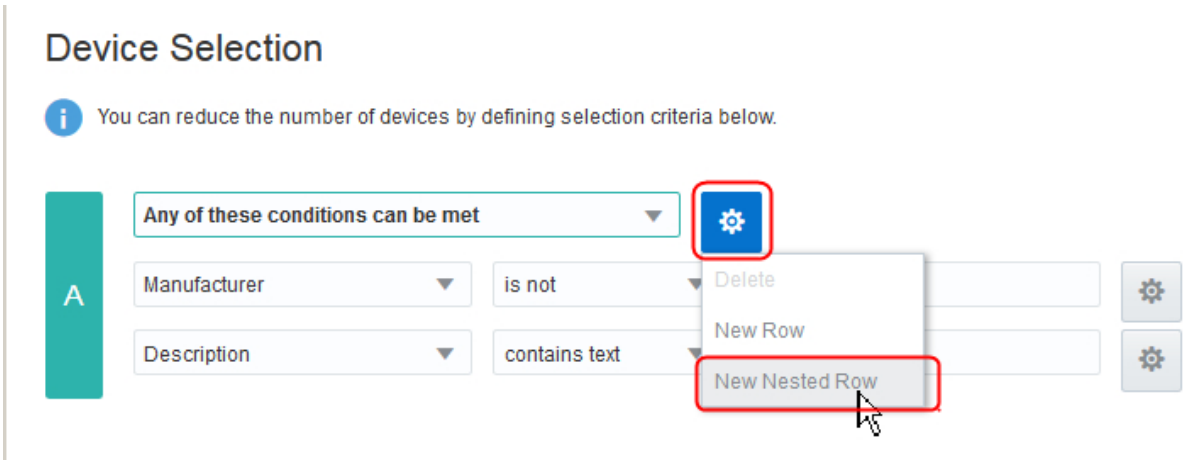


When the **New Row** option is selected, a new row for defining a selection criteria is added to the existing selection block A. When you have more than one row of device selection criteria, you can use the drop arrow to select whether all or any of the conditions specified must be met, as illustrated in the following image.

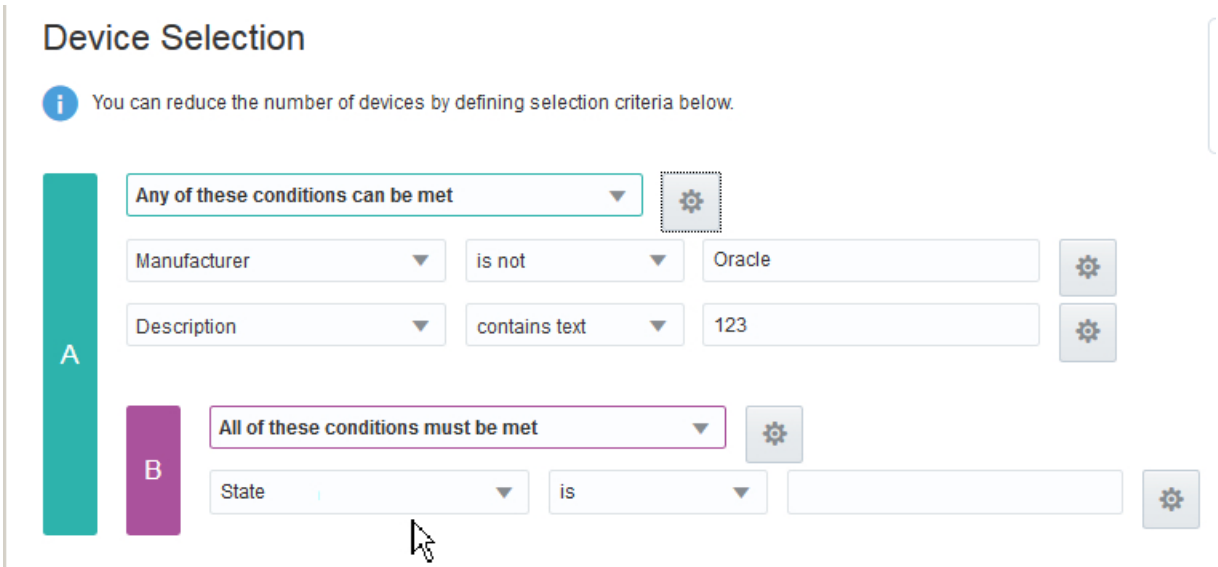


If you choose to keep the `All of these conditions must be met` option, it means that all the selection criteria you define must be met for the device to be included in the scope of the current IoT application. If you choose the `Any of these conditions can be met` option, notice that the selection block A is changed from the color purple to the color green. Also, the selection block A is changed from an “AND” selection block to an “OR” selection block, where any of the conditions defined can be met to determine whether a device should be included in the device selection group.

9. (Optional) Click the gear icon to create a nested group of selection criteria, as shown in the image below.



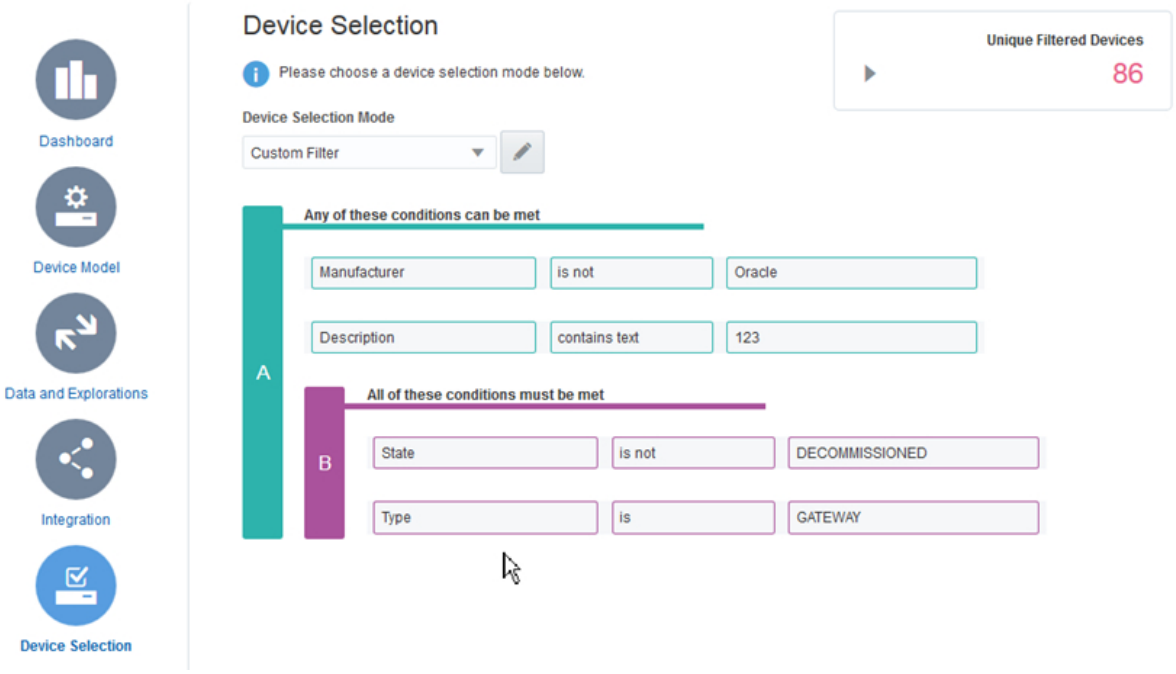
When **New Nested Row** is selected, a new selection block, labeled B, and a new nested row are created, as shown below.



The conditions included in the new selection block B is now also part of the conditions used in the selection block A. Just as in selection block A, you can add a new row or nested row to the selection block B.

10. (Optional) Delete a row in any of the selection blocks by clicking the gear icon for that block and selecting **Delete**.
11. (Optional) Move a row from its current selection block to another selection block by clicking the gear icon and selecting **Move to** and specify to which selection block you want to move the criteria row.
12. Click **Save** to keep all the changes you made to the custom filter, or click **Back** to return to the previous page without saving your changes.

    If you clicked **Back**, the **Confirm Back** dialog is displayed, informing you that your changes will be lost and prompts you whether you are sure you want to go back. Respond `Yes` or `No`. If you clicked **Save**, the Device Selection, similar to the following is displayed.



6

Learn About Oracle IoT Digital Twin

A digital twin is a digital representation of a physical device, asset, or a system. Oracle has implemented the IoT digital twin concept in a comprehensive manner.

Topics:

- About the IoT Digital Twin Framework
- About the Oracle IoT Digital Twin Implementation

About the IoT Digital Twin Framework

A digital twin is the digital proxy of a physical asset or device. A digital twin can help you successfully deploy and use an IoT application.

A digital twin may also be called a twin or a shadow. Digital twin technology may be referred to as device virtualization and can be implemented in differing ways.

About Implementing the Framework

On an IoT platform, a digital twin is a virtual representation of a physical asset, a machine, a vehicle, or a device. It digitally represents the data, processes, operation states, and lifecycle of the asset.

Implementing IoT with digital twin capabilities in a factory, an airport, or a machine plant enables:

- **Better visibility:** You can continually view the operations of the machines or devices, and the status of their interconnected systems.
- **Accurate prediction:** You can retrieve the future state of the machines from the digital twin model by using modeling.
- **What-if analysis:** You can easily interact with the model to simulate unique machine conditions and perform what-if analysis using well-designed interfaces.
- **Documentation and communication:** You can use the digital twin model to help you understand, document, and explain the behavior of a specific machine or a collection of machines.
- **Integration of disparate systems:** You can connect with back-end applications related to supply chain operations such as manufacturing, procurement, warehousing, transportation, or logistics.

The digital twin capabilities of an IoT platform depend on its design and implementation. Typically, you can implement a digital twin framework in two ways:

- **Simple device models:** In this method, you create and use a JSON document that stores the following information about a machine:
  - Name, serial number, and location
  - A set of observed attributes that the machine's sensors observe (for example, the current speed of the machine)

  - A set of desired attributes that the IoT application can set (for example, the desired speed of the machine)

  In this method, you use the attributes of the machines that its sensors capture. This method works best in situations where the sensors may not be continually available, or when communication with the sensors takes place asynchronously.

- **Industrial twins:** This method presents information about the design of a machine and model of a sensor device. The information represents the physics-based properties of the machine.

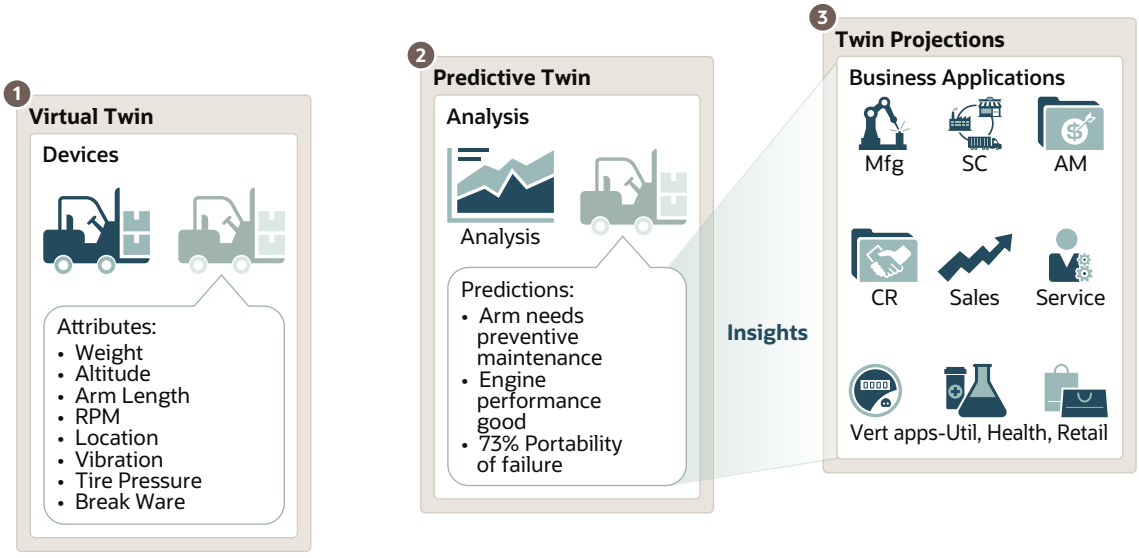
  This method works well with industrial IoT applications that obtain the required information from product lifecycle management (PLM) tools. In this type of implementation, you can represent the physical attributes, design information, and the real-time data of a machine in an asset-versus-model graph.

About the Oracle IoT Digital Twin Implementation

The Oracle IoT digital twin implementation has three pillars, each of which has its own use cases and advantages.

Elements of the Oracle IoT Digital Twin Implementation

The image illustrates the three pillars. The three pillars help solve different kinds of problems and offer different advantages.



Virtual Twin

In a virtual twin, Oracle's device virtualization feature creates a virtual representation of a physical device or an asset in the cloud. A virtual twin uses a JSON-based model that contains observed and desired attribute values and also uses a semantic model.

Applications and Use Cases

- An IoT application may not be able to connect to the physical asset. In such a scenario, the application must be able to retrieve the last known status and to control the operation states of the asset.
- Typically, devices use a variety of protocols and methods to connect to the IoT network. This complexity should not affect other business applications such as an

enterprise resource planning (ERP) application. Device virtualization provides an abstraction layer that enables typical business applications to securely communicate with devices.

Advantages

- The semantic model lets you specify the normal operating range of an attribute. This results in a simpler implementation of edge computing.
- The business rules that you declaratively define on top of the complex event processing (CEP) engine in Oracle IoT Cloud Service can be automatically invoked at the edge of the IoT network.

  Example: Typically, to detect threshold violations of a device attribute such as the temperature, a developer writes a separate application to process this information. The user then has to manage and maintain the application. With device virtualization, the device model detects abnormalities and generates appropriate alerts, eliminating the need to write and deploy a separate edge-computing program.
- The semantic awareness built into the device model optimizes the delivery mechanism and the volume of network traffic.
- Compared to the typical approach of using efficient protocols such as MQTT, a semantic model significantly reduces the cost of network bandwidth by automating statistical modeling at the edge.

  Example: In a fleet of connected automobiles, an IoT application monitors a collection of operating parameters. The semantics-based model enables the edge to:

  – Decide whether the operating parameters are in normal range.

  – Identify whether each message is urgent, important, or routine. The edge can intelligently detect that a break-failure notification is an urgent message, `a low pressure in tire` is an important message, and `oil viscosity is slowing` is a routine message.

  – Determine whether to send an alert message over a cellular network or download the device data when the vehicle connects to a Wi-Fi network at the end of a shift.

Predictive Twin

In a predictive twin, the digital twin implementation builds an analytical or statistical model for prediction by using a machine-learning technique. It need not involve the original designers of the machine. It is different from the physics-based models that are static, complex, do not adapt to a constantly changing environment, and can be created only by the original designers of the machine.

Applications and Use Cases

- When the model detects a future problem or a future state of a machine, then you can prevent or prepare for them.
- You can use the predictive twin model to determine trends and patterns from the contextual machine data. The model helps you address business problems.

Advantages

- A data analyst can easily create a model based on external observation of a machine and can develop multiple models based on the user's needs.
- The model considers the entire business scenario and generates contextual data for analysis and prediction.

- The IoT applications can generate contextual data not only from the machine data but also from the various business applications that are integrated with it. The models created from this data are effective and usable.
- The predictive models vary from being simple to complex based on the business problem that you want to solve.

  Examples

  - Oracle IoT Cloud Service includes the Oracle Stream Explorer, an event-processing engine, and uses it to create a simple model based on trends and patterns in the data.
  - Oracle IoT Cloud Service uses Apache Spark-based analytics for complex models. To generate time-series data, you can use the additional libraries for Apache Spark in Oracle IoT Cloud Service.
  - A data scientist can create a complex model by using Oracle R Advanced Analytics for Hadoop (ORAAH) and then execute it in the IoT data pipeline.
  - A business user can use the user-friendly interface of the Oracle Big Data Discovery product to create simple models.

Twin Projections

In twin projections, the predictions and the insights integrate with back-end business applications, making IoT an integral part of business processes.

For insight projections, Oracle IoT Cloud Service integrates with various products:

- Native pre-built integrations with Oracle ERP and Oracle CX applications
- Integration with 150 applications by using Oracle Integration Cloud Service (ICS)
- Integration using REST API libraries

Applications and Use Cases

- You can generate insights in the IoT platform and integrate them with business processes.
- When projections are integrated with a business process, they can trigger a remedial business workflow.
- Business applications can have access to the transactional and contextual IoT data that they need for their decisions.
- Business applications can monitor the predicted state of the machines and their environment.

Advantage

Prediction data offers insights into the operations of machines. Projecting these insights into the back-end applications infrastructure enables business applications to interact with the IoT system and transform into intelligent systems.

7

Use the Oracle IoT Digital Twin Simulator

The IoT digital twin simulator lets you create simulated devices for your environment without the need to connect and set up hardware. You can generate configurable live data, alerts, and events for these simulated devices.

Using the IoT digital twin simulator, you can create simulation models and then use them to create simulated device instances. The IoT digital twin simulator also provides ready-to-use simulation models that you can use to create simulated devices.

Simulated devices behave like real devices: You can control and monitor them in real time, turn them off and on, and configure them to generate alerts and events.

Use simulated devices to perform tests in your Oracle IoT Cloud Service environment, such as test applications and analytics. You can customize the device attributes, alerts, and actions in your simulation models. You can also add events, images, and data sources to your simulation models.

Topics

- Typical Workflow for Simulating Devices
- Access the IoT Digital Twin Simulator Dashboard
- Create Simulation Models
- Create Simulated Devices
- Use Simulated Devices
- Manage Simulated Devices
- Work with Simulation Models

Typical Workflow for Simulating Devices

Create simulation models using the IoT digital twin simulator. You can then use the simulation models to create simulated device instances.

Simulated devices behave like real devices: You can control and monitor the devices in real time, turn them off and on, and configure the devices to generate alerts and events.

Use simulated devices to perform tests in your Oracle IoT Cloud Service environment. For example, you can use simulated devices to test applications and analytics.

You can customize the device attributes, alerts, and actions in your simulation models. You can also add events, images, and data sources for your simulation models.

The following table provides an overview of tasks that you can perform using the IoT digital twin simulator:

Task	Description	More Information
Access the IoT Digital Twin Simulator Dashboard	The IoT Digital Twin Simulator Dashboard provides an overview of the existing simulation models, and the statuses of the simulated devices associated with these models. Use the Dashboard to work with simulated devices and simulation models.	• Access the IoT Digital Twin Simulator Dashboard – Navigate Back to the Simulations Page from Any Page
Create Simulation Models	Simulated devices are based on simulation models. Create a simulation model before you can create a simulated device. You can customize the device attributes, alerts, and actions in your simulation models.	• Create Simulation Models – Add a Predefined Simulation Model in Oracle IoT Digital Twin Simulator – Create a Custom Simulation Model – Import a Simulation Model
Create Simulated Devices	Simulated devices are live instances of simulation models. You can create directly connected simulated devices, or indirectly connected simulated devices that connect through a simulated gateway.	• Create Simulated Devices – Create a Simulated Gateway Device – Create a Simulated Device in IoT Device Simulator – Create a Simulated Device for a Registered Device
Work with Simulated Devices	You can start, stop, and monitor simulated devices like real devices. You can also generate events and alerts supported by the simulation model for your device.	• Use Simulated Devices • Manage Simulated Devices – View a Simulated Device – Delete a Simulated Device
Work with Simulation Models	You can edit a simulation model after creation. You can customize attribute values, alerts, and actions. You may also edit data sources, expressions, and functions. You can choose to export a simulation model to a zip file.	• Work with Simulation Models – Edit a Simulation Model – Use Expressions and Functions in Simulation Models – Export a Simulation Model – Delete a Simulation Model

Access the IoT Digital Twin Simulator Dashboard

The IoT digital twin simulator dashboard provides an overview of the existing device models, and the statuses of the simulated devices associated with these models. Use the dashboard to work with simulated devices and device models.

To access the IoT device simulator dashboard:

1. Navigate to the following URL:

   `https://hostname/ds`

   Here, *hostname* is the host name of your Oracle IoT Cloud Service instance.

2. Enter your Oracle IoT Cloud Service *username* and *password* to log in to the IoT digital twin simulator.

   On the **Simulations** page, to create a simulation model, you may proceed to Create Simulation Models.

Navigate Back to the Simulations Page from Any Page

You can navigate directly from any page in IoT digital twin simulator back to the simulations page.

1. Click **Menu** adjacent to the Oracle IoT Digital Twin Simulator title on the page.

   If you are currently in a device view, or editing a simulation model, you would need to close the page before you can access the menu.

2. Click **Simulations** in the left pane.

Create Simulation Models

Simulated devices are based on simulation models. Create a simulation model before you can create a simulated device in Oracle IoT Digital Twin Simulator.

A simulation model defines how the device is simulated. Simulation models specify the following:

- Device model to use.
- Device attributes and their values.
- Alerts that are generated.
- Device actions.

Simulation models may additionally define events that simulate the various states for the device. You can trigger these events for the simulated device when the simulation is in progress. For example: You may define an event to power on/off the device.

The IoT digital twin simulator includes some predefined simulation models that you can add and use. See Add a Predefined Simulation Model in Oracle IoT Digital Twin Simulator for more information. You can also create custom simulation models that use a device model from your Oracle IoT Cloud Service instance. See Create a Custom Simulation Model for more information. If you have previously exported a simulation model, you can import this into the IoT Digital Twin Simulator. See Import a Simulation Model for more information.

Add a Predefined Simulation Model in Oracle IoT Digital Twin Simulator

Oracle IoT Digital Twin Simulator includes ready-to-use, predefined simulation models that you can add and use.

1. In the Simulations page, click **Add** .
2. Select one of the predefined simulation models to add.

   The available predefined models are **Blue Pump**, **HVAC**, and **Truck**.

The new simulation model appears on the simulations page. You can next create devices using this model.

Create a Custom Simulation Model

Custom simulation models let you customize the simulation of a device model. You can customize attribute values, alerts, and actions. You can also use events, data sources, expressions, and functions.

1. On the Simulations page, click **Add** +.
2. On the **Select an option** page, select **Custom.**
3. In the General Settings section, provide the following values:
   - **Name:** Name for the new simulation model. For example: `My Sensor`.
   - **Messaging Interval:** Determines how frequently the simulator sends data messages. The default interval is every `1000` ms.
   - **Message Limit:** The maximum number of messages a simulator device can send during a simulation. The default value is `0`, which indicates that there is no limit. The simulator sends messages for as long as the simulated device is running.
4. In the Device Models section, select a device model for **Device Model #1.**

   The simulator model gets the attributes, alerts, and actions from the selected device model. You can add multiple device models.
5. (Optional) Under Attributes, customize the attribute simulation for the attributes defined in the selected device model.

   You can customize the following fields:
   - **Interactive:** Select if you want the attribute to appear in the simulated device view attribute list.
   - **Display Name:** Specify the display label to use for the attribute.
   - **Chartable:** Select if you want the attribute to appear in the simulated device view data chart.
   - **Units:** Specify a measurement unit for the attribute. For example: Celsius (C) for temperature.

     Leave the field empty if your attribute does not need a measurement unit.
   - **Format:** Specify the data format for numeric attributes.

     For example: The format `#.####` uses four decimal points. The format `###` uses three integer digits.
   - **Initial Value Expression:** Specify an expression to initialize the attribute when the simulation starts. The expression can contain constants, values of other attributes, and functions.

     For example:
     - You can use the `now()` function to initialize the `startTime` attribute.
     - You can use `$temp` to initialize the value of `mintemp` with the value of the `temp` attribute.
     - You can use the numeric value `37.391838` to initialize the `latitude` attribute.

- **Value Expression:** Specify an expression to evaluate the value of the attribute at any given point of time. The expression can contain constants, values of other attributes, and functions.

  For example: You can use `Math.max($maxTemp, $temp)` to calculate the `maxTemp` attribute. Here, `max` is a function of the `java.lang.Math` class in Java, and `$maxTemp` and `$temp` are attributes.

6. (Optional) If your device model defines alerts, you can choose to customize the following fields in the **Alerts** section:
   - **Interactive:** Select if you want to trigger the alert interactively using a button. Deselect if the alert should be triggered based on a condition.
   - **Display Name:** Specify the label to use for the interactive alert button. **Display Name** is enabled only for interactive alerts.
   - **Condition Expression:** Specify a boolean expression, which defines the condition for triggering the alert. **Condition Expression** is enabled only for non-interactive alerts.

     For example: `$temp >= $maxThreshold`.
   - **Image:** Choose an image to display after the alert is triggered. You can upload the new image in the Images section.
   - **Image Time:** Specify the time duration, in milliseconds (ms), for which the image is displayed. After the time elapses, the image reverts back to the image corresponding to the current state of the device.
   - **Expression:** If your device model defines alert fields, then you can specify a value expression for each field.
7. (Optional) If your device model defines actions, then in the Actions section, click **Add Attribute** and **Add Event** to add any additional attributes and events for an action.
   - **Attribute Expression:** Select the name of an available attribute and specify an expression to set the attribute value.

     For example, you can set the `startTime` attribute to the following expression: `$action.power ? now() : $startTime`. Here, `power` is an action.
   - **Event Expression:** Select the name of an available event and specify a boolean expression to set the event value. A true value turns on the event. A false value turns off the event.
8. (Optional) In the Events section, click **Add New Event** to add one or more events for your simulation model.

   Use events to change the state or mode of your simulator. For example: An event to power off/on the device. You can configure the following fields for each event:
   - **Name:** Specify a unique identifier for the event.
   - **Description:** Specify a label for the event switch.
   - **Image:** Specify an image to display when the event is on.
   - **Add Attribute:** Click to add an attribute.
   - **Attribute Expression:** Select an available attribute and specify the expression used to set the attribute value when the event is on.

     For example: You can set the `$temp` attribute to `sinInRange(60, 80)`. You can add multiple attributes for each event.

9. (Optional) In the Images section, click **Add New Image** to **Upload** new image files to be used for events and alerts.

   You can also modify the default images for `isOnImage` and `isOffImage`. These default images represent the running and stopped states of the device simulator.

   The following media types are supported:

   - JPG
   - GIF
   - PNG
   - Animated GIF
   - MP4

10. (Optional) In the Data Sources section, click **Add Data Source** to add a data source for attribute simulation.

    You can use a CSV (comma-separated value) file as the source for attribute data simulation. You can configure the following fields for a data source:

    - **Name:** Specify a unique identifier for the data source.
    - **Loop:** Select if you want the simulator to continue reading from the beginning of the CSV file after reaching the end.
    - **Upload File:** Click to select the CSV file to upload.

      Ensure that the CSV file does not contain any column heading names. For example, if you have a *`Temperature`* column containing temperature values, remove the header row title, so that the rows contain data values only.

    You may use the data source values in expressions. For example, to read values from the second column of a data source called `dsName`, use the following syntax: `$datasource.dsName[1]`. Here, the name of the data source is `dsName`, and the index value of 1 specifies the second column in the data source.

11. (Optional) In the Functions section, click **Add Function** to define your own function.

    You can use the function at any place where expressions are allowed. For example, use functions in expressions for attributes, alerts, and events.

    Specify values for the following fields:

    - **Name:** Specify a name for the function. For example: `absDiff`.
    - **Definition:** Use lambda syntax to define the body of the function. For example: `(param1, param2) -> Math.abs(param2 - param1)`

    You can now use the function in expressions. For example: `$attr0 > 50 ? absDiff($attr1, $attr2) : absDiff($attr3, $attr4)`.

12. Click **Save** to save your custom simulation model.

Import a Simulation Model

You can import a previously exported simulation model and add it to the dashboard page in IoT digital twin simulator.

1. On the Simulations page, click **Add** +.

2. On the **Select an Option** page, Select **Import.**
3. Choose the simulation model zip file to upload from your local disk.

The imported simulation model appears on the simulations page.

Create Simulated Devices

Simulated devices are live instances of simulation models. You can create directly connected simulated devices, or indirectly connected simulated devices that connect through a simulated gateway.

See Create a Simulated Gateway Device and Create a Simulated Device in IoT Device Simulator for more information on creating the gateway and devices.

When you create a simulated device in the IoT digital twin simulator, the device is automatically registered and activated for your Oracle IoT Cloud Service instance. Alternatively, if you have already registered a device in Oracle IoT Cloud Service, then you can create a simulated instance of the registered device using Oracle IoT Digital Twin Simulator. See Create a Simulated Device for a Registered Device for more information.

Create a Simulated Device in IoT Device Simulator

Use a simulation model to create simulated devices. When you create a simulated device, the device automatically registers and activates itself in Oracle IoT Cloud Service.

If you are creating an indirectly-connected simulated device, then you must create the gateway first. See Create a Simulated Gateway Device for more information.

1. On the Simulations page, click a simulation model name.
2. Click **Add** + to add a new simulated device for the simulation model.
3. Select **Register New Device.**
4. (Optional) If you are creating an indirectly-connected device, select **Connected via Gateway** and choose the **Gateway.**
5. (Optional) In the Customize Device section, enter values for the following fields:
   - **Name:** Specify a name for the simulated device.
   - **Description:** Specify a description for the simulated device.
   - **Manufacturer:** Enter a manufacturer name for the simulated device.
   - **Model Number:** Enter a model number for the simulated device.
   - **Serial Number:** Enter a serial number for the simulated device.

   If you choose not to specify these values, then the device is created with default values.
6. (Optional) In the Device GPS Coordinates section, add the following GPS location related information:
   - **Latitude**
   - **Longitude**
   - **Altitude**
7. Click **Create Device.**

The device is registered and activated in Oracle IoT Cloud Service.

The device page appears in the IoT digital twin simulator. You can control the simulated device from this page.

Create a Simulated Device for a Registered Device

If you have already registered a device in Oracle IoT Cloud Service, you can choose to create and activate the corresponding device in the IoT digital twin simulator.

Note the Device ID for the device registered in Oracle IoT Cloud Service. You would need to provide this ID when creating the simulated device in IoT digital twin simulator. To find the Device ID, open the Devices page in Oracle IoT Cloud Service, and click **Management.**

1. On the **Simulations** page, click the simulation model name.

   The simulation model must correspond with the device that you have registered in Oracle IoT Cloud Service.

2. Click **Add** + to add a new simulated device for the simulation model.
3. Deselect **Register New Device,** as you have already registered the device with Oracle IoT Cloud Service.
4. Enter the **Device ID** of the registered device that you noted from Oracle IoT Cloud Service.
5. Click **Add Device.**

The device is activated, and the device page appears in IoT digital twin simulator. You can now control the device from the device page.

Create a Simulated Gateway Device

You can create a simulated gateway device using the IoT digital twin simulator. You can then use this gateway to link to indirectly connected simulated devices.

1. On the Simulations page, click **Gateway**.
2. Click **Add +** to add a new simulated gateway device.
3. Select **Register New Device.**
4. (Optional) In the Customize Device section, enter values for the following fields:
   - **Name:** Specify a name for the gateway.
   - **Description:** Specify a description for the gateway device.
   - **Manufacturer:** Enter a manufacturer name for the gateway.
   - **Model Number:** Enter a mode number for the gateway.
   - **Serial Number:** Enter a serial number for the gateway.

   If you choose not to specify these values, then the gateway is created with default values.
5. (Optional) In the Device GPS Coordinates section, add the following GPS location related information:
   - **Latitude**
   - **Longitude**

- **Altitude**

6. Click **Create Device.**

You can now create indirectly-connected simulated devices that connect through this gateway.

Use Simulated Devices

You can monitor your simulated devices in real time. You can start or stop a device, and trigger events and alerts supported by the simulation model for your device.

1. On the Simulations page, click the appropriate simulation model name.

   The list of simulation devices available for the simulation model appears.

2. Click **View**  against the appropriate device row. Alternatively, click the Device ID in the first column.

   The simulator device page enables you to start the device, stop the device, and trigger device-specific events and alerts. Real-time device metrics also appear on the page.

3. Use the device-specific options to start the device, stop the device, or trigger events and alerts.

4. To view a graphical representation of real-time attribute values for the device, click **Data Chart.**

5. To close the device page, click **Close X**.

Manage Simulated Devices

Use the simulation model page to start and stop your simulated devices, or change the event states associated with one or more devices. You can also delete a simulated device from this page.

1. On the Simulations page, click the appropriate simulation model name.

   The list of simulation devices available for the simulation model appears. The top of the page contains metrics like the number of running devices and their event states.

2. Select **Running** to run the simulation, or deselect to stop the simulation.

   When you are on a device page, the **Running** option appears as the **Status** (On/Off) option.

3. Select/deselect an event against a device row to change the event state of the device.

   For example, a Blue Pump device lets you change the powered state of the device by toggling the **Power off** event. An HVAC device lets you trigger the **Motor Failure** event.

View a Simulated Device

You can open a simulation device from the simulation model page for the device.

1. On the Simulations page, click the appropriate simulation model name.

   The list of simulation devices available for the simulation model appears.

2. Click **View** against the appropriate device row. Alternatively, click the Device ID in the first column.

   You can control the device from the device page.

3. Click **Close X** to close the device page.

Delete a Simulated Device

Delete a simulated device from the simulation model page when it is no longer required.

Deleting a simulated device also decommissions the device in Oracle IoT Cloud Service.

If you delete a simulated gateway device, then all simulated devices that are indirectly connected through this gateway are also deleted.

1. On the Simulations page, click the simulation model name.
2. Click **Delete** against the appropriate device row.
3. Click **OK** .

Work with Simulation Models

Simulation models help you model your simulated device using one of the device models in Oracle IoT Cloud Service. You can further customize the simulation model to suit your needs.

You can edit a simulation model after creation. See Edit a Simulation Model for more information.

You can customize attribute values, alerts, and actions. You may also edit data sources, expressions, and functions. See Use Expressions and Functions in Simulation Models for more information.

You can also export a simulation model to a zip file. See Export a Simulation Modelfor more information.

Edit a Simulation Model

Edit a simulation model if you wish to customize attribute values, alerts, or actions. You may also edit data sources, expressions, and functions.

1. On the Simulations page, click **Edit** next to the simulation model name.
2. On the simulation model settings page, edit the fields that you want to change.

   Create a Custom Simulation Model provides detailed information on the settings that appear for a simulation model.

3. Click **Save** to save your changes.

Use Expressions and Functions in Simulation Models

Use expressions and functions to customize the attributes, alerts, actions, and events for your simulation model.

The IoT digital twin simulator supports the Expression Language 3.0 syntax without curly braces around expressions. You can use all static functions from the JDK 8 `java.lang.Math` class. You can also make use of the additional simulation functions that are provided.

The following are some examples of simulation model fields where you can use expressions and functions:

- **In Attributes:** Use expressions and functions to specify the **Initial Value Expression** and **Value Expression** for attributes. For example:
  - You can use the `now()` function to initialize the `startTime` attribute.
  - You can use `$temp` to initialize the value of `mintemp` with the value of the `temp` attribute.
  - You can use the numeric value `37.391838` to initialize the `latitude` attribute.
  - You can use `Math.max($maxTemp, $temp)` to calculate the value for the `maxTemp` attribute. Here, `max` is a function of the `Math` class, and `$maxTemp` and `$temp` are attributes.
- **In Alerts:** Use a boolean expression to define the **Condition Expression** used for triggering the alert. For example: `$temp >= $maxThreshold`.
- **In Actions:** Use expressions to set attribute and event values for actions. For example, you can set the `startTime` attribute to the following expression: `$action.power ? now() : $startTime`. Here, `power` is an action.
- **In Events:** Use expressions to set the **Attribute Expression** when the event is on. For example: Set the `$temp` attribute to `sinInRange(60, 80)`.
- **In Functions:** Create your own functions using expressions and existing functions. Use the lambda syntax to define the body of a function.

  For example, you can define a user function called `absDiff` as `(param1, param2) -> Math.abs(param2 - param1)`.

  You can now use the function anywhere expressions are allowed. For example: `$attr0 > 50 ? absDiff($attr1, $attr2) : absDiff($attr3, $attr4)`.

> **Note:**
>
> You can refer to device model attributes using the `$` prefix.
>
> When using a single device model, you may use names such as `$temperature`, where `temperature` is the name of a device model attribute. If you have several device models, then outside a device model, such as in an events attributes expression, use fully qualified attribute names. For example: `$urn:com:oracle:iot:device:temperature_sensor.temperature`. If you do not qualify the attribute name, then the first device model is assumed, by default.

Additional Simulation Functions

Oracle IoT Digital Twin Simulator provides the following functions in addition to the static functions available in the JDK 8 `java.lang.Math` class:

- `public static double` **`linearToTarget`**`(double` *`currentValue`*`, double` *`targetValue`*`, double` *`increment`*`)`

  Returns values that follow a linear pattern from *`currentValue`* to *`targetValue`* with the supplied *`increment`*.

- `static double` **`logToTarget`**`(double` *`currentValue`*`, double` *`targetValue`*`)`

  Returns values that follow a logarithmic pattern from *`currentValue`* to *`targetValue`*.

- `static long` **`now`**`()`

  Generates the current time in milliseconds.

- `static double` **`randomInRange`**`(double` *`lowerValue`*`, double` *`upperValue`*`)`

  Generates random float values in the range defined by the arguments.

- `static double` **`sinInRange`**`(double` *`lowerValue`*`, double` *`upperValue`*`)`

  Generates values that follow the `sin(x)` function pattern in the range defined by the arguments.

- `static double` **`spikesInRange`**`(double` *`lowerValue`*`, double` *`upperValue`*`, double` *`currentValue`*`)`

  Generates one of two values, passed as *`lowerValue`* and *`upperValue`*, different from the current value, which is passed as the *`currentValue`* argument.

Export a Simulation Model

You can export a simulation model to a zip file. You can later import this file into any IoT Digital Twin Simulator instance.

To export a simulation model:

1. On the Simulations page, click the simulation model name.
2. On the simulation model page, click **Export** next to the simulation model name.
3. Save the export zip file on your disk.

Delete a Simulation Model

Deleting a simulation model also deletes any simulated devices that use the model.

1. On the Simulations page, click the simulation model name.
2. Click **Delete**.
3. Click **OK**.

8

Create and Manage Connectors

Learn about creating and managing connectors.

Topics:

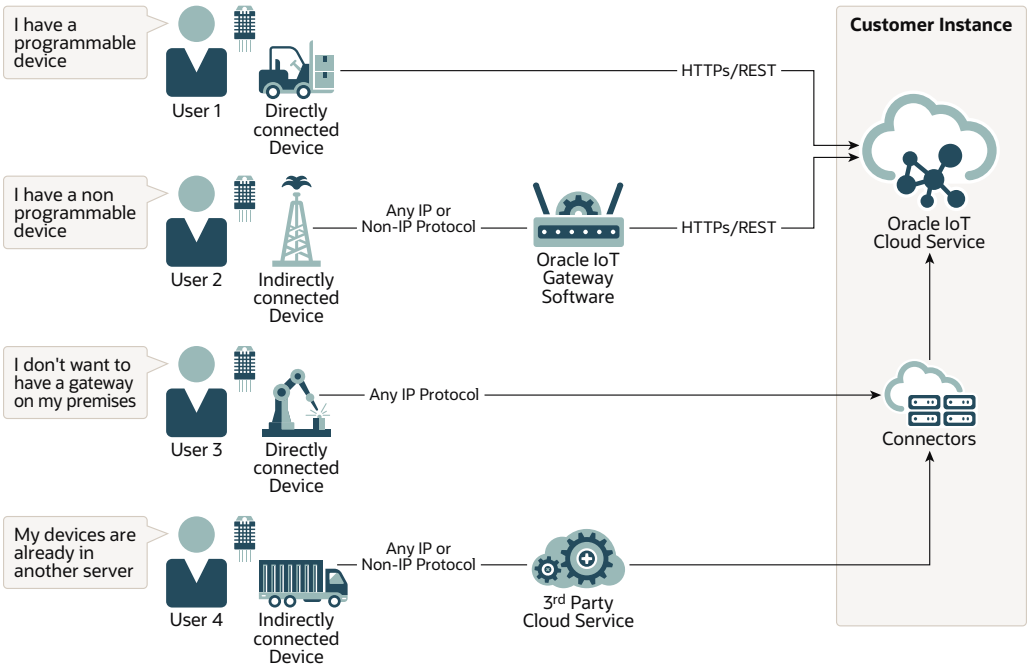
- About Connectors
- Use Cases for the Connectors
- Typical Workflow for Connecting Non-IP Devices
- Create and Configure Connectors
- About Interpreters
- Configure Interpreters
- Learn About Adding Grammars

About Connectors

Connectors provides connectivity from Oracle Internet of Things Cloud Service to devices that are part of a third-party cloud service, follow a proprietary protocol, or have a network provider that uses a long range (LoRa) network for wireless communication.

Typically, you can choose one of four methods to connect a device to Oracle Internet of Things Cloud Service.

The image illustrates the four types of device connectivity.



A connector enables IP devices to directly connect to Oracle Internet of Things Cloud Service without using an Oracle IoT Client Software Library. It also lets non-IP devices indirectly connect to Oracle Internet of Things Cloud Service through a third-party cloud service offered by the network provider.

Features and Functionality

A connector lies in between an Oracle Internet of Things Cloud Service instance and an external system such as a third-party cloud service that manages different devices.

It can perform the following tasks:

- Obtain metadata and telemetry data from devices
- Map metadata with a device model and transform proprietary message payloads to a format compatible with Oracle Internet of Things Cloud Service
- Register device types and activate devices that need to be integrated with Oracle Internet of Things Cloud Service

Connectors support HTTPS and MQTTS protocols. It has an HTTP server connector that accepts `POST` REST operations to receive device messages. It also has a MQTT client connector that subscribes for device messages from an MQTT broker. Both HTTP and MQTT based connectors may use HTTP to query additional details about devices from a remote service.

Connectors provides the following features:

- **A core set of connectivity functionality**: The functionality can identify devices, register or on-board them, receive messages from the devices, and also manage the interpreter.
- **A consistent runtime environment**: It instantiates the connector and provides wrapper services within the environment.
- **The ability to start, stop, scale, and manage the runtime**: It enables users to start and stop the connectors from the standard Oracle Internet of Things Cloud Service user interface. Because connectors can effectively scale the runtime

environment, users can create and run multiple instances of connectors simultaneously.

- **The ability to query user for configuration information** : The user is prompted to enter the required configuration information to run the connector.
- **The ability to query user for input or action at runtime**: Connectors allow to query the user for input. For example, during machine type on-boarding, it prompts the user to input or select the device type fields. This allows the connectors to include the specified fields when the device type is on-boarded.

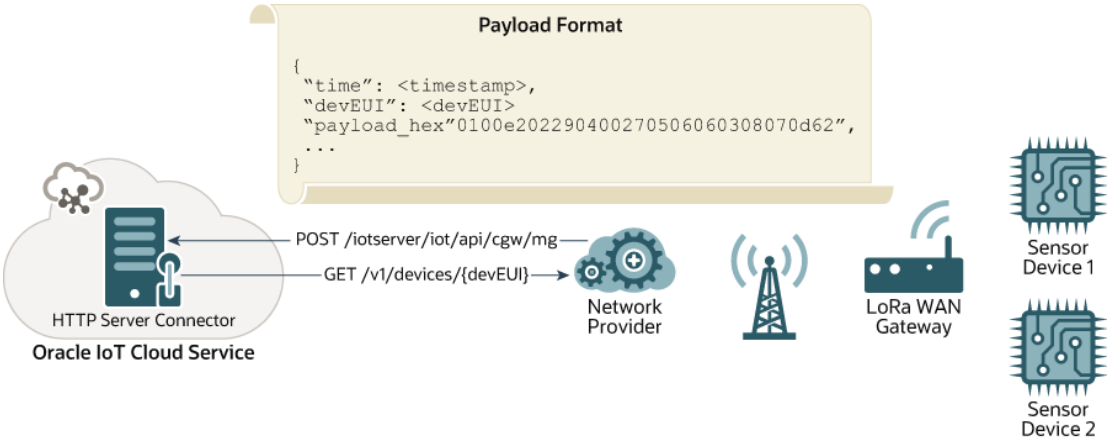
Use Cases for the Connectors

Connectors allow Oracle Internet of Things Cloud Service to accept data from devices that follow different protocols and formats or devices that are connected to third-party cloud services.

The connectors integrate devices with Oracle Internet of Things Cloud Service in one or more scenarios. A connector can be an HTTP server that receives POST messages from an external system, an MQTT client that subscribes to an MQTT broker and then indirectly receives data from devices that are in a remote network, or an MQTT Server connector that does not require a MQTT broker and communicates directly with remote and constrained network devices.

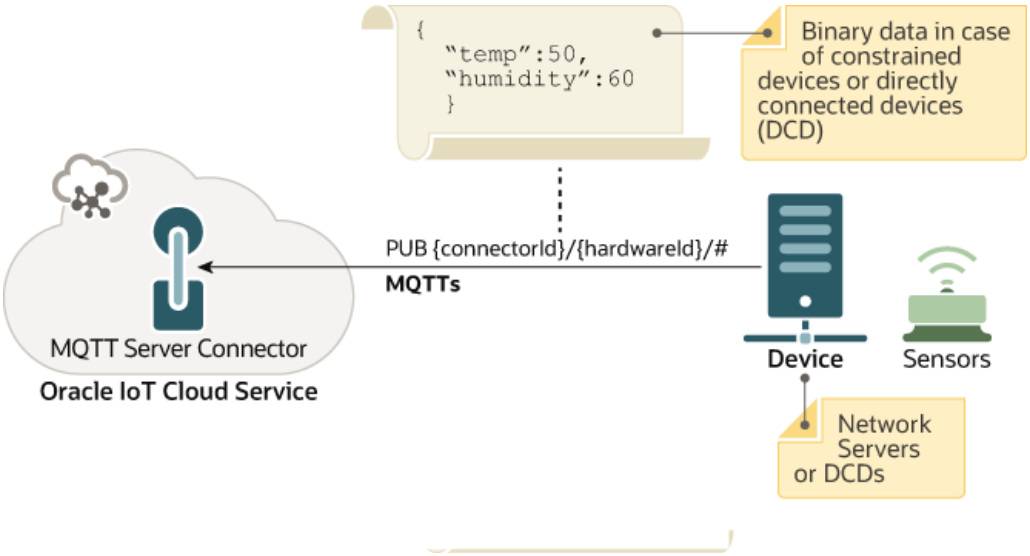
To receive data from devices over MQTT you use an MQTT client connector if you already have data in an MQTT broker that is reachable from the cloud otherwise use an MQTT server connector that can directly receive device telemetry.

The image displays an HTTP Server Connector that integrates with a third-party network provider.



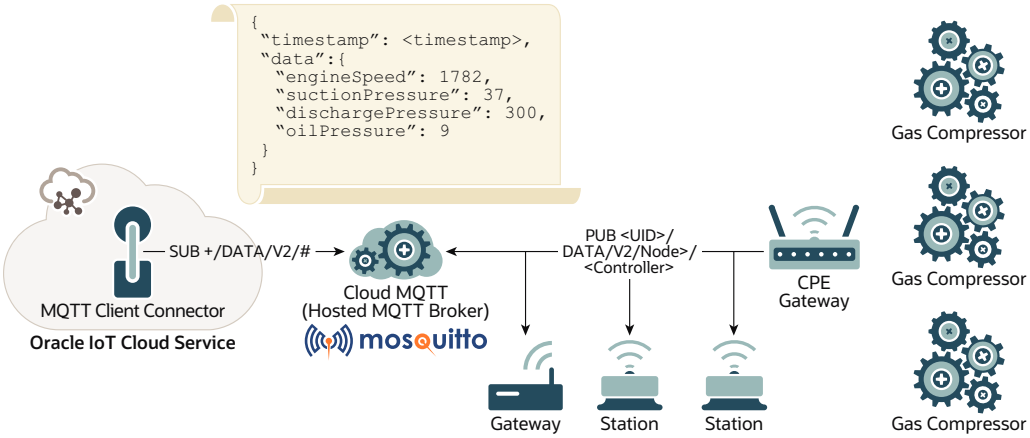
The third-party network provider acts as a sever to manage a network of devices and collects the data from devices in a long range (LoRa) network. The server sends the device data to a connector that processes and analyzes the device data. The HTTP Server Connector provides an endpoint that the third-party network provider can forward the device data to. The connector processes, maps, and transforms the metadata and telemetry data, and then sends it to Oracle Internet of Things Cloud Service. This integration is seamless and more efficient that a model where the connector needs to get the device data from an URL provided by the network provider.

The image displays an MQTT Server Connector that integrates directly with a remote network server or a device in a constrained network that sends data over MQTT.



In this use case, Oracle Internet of Things Cloud Service has an embedded MQTT Server to which a client device can directly publishes telemetry data.

The image displays an MQTT Client Connector that integrates with a customer-premisses-equipment (CPE) gateway through an MQTT broker.



In this use case, a set of gas compressors publish their data to Cloud MQTT through a CPE gateway. A MQTT Client Connector subscribes to Cloud MQTT, which acts as a broker and redirects the data to the connector. When the connector receives the device data, it processes and sends the data to Oracle Internet of Things Cloud Service. The integration with Cloud MQTT allows connectivity with native MQTT devices.

Typical Workflow for Connecting Non-IP Devices

Non-IP devices indirectly connect to Oracle Internet of Things Cloud Service through a third-party cloud service. Oracle Internet of Things Cloud Service uses connectors to receive and process the data from such devices.

Use case: Orange offers LoRa network connectivity and exposes cloud service APIs for HTTP and MQTT connectivity. These APIs can be used to obtain the devices' telemetry data

Typical workflow for connecting non-IP devices in a LoRa network with Oracle Internet of Things Cloud Service

Task	More Information	Description
Subscribe to an Oracle Internet of Things Cloud Service instance.	Purchase a subscription to Oracle Cloud, set up an Oracle Internet of Things Cloud Service instance, and then access it.	Get Started with Oracle Internet of Things Cloud Service Applications
Review the built-in connectors and identify the type that's required for your devices and their network.	Built-in connectors includes **generic**, **OrangeLora**, and **ProximusLora**.	Create and Configure Connectors
Create a connector, select its type, and configure it.	Based on the type of connector, enter appropriate credentials, grammar, URL, API-Key, and other details.	Create and Configure Connectors
Verify that the connector has started.	Oracle Internet of Things Cloud Service automatically starts the connector and displays its status on the **Connectors** screen.	Create and Configure Connectors
Upload the required device models for all the device types.	A device model is an interface that helps the connector to identify the structure and the meaning of the device data content.	Create a New Device Model
Create an interpreter or create an interpreter at runtime when the connector receives a device notification .	When you configure an interpreter, you specify its selection criteria, a device model, and the payload processing information. For each device type, you create an interpreter.	Configure Interpreters
Review the device registration.	The connector obtains telemetry information by subscribing (MQTT) or periodically polling (HTTP). It registers a device and converts the payload data into device messages based on the envelope and payload processing information that you provide.	Locate a Device

Create and Configure Connectors

Connectors manage the connection and communication with devices or with third-party cloud services. They also process the metadata and telemetry data from the devices or the services.

Oracle Internet of Things Cloud Service provides multiple built-in connectors that allow declarative configuration to connect with third-party cloud services or network providers.

Types of Built-in Connectors

Use the Oracle Internet of Things Cloud Service management console to configure the following built-in connectors:

- **Generic**: Use this connector for all devices that can communicate over HTTP or MQTT, and have a basic authentication mechanism. You can also create a generic connector for an Orange or a Proximus network.
- **OrangeLora**: Use this connector when your devices' network provider is Orange and the devices are deployed in a LoRa network.
- **ProximusLora**: Use this connector when your devices' network provider is Proximus and the devices are deployed in a LoRa network.

Before you can start a connector that integrates devices in an external network or cloud service, you need to configure a stream in the network provider's server. When you configure this stream, you provide the URL of your Oracle Internet of Things Cloud Service instance, authentication information, stream format, and other required details.

Create an HTTP Server Connector

Create an HTTP Server connector when your network provider is other than Orange or Proximus. You can also customize an HTTP Server connector for devices in an Orange or Proximus network.

An HTTP Server connector is a generic connector. You'll need the user name and password for basic authentication. The user should have the administrator role in the Oracle Internet of Things Cloud Service instance. For an MQTT Client connector, you'll also need the MQTT broker URL, the client ID, and the topic name.

1. Open the Oracle Internet of Things Cloud Service management console.
2. Click **Menu** .
3. Click **Devices**, and then click **Connectors**.
4. On the **Connectors** page, click **Create New Connector** .
5. Enter values in the **Name** and **Description** fields.
6. Select **1** in the **Scale Factor** field if you need only one instance of the connector.
7. In the **Type** field, select **HTTP Server**.
8. In the **Telemetry** section, select the appropriate values.
   - **Envelope**: Click **Add Grammar**, and add the details in the **Grammar** window. See Learn About Adding Grammars. Ensure that you use the **Map** field to map the required device metadata fields with the values of the interpreted envelope data.
   - **Payload encoding**: If your payload is in JSON, select **asciihex**. if your payload is in base64-encoded binary, select **base64** .
9. (Optional) Configure the **Discovery** section when the third-party network provider has provided a REST endpoint to retrieve additional device metadata. You can use this metadata to specify the selection criteria of an interpreter.
   - **Protocol**: Select **Https**.
   - **URL**: Enter the URL of the REST endpoint to retrieve data.

- **Authentication**: Select **Basic authentication**, **HTTP Header** or **HTTP Query**.
  - For **Basic authentication**, enter values in the **User name** and **Password** fields. The user must have administrative access.
  - For **HTTP header**, click **headers**, enter the key and the value, and then click **OK**.
  - For **HTTP query**, click **parameters**, enter the key and the value, and then click **OK**.
- **Device metadata**: Click **Add Grammar** and add the details in the **Grammar** window. See Learn About Adding Grammars. Ensure that you use the **Map** field to map the required device metadata fields with the values of the interpreted data.

10. Click **Save**. When you receive the message **Saved successfully**, click **Back**.

    On the **Connectors** page, the HTTP Server connector is listed and the **State** field displays the status of the connector. The five states are **STARTING**, **STARTED**, **FAILED**, **STOPPING**, and **STOPPED**. **STARTED** indicates that the connector is ready to accept device data.

    - To modify your HTTP Server connector, click **Edit** .
    - To remove the HTTP Server connector, click **Delete** .
    - To create an instance of your HTTP Server connector, click **Start** .

    > **Note:**
    >
    > The **Edit**, **Delete**, and **Start** buttons are enabled only when the connector is in **STOPPED** state.

    - To view the device notifications that the connector receives, click **connectors_messageALERT** .

    > **Note:**
    >
    > On the **Notifications** page, click the data under **Content** to view the device information. You can also create an interpreter from this page when device data is received for the first time. See Configure Interpreters at Runtime.

    - To view the details of your connector, click **View** .

Create an MQTT Server Connector

Create an MQTT Server connector when you have a client which is a device or a network server that publishes telemetry over MQTT protocol.

To create and use an MQTT Server connector, you'll also need a sample payload that will be published.

1. Open the Oracle Internet of Things Cloud Service management console.
2. Click **Menu** .
3. Click **Devices**, and then click **Connectors**.

4. On the **Connectors** page, click **Create New Connector** .
5. Enter values in the **Name** and **Description** fields.
6. Select **1** in the **Scale Factor** field if you need only one instance of the connector.
7. In the **Type** field, select **MQTT Server**.
8. In the **Telemetry** section, select or enter the appropriate values.
   - Notice that the **Host**, **MQTT Port**, **WebSocket Port**, and **Context Path** fields. are populated for you.
   - **Topic**: Modify the default value to specify the MQTT topic that your client would publish into.
   - **Envelope**: Click **Add Grammar**, and add the details in the **Grammar** window. See Learn About Adding Grammars. Ensure that you use the **Map** field to map the required device metadata fields with the values of the interpreted envelope data.
   - **Payload encoding**: If your payload is in JSON, select **asciihex**. if your payload is in base64-encoded binary, select **base64** .

   > **Note:**
   >
   > When you configure an MQTT client to connect to the Oracle IoT MQTT Server Connector, you need to provide the Oracle Identity Cloud Service (idcs) credentials of a user having an administrator role in the Oracle Internet of Things Cloud Service instance. Enter the username as *<connector-name>/<idcs-user>* and password as *<idcs-password>*.

9. (Optional) Configure the **Discovery** section when the third-party network provider has provided a REST endpoint to retrieve additional device metadata. You can use this metadata to specify the selection criteria of an interpreter.
   - **Protocol**: Select **Https**.
   - **URL**: Enter the URL of the REST endpoint.
   - **Authentication**: Select **Basic authentication**, **HTTP Header** or **HTTP Query**.
     - For **Basic authentication**, enter values in the **User name** and **Password** fields.
     - For **HTTP header**, click **headers**, enter the key and the value, and then click **OK**.
     - For **HTTP query**, click **parameters**, enter the key and the value, and then click **OK**.
   - **Device metadata**: Click **Add Grammar** and add the details in the **Grammar** window. See Learn About Adding Grammars. Ensure that you use the **Map** field to map the required device metadata fields with the values of the interpreted data.
10. Click **Save**. When you receive the message **Saved successfully**, click **Back**.

    On the **Connectors** page, the MQTT Server connector is listed and the **State** field displays the status of the connector. The five states are **STARTING**, **STARTED**, **FAILED**, **STOPPING**, and **STOPPED**. **STARTED** indicates that the connector is ready to accept device data.

- To modify your MQTT Server connector, click **Edit** .
- To remove the MQTT Server connector, click **Delete** .
- To create an instance of your MQTT Server connector, click **Start** .

> **Note:**
>
> The **Edit**, **Delete**, and **Start** buttons are enabled only when the connector is in **STOPPED** state.

- To view the device notifications that the connector receives, click **Notifications** .

> **Note:**
>
> On the **Notifications** page, click the data under **Content** to view the device information. You can also create an interpreter from this page when device data is received for the first time. See Configure Interpreters at Runtime.

- To view the details of your MQTT Server connector, click **View** .

Create an MQTT Client Connector

Create an MQTT Client connector when your network provider subscribes to an MQTT broker and is other than Orange or Proximus. You can also customize a HTTP Server connector for devices in an Orange or Proximus network.

For an MQTT Client connector, you'll also need the MQTT broker URL, the client ID, and the topic name.

1. Open the Oracle Internet of Things Cloud Service management console.
2. Click **Menu** .
3. Click **Devices**, and then click **Connectors**.
4. On the **Connectors** page, click **Create New Connector** .
5. Enter values in the **Name** and **Description** fields.
6.
> **Note:**
>
> The distribution of messages to the individual connector instances is a function of the MQTT broker and is not determined by the connector. Some brokers evenly distribute messages across each connector instance, while some duplicate messages to each instance. If the MQTT broker cannot distribute messages evenly across the instances, then use a single scale instance of the connector.

   Select **1** in the **Scale Factor** field if you need only one instance of the connector.
7. In the **Type** field, select **MQTT Client**.

8. In the **Telemetry** section, select the appropriate values.
   - Complete the **MQTT Broker URL**, **Client ID**, and **Topic** fields.
   - **Authentication**:
     - For **Basic authentication** enter your user name and password.
     - For **HTTP header**, click **headers**, enter the key and the value, and then click **OK**.
   - **Envelope**: Click **Add Grammar**, and add the details in the **Grammar** window. See Learn About Adding Grammars. Ensure that you use the **Map** field to map the required device metadata fields with the values of the interpreted envelope data.
   - **Payload encoding**: If your payload is in JSON, select **asciihex**. if your payload is in base64-encoded binary, select **base64** .
9. (Optional) Configure the **Discovery** section when the third-party network provider has provided a REST endpoint to retrieve additional device metadata. You can use this metadata to specify the selection criteria of an interpreter.
   - **Protocol**: Select **Https**.
   - **URL**: Enter the URL of the REST endpoint.
   - **Authentication**: Select **Basic authentication**, **HTTP Header** or **HTTP Query**.
     - For **Basic authentication**, enter values in the **User name** and **Password** fields.
     - For **HTTP header**, click **headers**, enter the key and the value, and then click **OK**.
     - For **HTTP query**, click **parameters**, enter the key and the value, and then click **OK**.
   - **Device metadata**: Click **Add Grammar** and add the details in the **Grammar** window. See Learn About Adding Grammars. Ensure that you use the **Map** field to map the required device metadata fields with the values of the interpreted data.
10. Click **Save**. When you receive the message **Saved successfully**, click **Back**.

    On the **Connectors** page, the MQTT Client connector is listed and the **State** field displays the status of the connector. The five states are **STARTING**, **STARTED**, **FAILED**, **STOPPING**, and **STOPPED**. **STARTED** indicates that the connector is ready to accept device data.

    - To modify your MQTT Client connector, click **Edit** .
    - To remove the MQTT Client connector, click **Delete** .
    - To create an instance of your MQTT Client connector, click **Start** .

    > **Note:**
    >
    > The **Edit**, **Delete**, and **Start** buttons are enabled only when the connector is in **STOPPED** state.

- To view the device notifications that the connector receives, click **connectors_messageALERT** .

> **Note:**
>
> On the **Notifications** page, click the data under **Content** to view the device information. You can also create an interpreter from this page when device data is received for the first time. See Configure Interpreters at Runtime.

- To view the details of your MQTT Client connector, click **View** .

Create an OrangeLora Connector

You create an OrangeLora connector for devices deployed in a LoRa network provided by Orange

Before you create the connector, you should have the value of the API key of the Orange live objects that is needed for authentication.

1. Open the Oracle Internet of Things Cloud Service management console.
2. Click **Menu** .
3. Click **Devices** and then click **Connectors**
4. On the **Connectors** page, click **Create New Connector** .
5. Enter the values in the **Name** and **Description** fields.
6. In the **Scale Factor** field, select **1** .
7. In the **Type** field, select **OrangeLora**.
8. In the **API Key** field, enter the API key for the Orange live objects.
9. Click **Save**. When you receive a message**saved successfully**, then click **Back**.

   On the **Connectors** page, your OrangeLora connector is listed and its **State** is displayed.

   - To modify any of the fields of your OrangeLora connector, click **Edit** .
   - To remove the OrangeLora connector, click **Delete** .
   - To create an instance of your OrangeLora connector, click **Start** .

   > **Note:**
   >
   > The **Edit**, **Delete**, and **Start** buttons are enabled only when the connector is in **STOPPED** state.

   - To view the notifications received by the connector, click **connectors_messageALERT** .
   - To view the details of your OrangeLora connector, click **View** .

Create a ProximusLora Connector

You create a ProximusLora connector for devices deployed in a LoRa network provided by Proximus. It is an HTTP Server type connector.

Before you create the connector, you should have the user name and password that is needed for basic authentication.

1. Open the Oracle Internet of Things Cloud Service management console.
2. Click **Menu** .
3. Click **Devices**, and then click **Connectors**.
4. On the **Connectors** page, click **Create New Connector** .
5. Enter the values in the **Name** and **Description** fields.
6. In the **Scale Factor** field, select **1**.
7. In the **Type** field, select **ProximusLora**.
8. In the **Connection** section, ensure that the value for **HTTP URL** is the one that you entered when you configured the Proximus stream.
9. In the **Authentication** section, enter the user name and password.
10. Click **Save**. When you receive a message **saved successfully**, then click **Back**.

    On the **Connectors** page, your Proximus connector is listed and its **State** is displayed.

    - To modify any of the fields of your Proximus connector, click **Edit** .
    - To remove the Proximus connector, click **Delete** .
    - To create an instance of your Proximus connector, click **Start** .

    > **Note:**
    >
    > The **Edit**, **Delete**, and **Start** buttons are enabled only when the connector is in **STOPPED** state.

    - To view the notifications received by the connector, click **connectors_messageALERT** .
    - To view the details of your Proximus connector, click **View** .

About Interpreters

An interpreter parses the payload and extracts the device telemetry data. The connector then uses the data to form a device message in Oracle Internet of Things Cloud Service. Interpreters can be categorized as JSON or binary. Binary interpreters support additional payload grammars which can be used to parse device-native binary payloads

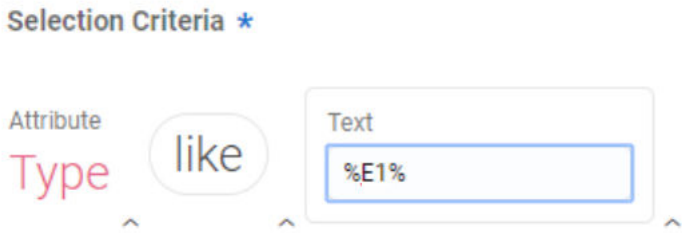
An interpreter is needed to parse each type of payload. When you configure an interpreter, you provide its selection criteria, which the connector uses to select an interpreter.

Declarative Configuration of an Interpreter

In Oracle Internet of Things Cloud Service, you create an interpreter using the management console. You can also create an interpreter at runtime when a connector receives a notification from a device.

Before you configure an interpreter, you need these details:

- Selection criteria: Provide information that the connector uses to select a suitable interpreter to parse the payload data. For a connector that lets you connect multiple types of devices, the envelope data of the connector must have some suitably unique field value that can be used to identify different payload types. An example, if the envelop contains a `type` field which contains the sub-string `E1` where `E1` defines a type of payload data, then the selection criteria will look like `type LIKE "%E1%"` where `type` is a metadata field parsed from the envelop data by the connector.



- Device models: Specify the name of the device model for the interpreter. Your Oracle Internet of Things Cloud Service instance should include all the required device models.
- Sample payload data and payload grammar: Provide a sample payload data for a payload type. If the sample payload is in JSON, the grammar is built in and the parsing is done by the connector. If the sample payload in binary, you'll need to provide the grammar for parsing the binary payload. See Learn About Adding Grammars
- Device metadata: Map the standard metadata fields such as Name, Hardware ID, or custom fields to the parsed envelope data. This helps the connector to create a device record when it registers the device in Oracle Internet of Things Cloud Service.
- Alerts and Custom Data: Add any alerts or custom data message types contained in the selected device model to be signaled through this connector. Each alert or custom data format can have its own sample payload and field map.

Configure Interpreters

Use the management console of your Oracle Internet of Things Cloud Service instance to create an interpreter.

Create an interpreter for each type of payload. The connector selects an interpreter by matching its data with the selection criteria that you provide. To create an interpreter from the management console:

1. Open the Oracle Internet of Things Cloud Service management console.
2. Click **Menu** .
3. Click **Devices**, and then click **Interpreters**.

4. On the **Interpreters** page, click **Create New Interpreter** .
5. Enter values in the **Name** and **Description** fields.
6. In the **Selection Criteria** field, select an attribute and map it to a pattern or value. The connector uses the selection criteria to select the interpreter.
7. Select the **Device Model**, and click **Add Grammar**.
   a. Do one of the following:

      For a JSON payload, enter the payload data in **Sample Data** and click **Validate**. The interpreted values are listed.

      For a binary payload, enter the payload in **Sample Data**, enter the binary grammar in **Grammar**, and click **Validate**. The interpreted values are listed.

   b. From the **Map** list box, select an attribute of the device model.
   c. From the adjacent combo box, select **Property**, **ATTRIBUTE**, and then the field of one of the interpreted values. Values can also be combined or manipulated using the **Formula** menu.
   d. To select another attribute, click **Add**. Repeat steps c and d to map the attributes of the device model that you need in the device message.
8. If the selected **Device Model** contains alert or custom data format definitions, you may define additional criteria and mapping for them. In such a case a section titled **Alerts and Custom Data** appears.
   a. For each additional alert or custom data message you would like to define, from the drop-down, select the format similar to that in the device model. You may add additional entries by clicking
   b. In **Click here to insert criteria**, use the editor to enter the criteria formula that will be evaluated to generate the data message or alert.
   c. Click the **Map Fields** button to enter attribute mapping as was done previously with the **Device Model** grammar.
9. In the **Device Metadata** section, Select a field from the list box and map it to an attribute. Click **Add**, and repeat the step for all the fields that you wish to map. These fields appear as metadata in the device record when the device is registered in Oracle Internet of Things Cloud Service.
10. Click **Save**.
11. When you receive a message **Saved successfully**, then click **Back**.

    On the **Interpreters** page, your interpreter is listed and its **Device Model** is displayed.

    - To modify any of the fields of your interpreter, click **Edit** .
    - To remove the interpreter, click **Delete** .

Configure Interpreters at Runtime

You can create an interpreter at runtime after the connector receives data from an unknown device.

When a connector is configured to a provider's network, it can receive data from all the devices in that network. Complete the following steps to configure an interpreter at runtime only when a connector receives data from an unknown device. This interpreter

will help in registering and activating the device, and then saving the device message in Oracle Internet of Things Cloud Service. Ensure that the required device model is already added to Oracle Internet of Things Cloud Service.

1. Open the Oracle Internet of Things Cloud Service management console.
2. Click **Menu** .
3. Click **Devices**, and then click **Connectors**.
4. On the **Connectors** page, identify a connector whose **State** is **STARTED** and then click **connectors_messageALERT** .
5. On the **Notifications** page, review the **Content** column, and identify the notification record that is received from an unknown device type. Click the **Notification Accept**icon.
6. On the Device page, click **Create Interpreter**.
7. To complete the configuration, follow Step 5 through Step 10 of Configure Interpreters.

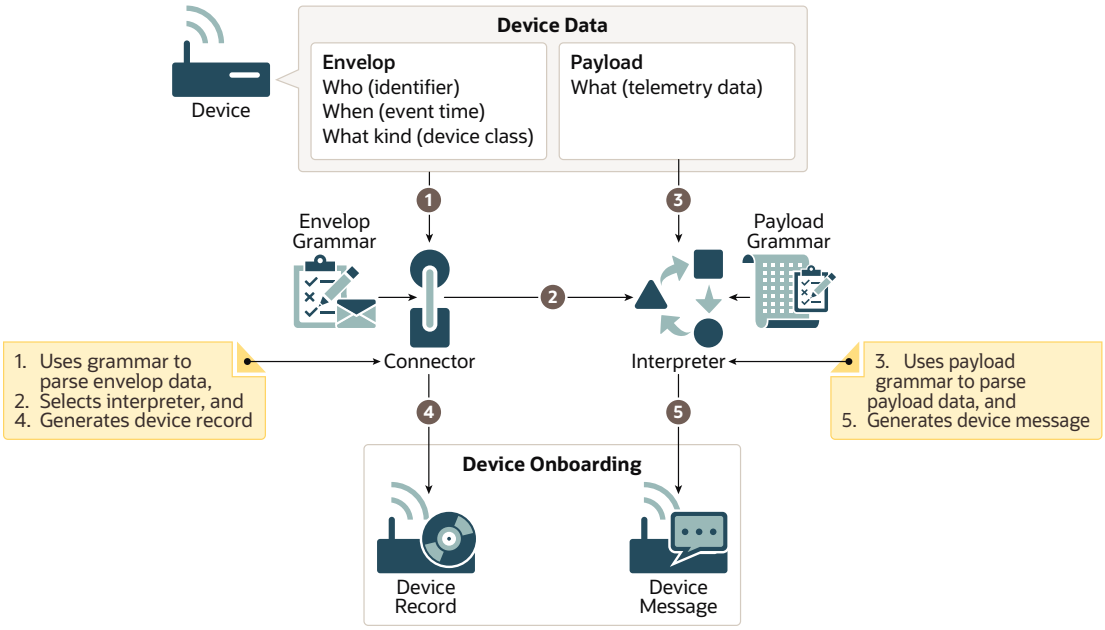
Learn About Adding Grammars

Connectors and interpreters use grammar to parse the content of device data. Typically, the device data is in JSON and connectors use a standard grammar to parse it. However, when device data is obtained from a bandwidth constrained network such as a LoRa network, the data is in a binary form, then the user has to provide the grammar that the connector and the interpreter use to parse the device data.

In a generic connector and an interpreter, you add a grammar that helps parse the device data. The connector uses the parsed data to create a device record and a device message for device on-boarding in Oracle Internet of Things Cloud Service.

Envelope Grammar and Payload Grammar

The image displays the usage of grammar in connectors and interpreters.



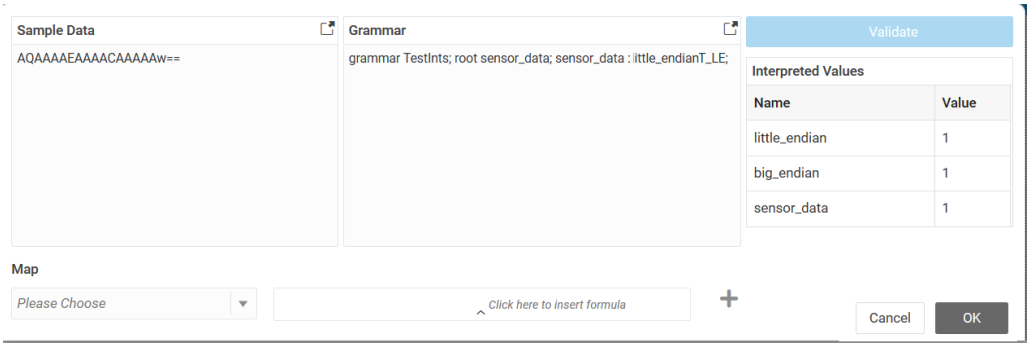
Device data has two parts: an envelope and a payload. Envelope data has the same structure for all types of device data. It has an identifier, an event time, and a device type.

Connectors read the device data, use grammar to parse it, and then interpret the envelope data. When you configure a connector, you also need to map a few standard metadata fields with the interpreted envelope data. The connector uses the mapped data to generate a device record that is used for registering or on-boarding the device in Oracle Internet of Things Cloud Service.

Payload data can be of different types. For every type of payload, you create an interpreter. When you configure an interpreter, you provide its selection criteria. The connector selects the interpreter based on this selection criteria. The selected interpreter then uses grammar to parse and interpret the payload data. You also need to map the fields of the device model with the interpreted payload data. The interpreter uses the mapped data to generate a device message for the registered device.

When you configure a connector, on the **Grammar** page, you provide the device data in JSON in the **Sample Data** text area and then click **Validate** to display the interpreted values. You don't have to enter grammar for parsing data in JSON because JSON grammar is standardized. If the data is in binary, then you enter the grammar to parse the binary data. You use the **Map** list box to map a standard set of built-in device metadata such as Hardware ID, name, time-stamp, etc. with the interpreted values. You can also create customized metadata. When the connector registers the device in Oracle Internet of Things Cloud Service, the mapped fields and the interpreted values form the device record.

The image displays the Add Grammar page.



When you configure an interpreter, on the **Grammar** page, you paste the payload data in the **Sample Data** text area. If the data is in binary then you need to enter the grammar to parse the Base64-encoded binary data. On validation, the interpreted values are displayed. You use the **Map** list box to map the attributes of the device model with the interpreted values. The mapped fields and interpreted values form the device message that the connector saves in Oracle Internet of Things Cloud Service for the registered device.

On the **Grammar** page of the **Connector** screen, in the **Sample Data** text area, you provide the complete device data in JSON and then click **Validate**.

```
{
    "companyName": "TestDemo",
    "Name": "Temperature and humidity",
    "Version": "1",
    "Type": "Temperature and humidity",
    "vendor": "XEE",
    "latitude": "",
    "longitude": "",
    "description": "",
```

```
    "locationFriendlyName1": "Brazil test towers",
    "locationFriendlyName2": "",
    "containerFriendlyName": "Temperature",....
     .....
     "payload": "0902000029067a"}
```

On the **Grammar** page of the **Interpreter** screen, in the **Sample Data** text area, you provide the payload data in binary.

```
 AWDiAikEAcCxXyYDCAcNYg==
```

On the **Grammar** page of the **Interpreter** screen, in the **Grammar** text area, you provide the grammar in binary and then click **Validate.**

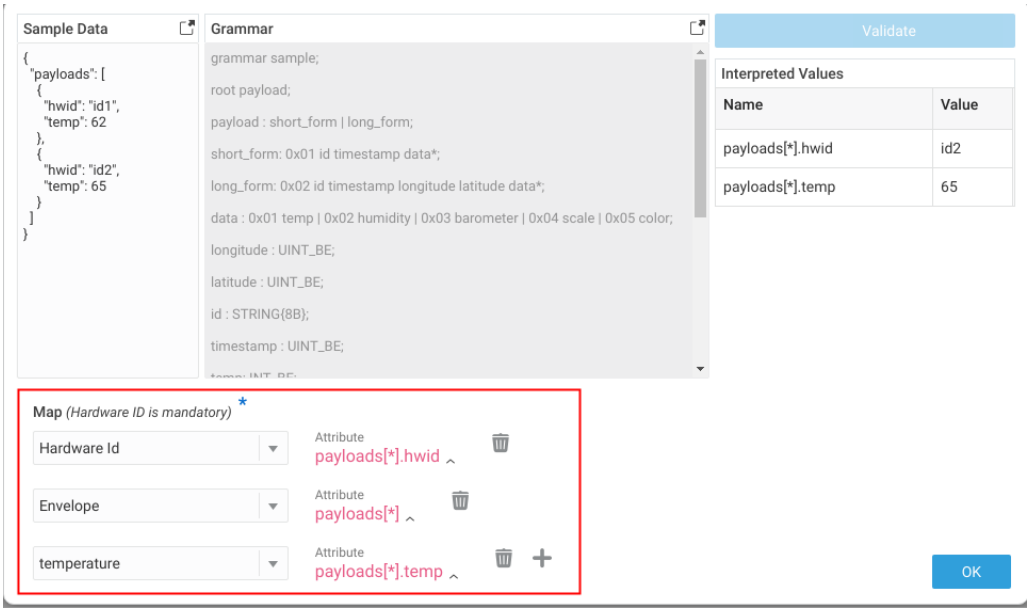
```
grammar xyz; urn:lora:xxxxx:lpp; root payload; payload: data*;
data: channel 0x00{8b} digitalInput | channel 0x01{8b} digitalOutput
| channel 0x02{8b} analogInput | channel 0x03{8b} analogOutput |
channel 0x65{8b} illuminance | channel 0x66{8b} presence | channel 0x67{8b}
temperature | channel 0x68{8b} humidity | channel 0x71{8b} accelerometer |
channel 0x73{8b} barometer | channel 0x86{8b}.............
```

Support for Compound Payloads

When configuring connectors, you can map the `Envelope` variable to support JSON data that contains multiple payloads in an array. For example, the following JSON has a field `payloads` which is an array containing objects, each of which represent a stand-alone measurement.

```
{
 "payloads": [
   {
     "hwid": "id1",
     "temp": 62
    },
    {
      "hwid": "id2",
      "temp": 65
     }
   ]
}
```

In the example, when the `Envelope` field is configured to point to the array (`payloads[*]`), the connector framework will treat each object in the payload's array as a separate payload. Fields within the contained objects may be referenced and mapped through similar array notation (e.g. `payloads[*].temp`).

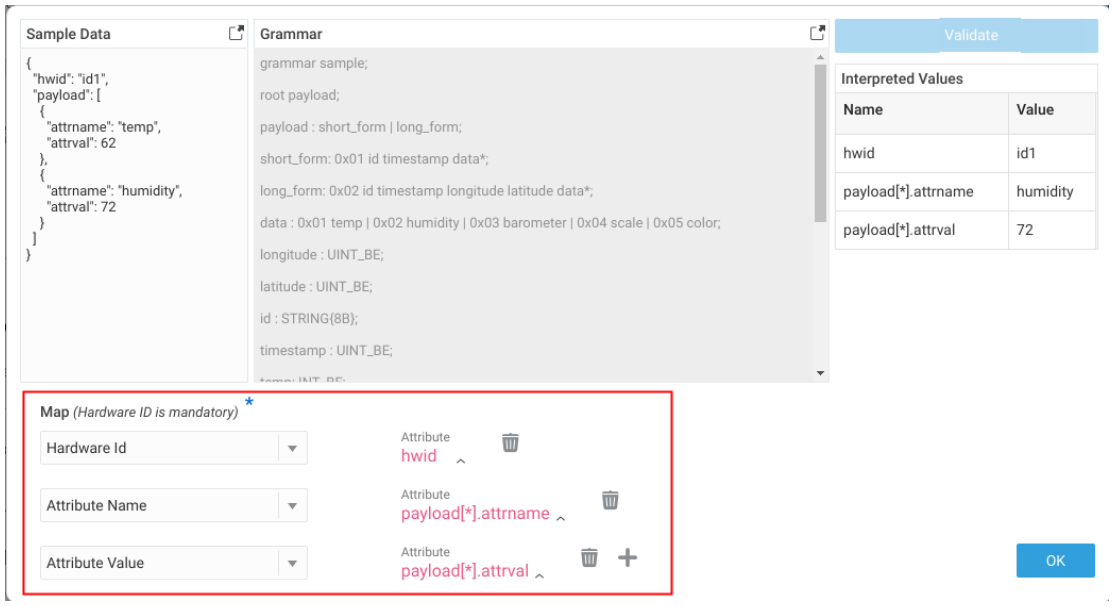


Support for Value-Named Fields

There are some circumstances where the attributes names of the measurement data are themselves contained within the JSON string values. Consider the following example, where measurements are sent as an array of objects and the field names are contained in the `attrname` field:

```
{
  "hwid": "id1",
  "payload": [
    {
      "attrname": "temp",
      "attrval": 62
    },
    {
      "attrname": "humidity",
      "attrval": 72
    }
  ]
}
```

This payload contains a `temp` measurement with a value of `62`, and a `humidity` measurement with a value of `72`. This can be handled by the mapping the `Attribute Name` and `Attribute Value` fields. The connector framework will convert the above payload into two measurements (`temp = 62, humidity = 72`) with the configuration as shown in the image.


Sample Data
{
"hwid": "id1",
"payload": [
{
"attrname": "temp",
"attrval": 62
},
{
"attrname": "humidity",
"attrval": 72
}
]
}
Grammar
grammar sample;
root payload;
payload : short_form | long_form;
short_form: 0x01 id timestamp data*;
long_form: 0x02 id timestamp longitude latitude data*;
data : 0x01 temp | 0x02 humidity | 0x03 barometer | 0x04 scale | 0x05 color;
longitude : UINT_BE;
latitude : UINT_BE;
id : STRING{8B};
timestamp : UINT_BE;
Validate
Interpreted Values
Name
Value
hwid
id1
payload[*].attrname
humidity
payload[*].attrval
72
Map (Hardware ID is mandatory)
Hardware Id
Attribute
hwid
Attribute Name
Attribute
payload[*].attrname
Attribute Value
Attribute
payload[*].attrval
OK

9

Create and Manage Explorations

Use these topics to learn how to create and manage explorations. Use explorations to process and analyze data messages sent from your devices to Oracle Internet of Things Cloud Service. **Note that the creation of explorations is now deprecated and the feature will be removed in future releases.**

Topics

- Add a Message Exploration to an Application
- Add Stream Explorer References to Static Data
- Create a New Map
- Use Maps to Create Explorations
- Edit a Map
- Edit Explorations

About Oracle IoT Cloud Service Stream Exploration

The stream exploration feature of Oracle IoT Cloud Service allows you to process and analyze data messages that are sent from your devices to Oracle IoT Cloud Service. **This feature is now deprecated and it will be removed in future releases.**

> **Note:**
>
> This feature is now deprecated and will be removed in a future release.

The stream exploration feature provides a business focused visual approach to real-time streaming event analytics. It is based on the Oracle Stream Analytics tool and exposes some of the features that the tool offers. Using the Oracle IoT Cloud Service Management Console, you create an exploration by using one of your previously defined exploration sources that you want to analyze further. You can also annotate the data message being sent with device-specific metadata. Doing so will give you additional identifying criteria to use during your data analysis. See Understanding Explorations in *Using Oracle Stream Analytics* and Add a Message Exploration to an Application for more information.

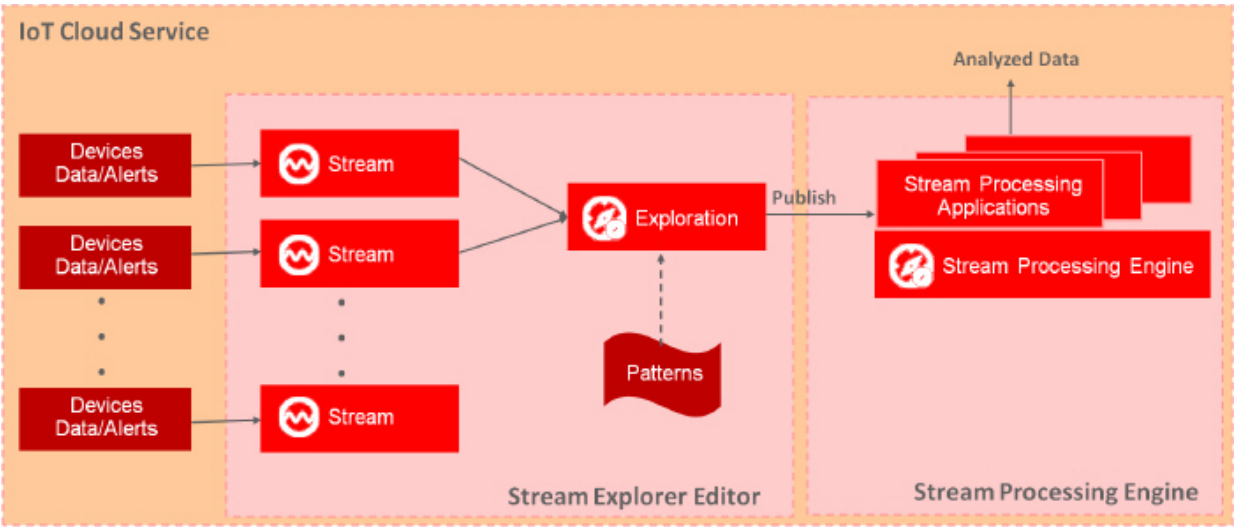
Once an exploration is published, it becomes available as one of the message formats to choose from when you're configuring your IoT Application's integration with external services, such as Oracle Business Integration (BI) Cloud Service, Oracle Mobile Cloud Service, JD Edwards EnterpriseOne IoT Orchestrator, Oracle Storage Cloud Service, or an enterprise application.

Some of the limitations of the explorations feature in Oracle IoT Cloud Service include the following:

- There is no support for references.

- Publishing explorations to targets outside of Oracle IoT Cloud Service is not supported.
- There is no support for creating data streams from external sources.

The image below shows the workflow of the real-time analytics application.



Streams

Streams in the above diagram represent the streams of data being sent to the stream explorer tool. Using the Data and Explorations page within the scope of an IoT Application, you create exploration using previously defined exploration sources data or alert messages that you want to further explore. See Add a Message Exploration to an Application for more information.

Explorations

Oracle IoT Cloud Service Stream Explorer explorations are visual representation of real time disparate event data flows for insightful data interrogation and to quickly apply sophisticated business intelligence. They allow you to define business criteria for managing data. They are the visual representation of the data streamed from Oracle IoT Cloud Service devices. These explorations allow you to perform the following tasks:

- Correlate data or alerts from multiple device data streams
- Filter and group data.
- Specify the time/event-based windows for aggregation functions.
- Specify aggregation functions to be used in summaries.
- Review incoming events (before you apply the logic) and resulting events (after you apply the logic) in tabular and graph forms

Patterns

A Stream Explorer pattern provides you with a simple way to explore event Streams based on common business scenarios. A pattern is a template of an Oracle Stream Analytics application that already has the business logic built into it. The visual

representation of the stream varies from one pattern type to another based on the key fields in the stream you choose to use.

The following table lists some of the patterns available in Stream Explorer.

Pattern Name	Description
Top N	First N events in the specified time window
Bottom N	Last N events in the specified time window
Up Trend	Detects when the up trend starts and continues to grow. E.g., use this pattern to identify when the temperature value from a sensor device starts increasing continuously
Down Trend	Detects when a numeric event field shows a change in a specified trend, lower in value. E.g., to identify when the temperature value from a sensor device starts continuously decreasing
Fluctuation	detects when an event data field value changes in a specific upward or downward fashion within a specific time. E.g., to identify the variable changes in pressure values within acceptable ranges
Eliminate Duplicates	Eliminates duplicate events in a specific event stream
Detect Duplicates	Detects when an event data field has duplicate values within a specified period of time
Detect Missing Event	Detects when an expected event does not occur within a specified time window. e.g, use this pattern in circumstances when the next heartbeat event is missing
W pattern	Detects when an event data field value rises and falls in a W fashion over a specified time period
Inverse W	Detects the inverses of W
Spatial General	Analyzes streams containing geo-location data and determine how events relate to pre-defined geo-fences in your maps.

Add a Message Exploration to an Application

Add a message exploration to your application to analyze and process the message data sent by the devices that are specified in the application.

1. Open the Oracle Internet of Things Cloud Service Management Console.
2. Click the **Menu** (☰) icon next to the Oracle Internet of Things Cloud Service title on the Management Console.
3. Click **Applications**.
4. Click the application name and then click **Data and Explorations**.
5. Click the **Explorations** tab.
6. Create the exploration source:
    a. Click **Add** in the **Explorations Sources** area.
    b. Enter a name for the exploration source in the **Name** field.
    c. Select **Message** in the **Source Type** list.
    d. Select a message format. in the **Message Format** list.
    e. Select the device metadata you want to collect in the **Device Metadata** list.
7. Click **Confirm**.

8. Associate the exploration source with an exploration:
   a. Click **Add** in the **Explorations** area.
   b. Select **Exploration** in the **Exploration Type** list.
   c. Enter a name for the exploration in the **Exploration Name** field.
   d. Select the exploration source you created in step 6 in the **Exploration Source** list.
9. Click **Confirm**.
10. Add filters, business rules, and then publish the exploration:
    a. Select the exploration in the **Explorations** list.
    b. Click the **Edit** ( ) icon.
    c. Add optional filters for the exploration in the **Filters** area.
    d. Add optional business rules in the **Business Rules** area.
    e. Click **Actions** and select **Publish**.
11. Click **Back** to return to the**Data and Explorations** pane.
12. Create an integration and add the exploration as a stream to the integration.

Add Stream Explorer References to Static Data

Add references to enrich messages sent by devices to analyze and process the data. Once the data appears in the messages source, you can explore and enrich the data.

To add a database table to the list of available exploration sources:

1. Go to the **Explorations** tab in the Data and Exploration page of your application. The upper part of the page allows you to add and delete exploration sources which can be device messages, database tables, or maps. The lower part allows you to add, edit, and delete explorations. Use the Add, Edit, and Delete icons to perform required actions.
2. Click the **Add** icon in the exploration source section, enter a name for the new source, and select **Database** as the source type.
3. If your database connection is already defined, select it from the list. Otherwise, select **Create a New Connection**, enter the database connection details, and click **Create**. The connection details are similar to the ones required when configuring DBaaS as a connection in SQL developer.
4. Select a table from the list in the **Table** or **View** field.

To select the database table as a source in an existing exploration:

1. In the **Exploration** tab, select the exploration you want to modify and then click **Edit**.
2. Click anywhere inside the **Sources** box.
3. Select the name of the database table exploration source created in step 1.
4. Select a property from the device message source and database table source to correlate in the **Correlation** box.

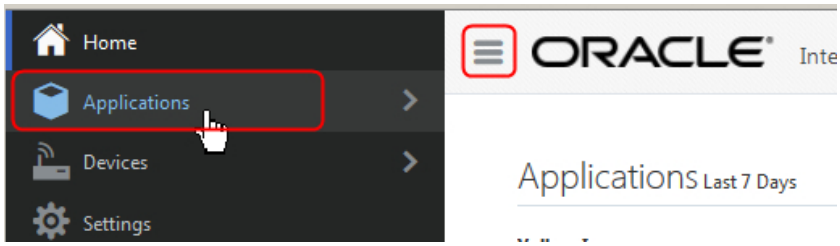
Based on the correlation properties defined, incoming device messages will be enriched in the live data stream.

Create a New Map

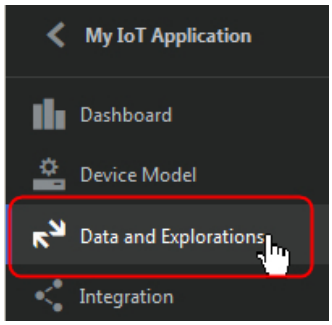
A map is a virtual fencing over a geographical area to define boundaries. The maps help you in defining boundaries over a specific area and then use it to analyze the data for your business. Based on the data you analyze, you can use the data to improve your business performance.

To create a new map:

1. From the Management Console's **Home** menu, click **Applications**.



2. Click the IoT Application's name to go to its home page and then click **Data and Explorations**.

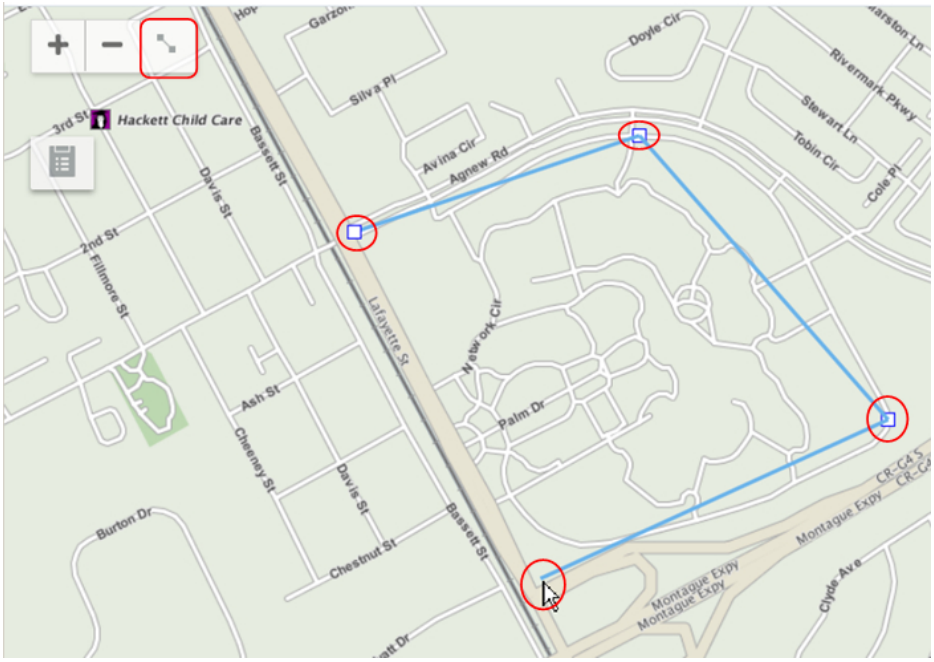


3. From the Data and Explorations page, click **Explorations** and in the Maps section, click **Add**.
4. In the Add dialog, enter a name for the map in the **Name** text field.
5. (Optional) Modify the default value in the **Description** text field.
6. Click **Confirm**.

   A new table with a row for the new map is added in the Maps section
7. Select the row for the newly created map and click **Edit** to define its boundaries.

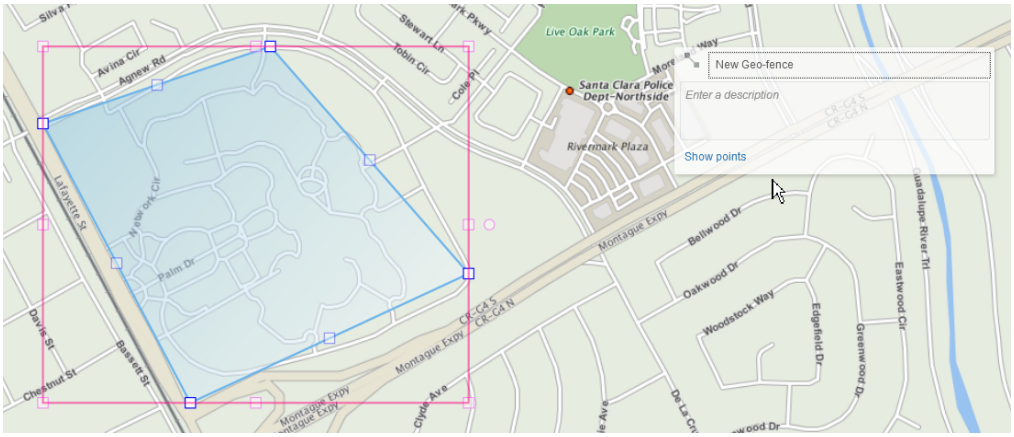
The Edit Map page, similar to the following image, is displayed if you opted to not share your current location.

8. Use the zoom in and zoom out controls to change the view of the map.
9. Use the polygon tool control to define the boundaries.
    a. Click the polygon tool icon. 
    b. Click on an initial point on the map to begin defining the boundaries of the Geo-fence.
    c. Continue clicking specific points to create the boundaries of the geo-fence..



    d. Complete the geo-fence by clicking on the initial point again.

The map is updated with the new geo-fence and the polygon tool dialog.



e. Use the polygon tool dialog to enter a name and description for the new geo-fence.

f. Click **Show points** to display the coordinates used for the geo-fence.

The map is now ready for you to use with your exploration. See Use Maps to Create Explorations.

Use Maps to Create Explorations

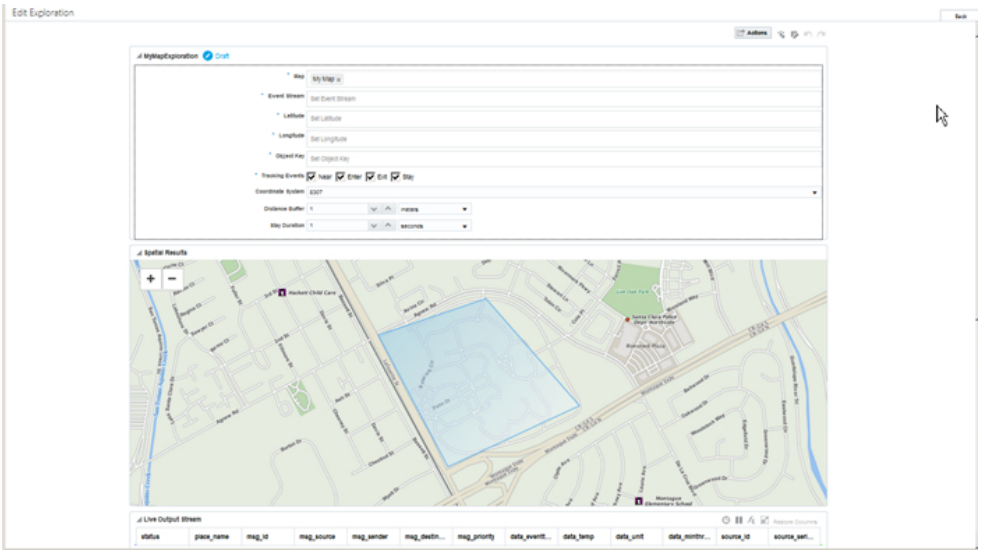
You can use maps to set up alerts for your IoT application. Whenever a device enters or exits the defined boundaries, the you can be alerted .

Ensure that you have already defined at least one map using the information in Create a New Map before you proceed with the following steps.

To create explorations using maps:

1. In the Explorations section of the Data and Explorations page, click **Add**.
2. In the **Exploration Type** text field, select Spatial General from the list of Patterns.
3. Enter a value in the **Exploration Name** text field.
4. Click **Confirm**.

The **Edit Exploration** page is displayed, similar to the following image:



5. Click in **Map** and select from the list of maps you've created earlier.
6. Click in **Event Stream** field, and select from the list of available sources.
7. Select a field to be used for **Latitude**.
8. Select a field to be used for **Longitude** for the spatial.
9. Select a field to be used as the **Object Key**.

   The object key, latitude, and longitude determine the way spatial pattern recognizes data from the incoming messages and correlatse them with the geo-fences in the map.

   Typically the Object Key is the `message_source` or a field that identifies a device uniquely.
10. Select the options for **Tracking Events**. This field determines the proximity of the location for the pattern. The possible values are:
    - **Near** - whenever a new event occurs in the defined spatial region defined by the specified latitude and longitude.
    - **Enter** - whenever an existing event enters the spatial region defined by the specified latitude and longitude.
    - **Exit** - whenever an event exits the spatial region defined by the specified latitude and longitude.
    - **Stay** - when an event stays (the duration is specified in step 11) in the spatial region defined by the specified latitude and longitude.
11. Select the **Coordinate System** ID from the list.

    The possible values are:
    - *3857* - also known as WGS 84/Pseudo-Mercator. This is a geodetic projected Cartesian 2d coordinate system
    - *8307* - also known as Longitude/Latitude (WGS 84). This an Elipsoidal 2d coordinate system. This coordinate system is introduced by Oracle and is used by default.

12. Enter a numerical value greater than zero for the **Distance Buffer**. This field is enabled only when you select **Near** in **Tracking Events**.
13. Enter a numerical value greater than zero for the **Stay Duration**. This field is enabled only when you select **Stay** in **Tracking Events**

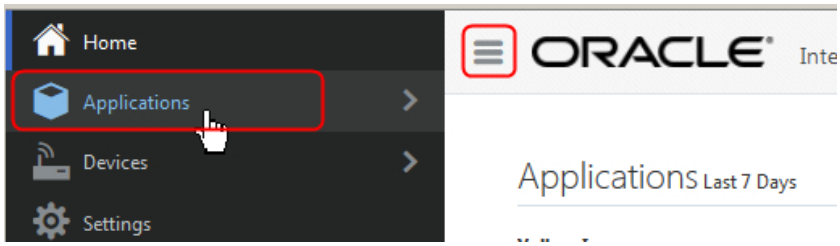
The exploration is now visually represented in the Spatial Results section in the form of a map that's based on the data you have specified in the previous fields.

Edit a Map

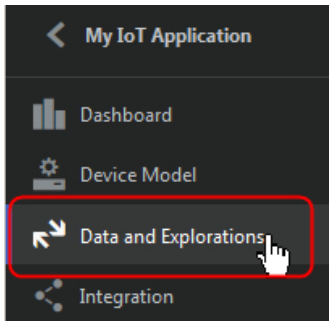
After you create the map, you need to edit it to define its boundaries of the Geo-Fence you want use with your IoT application.

To edit a map:

1. From the Management Console's **Home** menu, click **Applications**.



2. Click the IoT Application's name to go to its home page and then click **Data and Explorations**.



3. From the Data and Explorations page, click **Explorations** and in the Maps section, select a map and click **Edit**.
4. Click the **Saved Geo-Fences** icon to view the saved maps 
5. From the *map's* list, select the saved geo-fence.
6. Use the dialog to modify the information about the saved geo-fence.
7. Use the **Visibility** icon to toggle the visibility of the selected geo-fence on the map.
8. Use the **Delete** icon to remove the saved geo-fence.
9. Click **Back** to return to the Data and Explorations page.

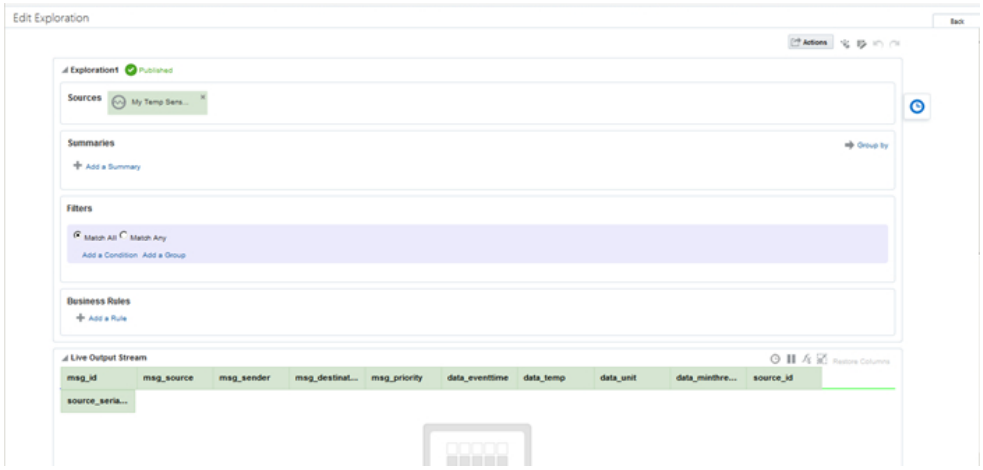
Edit Explorations

After you create an exploration, you can edit it using the Exploration Editor. You have the option to edit the parameters of the exploration, add or modify the data sources for the exploration, and publish the exploration.

To edit an exploration:

1. Navigate to the **Data and Explorations** page for your IoT Application.
2. Go to the **Explorations** section, select the exploration you want to edit, and click **View**.

   The **Exploration Editor** is displayed, as illustrated below:



3. Click the **Actions** menu located at the top right corner of the Exploration Editor and choose **Publish**, **Unpublish**, or **Discard**.
4. Click the **Show Topology** icon to view the topology. For more information, see Topology Viewer.
5. Click the pencil icon at the top-right corner to view the information of the exploration.
   a. Click **Edit Info** to update Name, Description, or Tags of the exploration.
   b. Click **Done** after updating the information.
6. Click **Undo Changes** icon at the top-right corner to undo the changes you have made. Click **Redo Changes** icon at the top-right corner to re-apply the changes you have made.
7. Click anywhere in the **Sources** field to add or remove data sources. If you have more than one data source, the sources must be correlated
8. Define the **Correlations** if the exploration has more than one data source.
9. Click **Add a Summary** in the **Summaries** area to add a summary to the exploration.
10. Click **Add a Condition** or **Add a Group** in the **Filters** area to add filters to the output. Choose **Match All** or **Match Any** to define how the filters should be applied. The filters further refine the streaming data based on the conditions you've added.

11. Click **Add Rule** in the **Business Rules** area to add business rules to the exploration. Business rules build conditional logic into your applications. See Business Rules.
12. Use the **Live Output Stream** to view the data stream that meets the exploration's criteria.
    a. Click **Timestamp** to display the timestamp in the **Live Output Stream**. The timestamp at which the streaming data was captured appears against each row of live output stream. This is a toggle button and you can use it to switch ON or OFF the timestamp in the live data.
    b. Pause the live data that is streaming using the **Pause** icon and resume the stream using the **Resume** icon.
    c. Click **Restore Columns** to restore any of the columns that you have hidden earlier.
    d. Click **Detach** to detach the Live Output Stream into a separate window. You can attach it later again.
    e. Manage the header of any columns by clicking the header and renaming the columns. You can also click and drag any of the columns in the **Live Output Stream** to reorder the columns. Resize the columns using the mouse. You can also hide or reveal the columns.
    f. Add expressions or functions to the live output stream to calculate certain data based on the data that is streaming. Use the Expression Editor to add a function or expression. Click the expression editor and enter = followed by the required function in the **Enter your expression** field. Click the **Done** icon to finish adding the expressions and save the added expressions or functions. Click **Cancel** to abort the operation at any time. For more information, see Expression Builder.
13. Configure the number of graphs you want to be displayed for the output stream in the **Charts** area.
14. Click **Publish**.

10

Integrate with External Services

Use these topics to learn how to integrate Oracle Internet of Things Cloud Service with an enterprise application or an external service.

Topics

- Integrate Enterprise Applications with Oracle IoT Cloud Service
- Integrate Oracle Analytics Cloud with Oracle Internet of Things Cloud Service
- Integrate Oracle Mobile Cloud Service with Oracle IoT Cloud Service
- Integrate Oracle IoT Cloud Service with JD Edwards EnterpriseOne IoT Orchestrator
- Integrate Oracle Storage Cloud Service with Oracle IoT Cloud Service
- Integrate Oracle Big Data Cloud Service with Oracle Internet of Things Cloud Service
- Integrate Oracle IoT Cloud Service with Oracle Integration Cloud Service
- Add a Custom Certificate for an Integration
- Verify Integration Connectivity

Integrate Enterprise Applications with Oracle IoT Cloud Service

Integrate your enterprise application to send, receive, and manage Oracle Internet of Things Cloud Service device data.

See Developing Enterprise Applications That Use Oracle IoT Cloud Service to learn about the typical workflow for developing enterprise applications.

1. Open the Oracle Internet of Things Cloud Service Management Console. See Access Oracle IoT Cloud Service.
2. Click the **Menu** ( ) icon adjacent to the Oracle Internet of Things Cloud Service title on the Management Console.
3. Click **Applications**.
4. Click an application.
5. Click **Integration**.
6. Select one of these options:
   - If you have not previously created an integration, click **Create Integration** and then select **Enterprise Application**.
   - If you have previously created an integration, click the **Add** ( ) icon and then select **Enterprise Application**.
7. Complete these fields:
   - **Name**: Enter a unique name for the Oracle Internet of Things Cloud Service integration.

- **Description**: Enter an optional description for the Oracle Internet of Things Cloud Service integration.
- **URL**: Enter the URL that Oracle Internet of Things Cloud Service will use to transfer data.

8. (Optional) Click **Verify Connectivity** to test the connection between theOracle Internet of Things Cloud Service instance and the target service or application.

   The connectivity test can include:

   - DNS Resolution Test: Checks whether the server name resolves to an IP address.
   - Connectivity Test: Checks whether the IP address is reachable.
   - IP/Port test: Checks whether you can talk to the target URL on the specified port.
   - SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.
   - Authentication Test: Checks whether you can exchange messages with the target.

9. Click **Create**.

10. Select the integration in the **Integrations** list and then click the **Edit** ( ) icon.

11. To manage devices using the enterprise application:

    a. Click the **Overview** tab.

    b. Enter a password in the **File Protection Password** field to encrypt the provisioning file that contains the configuration and credentials to activate your integration.

    c. Enter the password again in the **Confirm Password** field.

    d. Click **Download Provisioning File**.

    e. Click **Save File**.

    f. Browse to a location to save the file and then click **Save**.

    g. Click the **Connection** tab.

    h. Enter a URL in the **URL** field.

       If you intend to use the enterprise application to control and manage devices, enter a value such as **http://notused**.

12. To send unmodified device data to an enterprise application listening on an HTTP URL endpoint:

    a. Click the **Connection** tab.

    b. Enter the URL for the enterprise application server in the **URL** field.

    c. Select **Enable authentication for this URL** if user authentication is used on the enterprise application server.

    d. Enter a user name in the **Username** field.

    e. Enter a password in the **Password** field.

13. (Optional) To add headers to the outgoing HTTP requests being sent to your enterprise application:

> **Note:**
>
> By default, the `Content-Type; application/json` header is included when the integration is created.

a. Click the **Custom Headers** tab.
b. Click **Create a Header**.
c. Enter a name for the header in the **Header Name** field.
d. Enter a value for the header in the **Header Value** field.
e. Click the **Add** (⊕) icon to add another header.

14. Add a stream to the integration:
    a. Click the **Streams** tab.
    b. Click **Create a Stream**.
    c. Select a message format in the **Message Format** list.

       The same message format cannot be used by multiple streams. Select a unique message format for each stream you create.
    d. Select annotations in the **Annotations** list. Annotations identify the metadata sent to the enterprise application.

       If you previously created custom metadata keys for your devices and streams, then these also appear in the list of annotations.

       You can also type a new annotation name and select the data type for the annotation.
    e. Click the **Add** (⊕) icon to add another stream.
15. Click the **Messages** tab to view integration messages.
16. Click **Save** .

Integrate Oracle Business Intelligence Cloud Service with Oracle Internet of Things Cloud Service

Use your existing Oracle Business Intelligence Cloud Service instance to analyze device message data. To perform the analysis, use this procedure to integrate your existing Oracle Business Intelligence Cloud Service with your Oracle Internet of Things Cloud Service instance.

These are the prerequisites for the successful completion of this procedure:

- An active Oracle Business Intelligence Cloud Service instance.
- An active Oracle Internet of Things Cloud Service instance.
- An Oracle Internet of Things Cloud Service application.
- A device model associated with the Oracle Internet of Things Cloud Service application. The device model determines what message format can be used in the data stream. See Assigning a Device Model to a Cloud Service.

1. Log on to your Oracle Internet of Things Cloud Service instance.

2. Click **Menu** and click **Applications**.
3. Click an application to integrate with your Oracle Business Intelligence Cloud Service instance.
4. Create the integration:
    a. Click **Integration**.
    b. Click **Create Integration** if this is your first integration, or click **Add** if you have existing integrations.
    c. Select **Business Intelligence Cloud Service**.
    d. Complete these fields:
        - **Name** Enter a name for the integration.
        - **Description** (Optional) Enter an optional description for the integration.
        - **URL:** Enter the URL for the Oracle Internet of Things Cloud Service instance.
        - **Identity Domain:** Enter the identity domain for the Oracle Internet of Things Cloud Service instance.
        - **Username:** Enter the user name for the Oracle Internet of Things Cloud Service instance.
        - **Password:** Enter the password for the Oracle Internet of Things Cloud Service instance.
    e. (Optional) Click **Verify Connectivity** to test the connection between the Oracle Internet of Things Cloud Service instance and the target service or application.

        The connectivity test can include:

        - DNS Resolution Test: Checks whether the server name resolves to an IP address.
        - Connectivity Test: Checks whether the IP address is reachable.
        - IP/Port test: Checks whether you can talk to the target URL on the specified port.
        - SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.
        - Authentication Test: Checks whether you can exchange messages with the target.
    f. Click **Create**.
5. Select the integration you created in step 4 and then click **Edit** .
6. Define the data stream for the Oracle Internet of Things Cloud Service instance:
    a. Click **Streams** tab.
    b. Click **Create a Stream**.
    c. In the **BICS Table Name** field, enter the name of the Oracle Business Intelligence Cloud Service table that will hold the Oracle Internet of Things Cloud Service device data.
    d. Select a message format for the data stream in the **Message Format** list.

e. Select the metadata types to add to the outgoing stream in the **Annotations** field. An annotation is the data type that you want to retrieve from the device message stream.

f. (Optional) Click **+** to add more message formats.

g. Click **Save**.

7. Click **Synchronize Now** to synchronize the Oracle Business Intelligence Cloud Service and Oracle Internet of Things Cloud Service data.

8. Select the integration you created in step 4 and then click **Edit** ( ).

9. Click the **Synchronization** tab.

10. Select the **Automatic Synchronization** check box to enable automatic batch synchronization.

11. Select a synchronization interval in the **Synchronization Interval** list.

    Some data streams may not get updated during the synchronization process. The synchronization operation may take some time to complete. A progress indicator indicates the synchronization status.

12. Click the **Messages** tab to view data messages that are streaming from the devices that are online.

13. Click **Save** to save your changes to the Oracle Business Intelligence Cloud Service integration.

14. Log on to your Oracle Business Intelligence Cloud Service instance and confirm the table you created in step 6 is being populated with Oracle Internet of Things Cloud Service data.

Integrate Oracle Analytics Cloud with Oracle Internet of Things Cloud Service

Use your existing Oracle Analytics Cloud instance to analyze device message data. To perform the analysis, use this procedure to integrate your existing Oracle Analytics Cloud with your Oracle Internet of Things Cloud Service instance.

These are the prerequisites for the successful completion of this procedure:

- An active Oracle Analytics Cloud instance.
- An active Oracle Internet of Things Cloud Service instance.
- An Oracle Internet of Things Cloud Service application.
- A device model associated with the Oracle Internet of Things Cloud Service application. The device model determines what message format can be used in the data stream. See Assigning a Device Model to a Cloud Service.

1. Log on to your Oracle Internet of Things Cloud Service instance.

2. Click **Menu** and click **Applications**.

3. Click an application to integrate with your Oracle Analytics Cloud instance.

4. Create the integration:

   a. Click **Integration**.

b. Click **Create Integration** if this is your first integration, or click **Add** if you have existing integrations.

c. Select **Oracle Analytics Cloud Service**.

d. Complete these fields:

- **Name** Enter a name for the integration.
- **Description** (Optional) Enter an optional description for the integration.
- **URL:** Enter the URL for the Oracle Internet of Things Cloud Service instance.
- **Identity Domain:** Enter the identity domain for the Oracle Internet of Things Cloud Service instance.
- **Username:** Enter the user name for the Oracle Internet of Things Cloud Service instance.
- **Password:** Enter the password for the Oracle Internet of Things Cloud Service instance.
- **Connection Name:** Enter a name to identify the Oracle Analytics Cloud connection.

e. (Optional) Click **Verify Connectivity** to test the connection between the Oracle Internet of Things Cloud Service instance and the target service or application.

The connectivity test can include:

- DNS Resolution Test: Checks whether the server name resolves to an IP address.
- Connectivity Test: Checks whether the IP address is reachable.
- IP/Port test: Checks whether you can talk to the target URL on the specified port.
- SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.
- Authentication Test: Checks whether you can exchange messages with the target.

f. Click **Create**.

5. Select the integration you created in step 4 and click **Edit**  .

6. Define the data stream for the Oracle Internet of Things Cloud Service instance:

a. Click **Streams** tab.

b. Click **Create a Stream**.

c. In the **OACS Table Name** field, enter the name of the Oracle Analytics Cloud table that will hold the Oracle Internet of Things Cloud Service device data.

d. Select a message format for the data stream in the **Message Format** list.

e. Select the metadata types to add to the outgoing stream in the **Annotations** field. An annotation is the data type that you want to retrieve from the device message stream.

f. (Optional) Click **+** to add more message formats.

g. Click **Save**.

7. Click **Synchronize Now** to synchronize the Oracle Analytics Cloud and Oracle Internet of Things Cloud Service data.
8. Select the integration you created in step 4 and then click **Edit**  .
9. Click the **Synchronization** tab.
10. Select the **Automatic Synchronization** check box to enable automatic batch synchronization.
11. Select a synchronization interval in the **Synchronization Interval** list.

    Some data streams may not get updated during the synchronization process. The synchronization operation may take some time to complete. A progress indicator indicates the synchronization status.
12. Click the **Messages** tab to view data messages that are streaming from the devices that are online.
13. Click **Save** to save your changes to the Oracle Analytics Cloud integration.
14. Log on to your Oracle Analytics Cloud instance and confirm the table you created in step 6 is being populated with Oracle Internet of Things Cloud Service data.

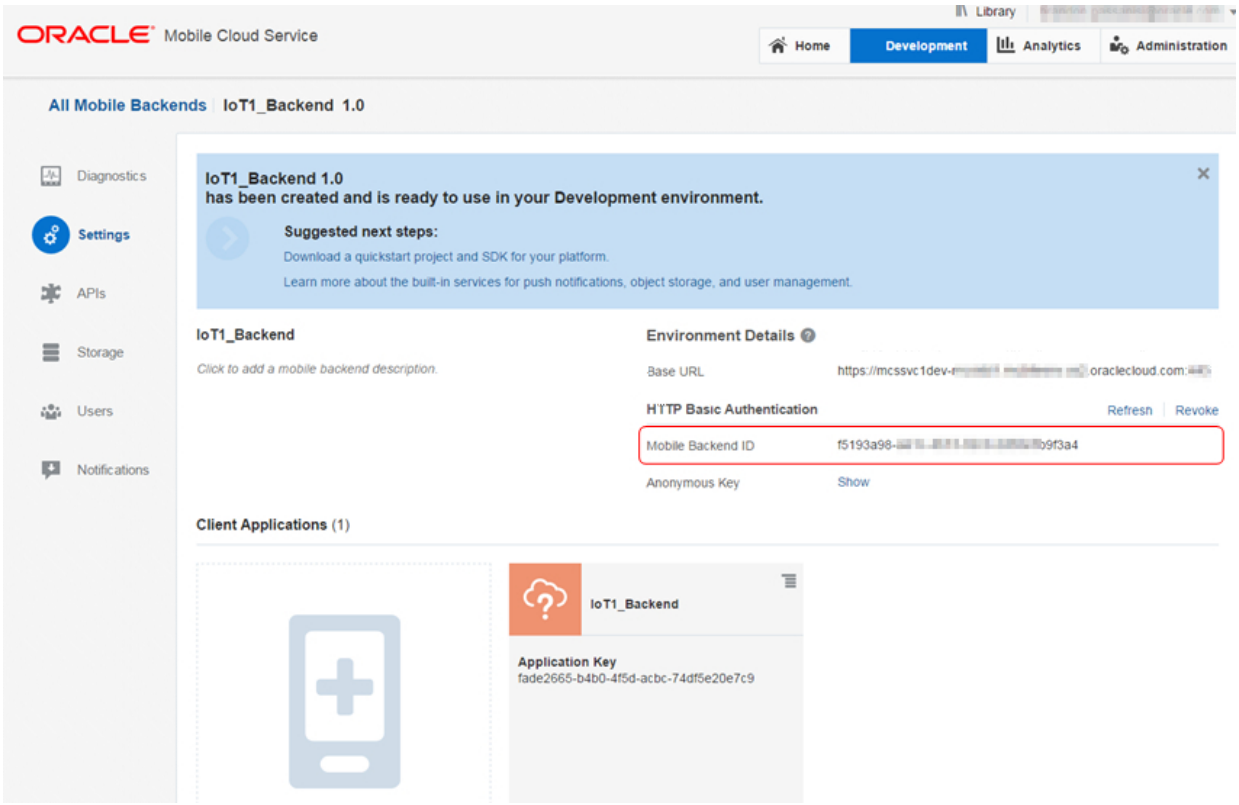
Integrate Oracle Mobile Cloud Service with Oracle IoT Cloud Service

Integrating your Oracle IoT Cloud Service instance to Oracle Mobile Cloud Service allows you to process and analyze the data received from your mobile devices. The integration requires that you already have a subscription to Oracle Mobile Cloud Service.

To create the integration to Oracle Mobile Cloud Service:

1. Create a Mobile Backend using the Developer Portal page of your Oracle Mobile Cloud Service instance.
2. Select that Mobile Backend and click **Settings** on the left navigation area.

   The Settings page is displayed, similar to the image below.



3. In the HTTP Basic Authentication section, copy and save the value that's in the **Mobile Backend ID** text field, which is circled in the above image.

   You will need this value when you set up the integration.

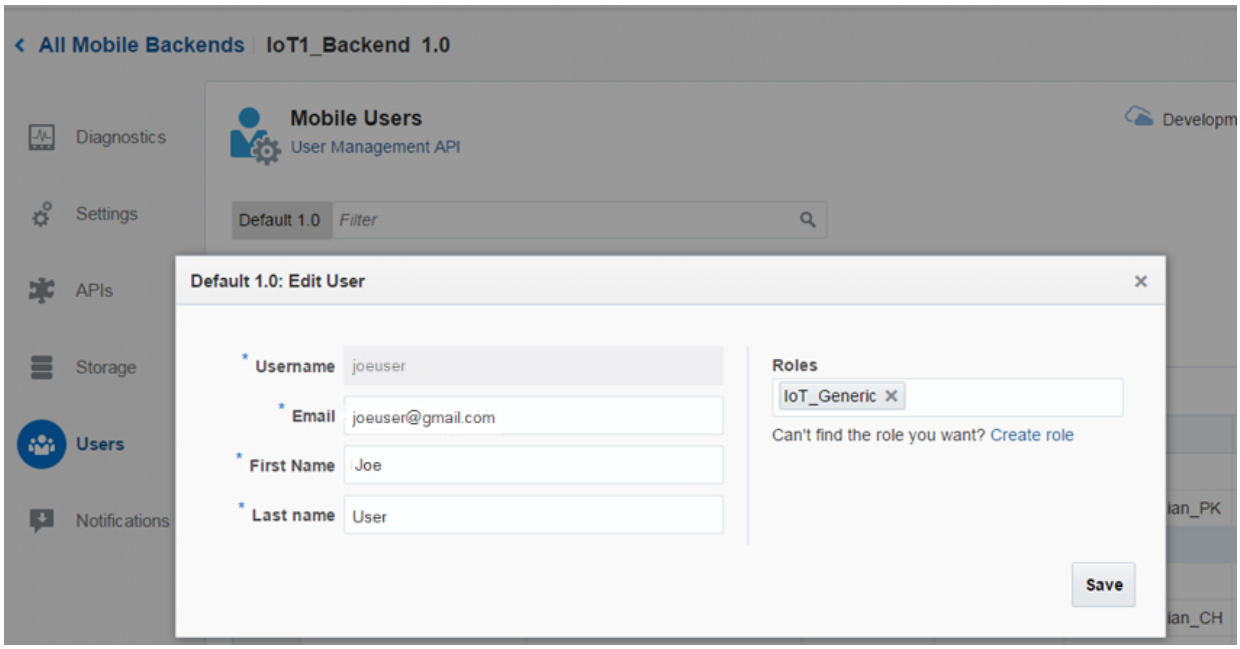
4. Create a mobile user whose credentials you will use when you configure your Mobile Cloud Service integration.

   a. Click **Users** in the left navigation area and then click **New User** on the right pane.

   b. In the New User dialog, enter the required values for **Username**, **Email**, **First Name**, and **Last Name**.

   c. In the **Roles** section, specify the`IoT_Generic` role for this new user, as shown in the image below.

      If the `IoT_Generic` isn't defined in your backend yet, click **Create role** to create the `IoT_Generic` role, as illustrated below.



5. Define a collection in your Oracle Mobile Cloud Service instance.

   This collection will provide a way to store the messages that will be received via the Oracle IoT Cloud Service integration.

   a. Click **Storage** in the left navigation area.

   b. Click **New Collection** on the right pane.

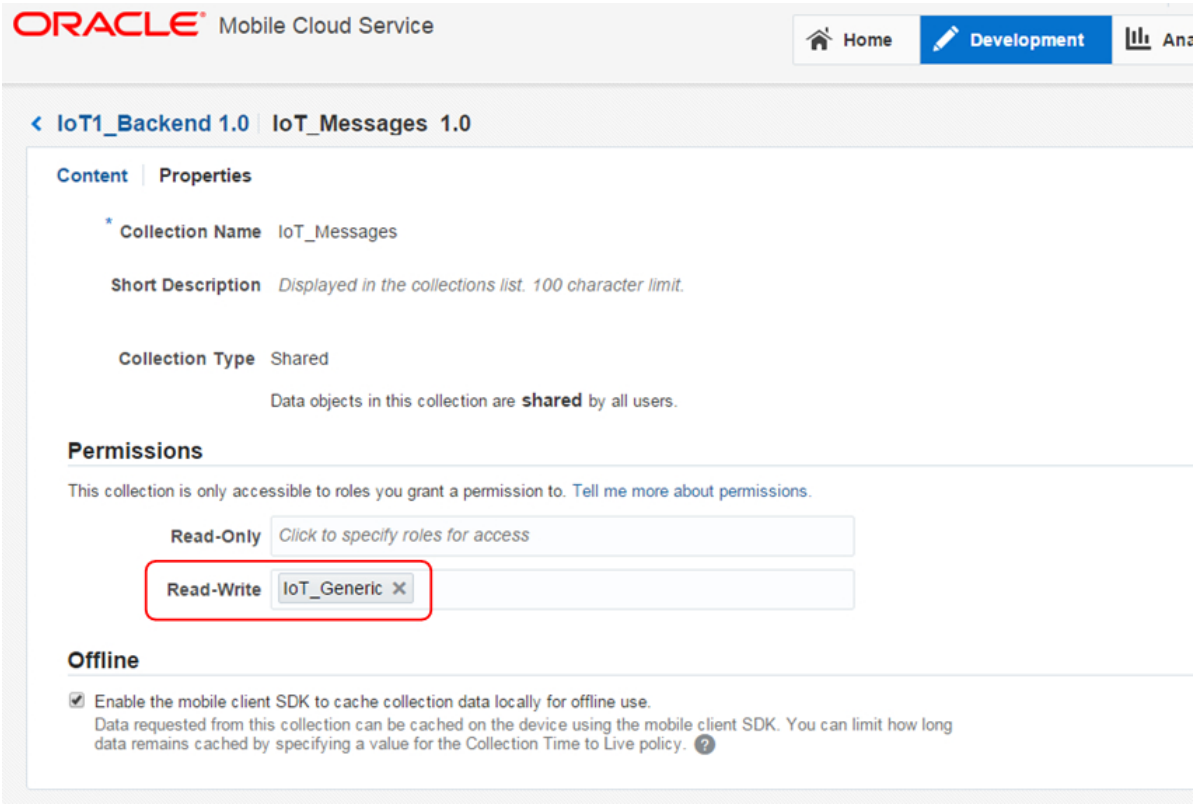
   c. Specify the required value for the **Collection Name** text field..

      For example, you can name it `IoT_Messages`.

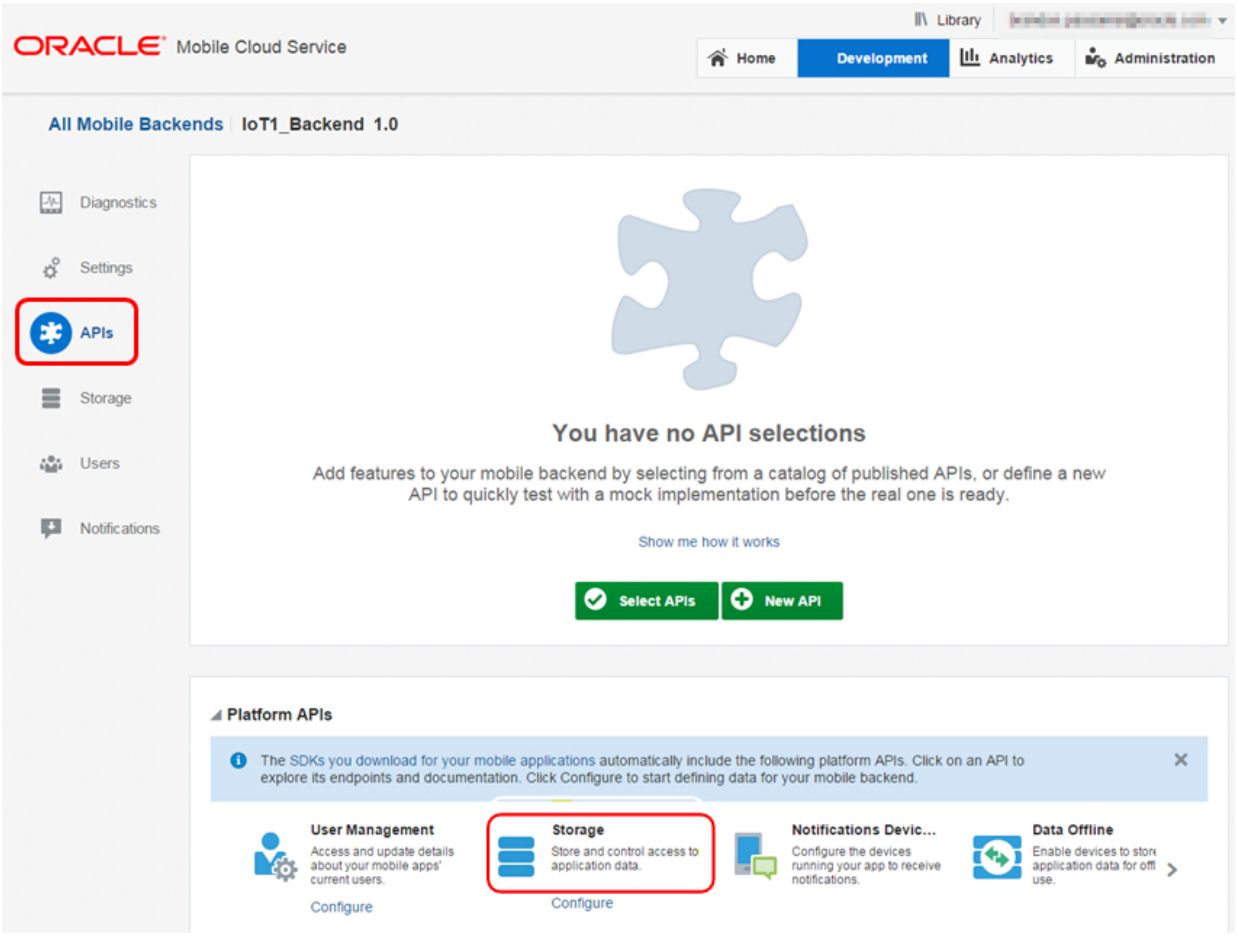
6. Specify the user role to use for the collection that you just created.

   a. In the Storage page, select the newly created collection and edit it by clicking the pencil icon on the right side.

   b. In the Permissions section, specify the `IoT_Generic` role in the **Read-Write** setting, as illustrated in the image below.



7. Determine the REST URL and payload that will be used by Oracle IoT Cloud Service to send messages to the collection that was just created.
   a. Click **APIs** in the left navigation area.
   b. Click **Storage** on the bottom of the screen, as shown below.



In the resulting page, you will see information on the REST URLs for connecting to the collection.

c. Click the **POST** link in the left pane, as shown in the image below.

The top portion of the right pane is updated with the POST URL, as circled at the top portion of the image below.

d. Copy and save the POST URL value.

You will use this value when you set up the integration between Oracle IoT Cloud Service and Oracle Mobile Cloud Service.

8. In Management Console UI of your Oracle IoT Cloud Service instance, click **Applications**.

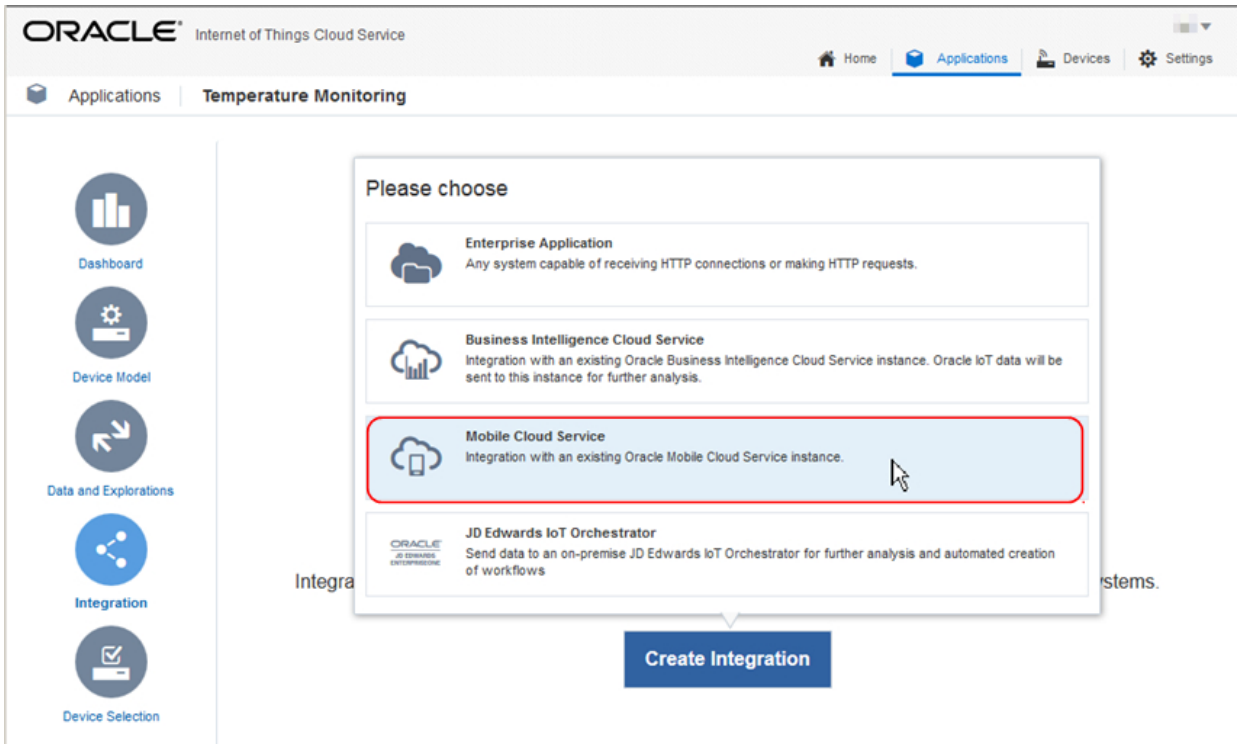
   Integrations are configured within the scope of an IoT Application.

9. Click the name of the IoT Application that you want to integrate with Oracle Mobile Cloud Service or create a new one, if necessary and click **Integration** on the left navigation area

10. Configure the integration with Oracle Mobile Cloud Service.

    a. Click **Create Integration**.

    b. Select Mobile Cloud Service, as shown in the image below.



    c. In the Mobile Cloud Service dialog, pictured below, enter all the required values.

    - **Name** - name of your choice for your integration
    - **Description** - optional value of your choice
    - **URL** - the POST URL you obtained in step 6c above.
    - **Username** and **Password** - the mobile user credentials you created in step 4 above.
    - **MCS Backend ID** - the Mobile Backend ID you obtained in step 3 above.

d. (Optional) Click **Verify Connectivity** to test the connection between Oracle Internet of Things Cloud Service instance and the target service or application.

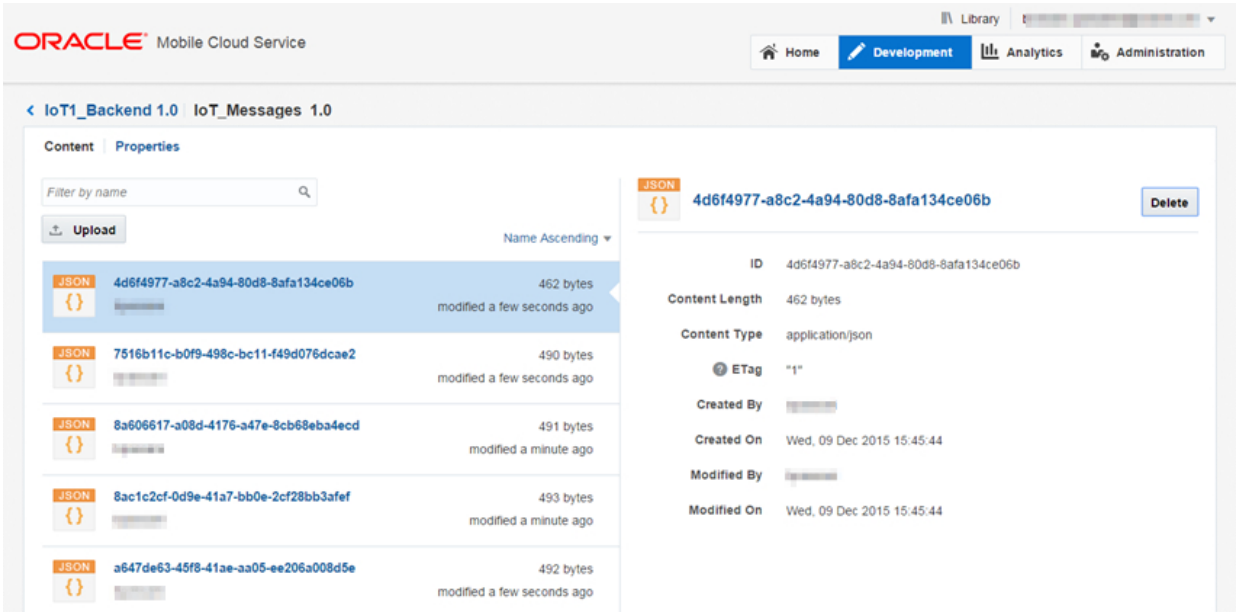
   The connectivity test can include:

   - DNS Resolution Test: Checks whether the server name resolves to an IP address.
   - Connectivity Test: Checks whether the IP address is reachable.
   - IP/Port test: Checks whether you can talk to the target URL on the specified port.
   - SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.
   - Authentication Test: Checks whether you can exchange messages with the target.

e. Click **Create**

11. After you've completed the integration set up and your IoT application is completely configured, verify that the messages that originated from one of the registered devices was sent to your Oracle IoT Cloud Service instance, processed by an IoT Application containing an Oracle Mobile Cloud Service integration, and sent to the Oracle Mobile Cloud Service collection that you've set up above.

    The image below shows what a collection that was received from Oracle IoT Cloud Service instance would look like when viewed via the Storage page of an Oracle Mobile Cloud Service instance.



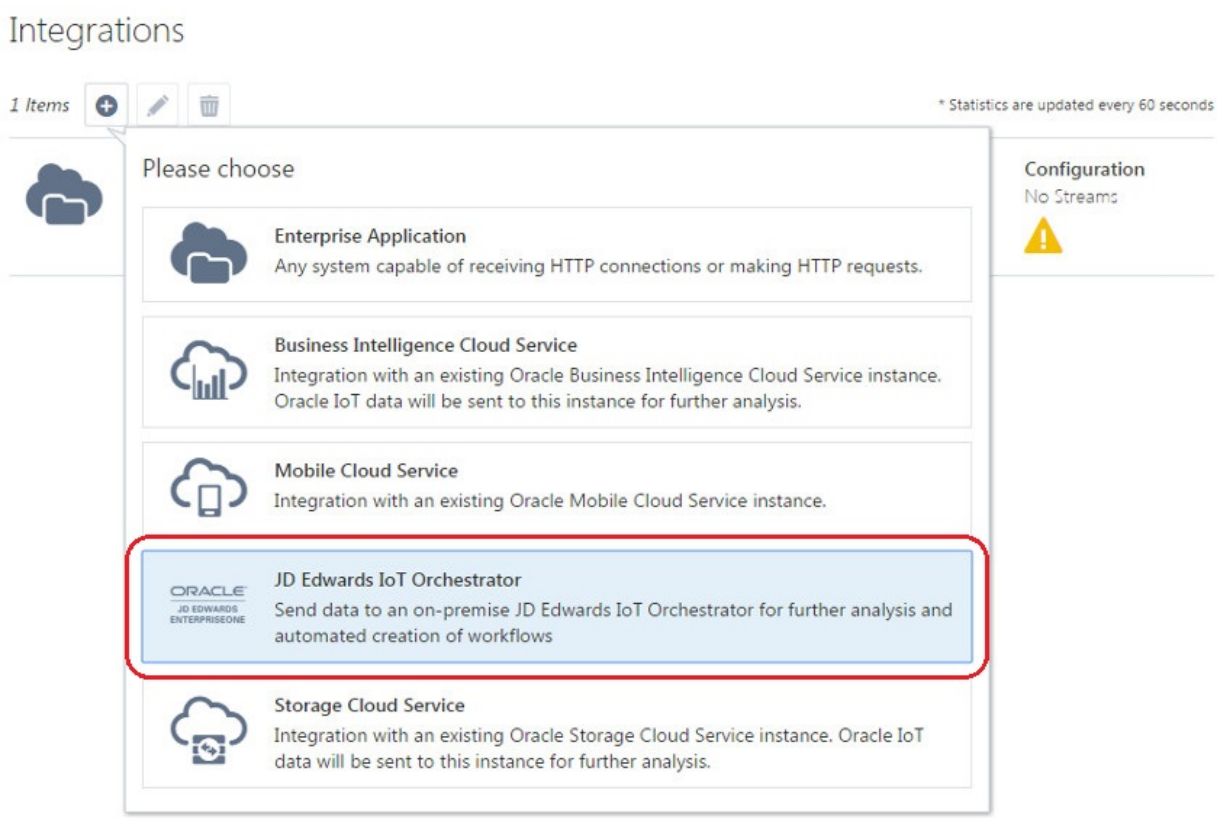
The integration with Oracle Mobile Cloud Service is now configured. If you run into issues, make sure that you've correctly entered the values you obtained from the set up steps you performed using the Oracle Mobile Cloud Service instance.

Integrate Oracle IoT Cloud Service with JD Edwards EnterpriseOne IoT Orchestrator

You can integrate Oracle IoT Cloud Service with an already existing JD Edwards EnterpriseOne instance with IoT Orchestrator by defining a new integration in your IoT Application.

Before setting up the integration, you must already have the access information for the JD Edwards EnterpriseOne Internet of Things (IoT) Orchestration that you want to integrate.

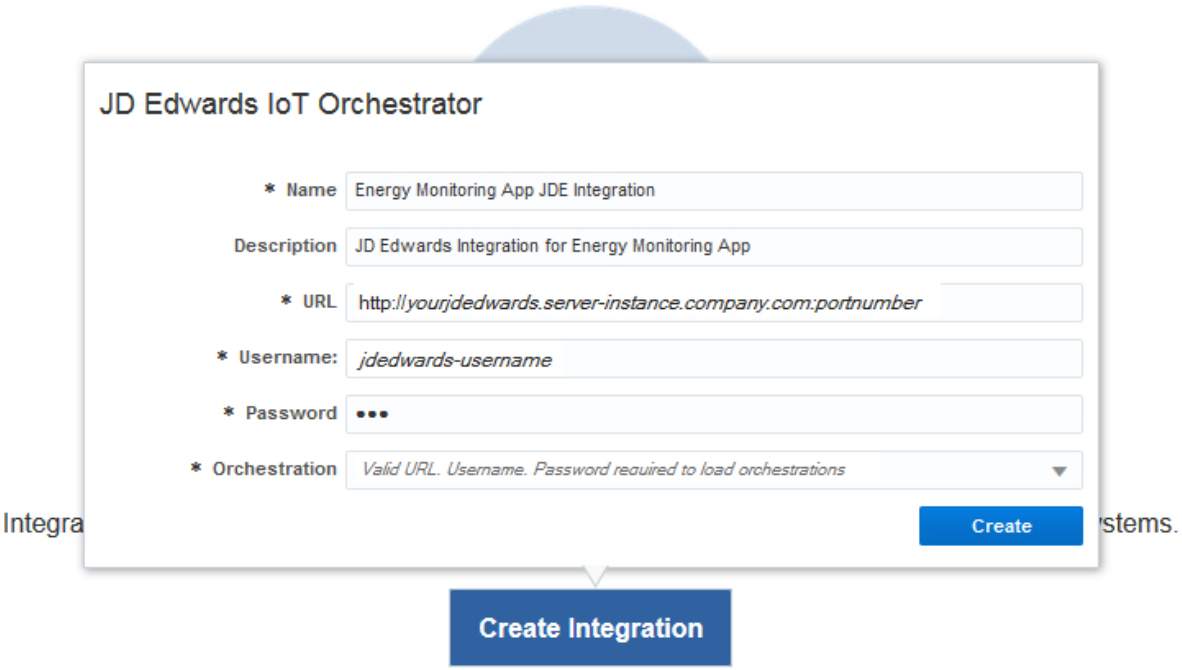
1. From your IoT Application's page, click **Integration** on the left navigation area.
2. If you don't have any integration defined yet, click **Create Integration**. If you already have an existing integration, click the **+** icon to add a new one.
3. From the pop-up dialog, click **JD Edwards IoT Orchestrator**, as shown in the image below.



4. Enter the mandatory values for the following fields.

   These fields are required to obtain the list of available orchestrations to use with the integration you're configuring.

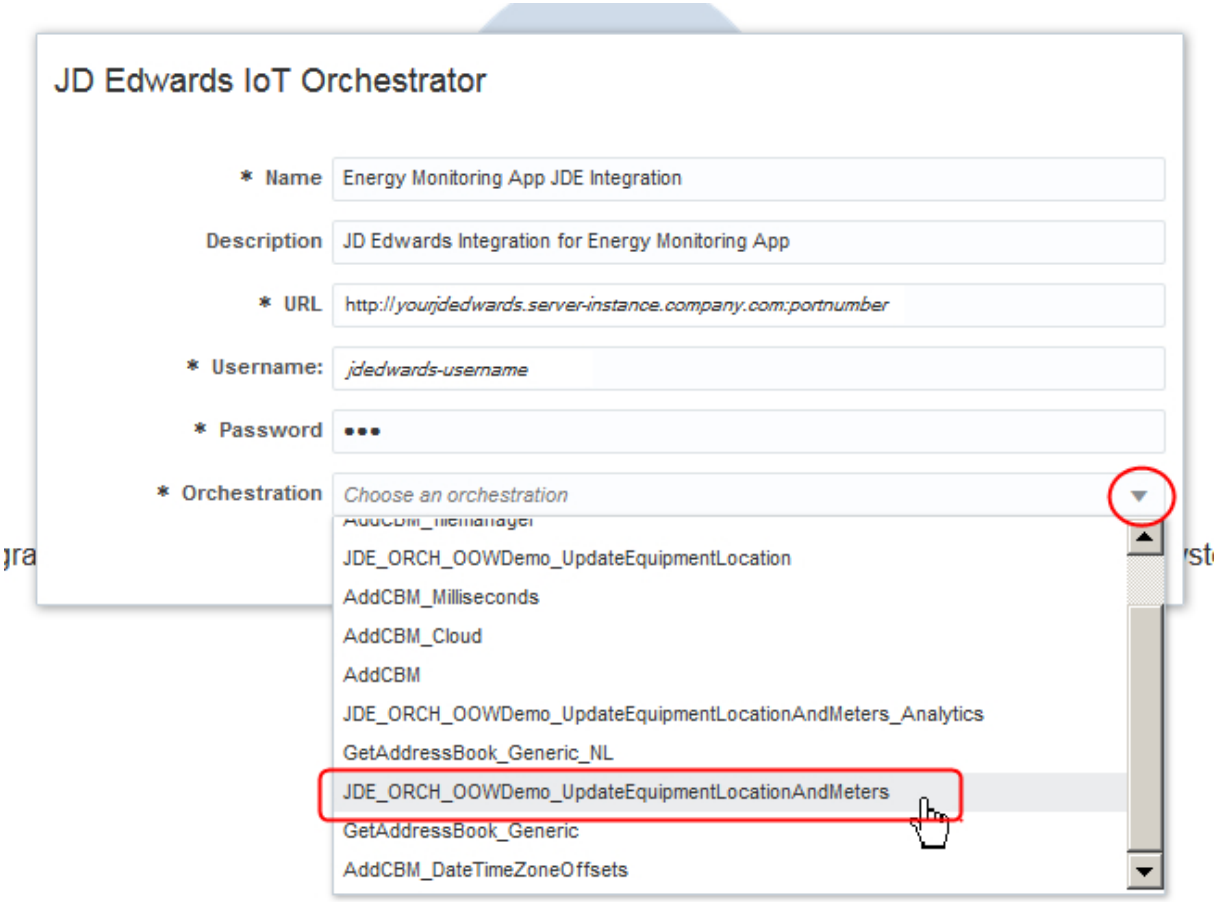
   - **Name** - This is a unique name of your choice for your JD Edwards IoT Orchestrator integration.
   - **URL** - This value is the URL and port number that you need to access the JD Edwards EnterpriseOne instance with IoT Orchestrator features that you want to integrate with your IoT Application.
   - **Username** and **Password** - These are the user credential values required to access the JD Edwards IoT Orchestrator instance.



After you entered the valid URL, username, and password values for your JD Edwards EnterpriseOne instance, Oracle IoT Cloud Service makes an attempt to connect to that specified JD Edwards EnterpriseOne instance. It tries to retrieve the list of available orchestrations that you can use for the integration. The list retrieved is used to populate the **Orchestration** drop list..

> **Note:**
>
> Only the orchestrations that have been designed using the generic input message format are retrieved from your JD Edwards EnterpriseOne instance. See Understanding the JD Edwards EnterpriseOne Internet of Things Orchestratorin *JD Edwards EnterpriseOne Tools Internet of Things Orchestrator Guide* for more information about the supported input message formats and designing an orchestration.

5. Click in the **Orchestration** text field and choose an orchestration from the displayed list, as illustrated in the image below.



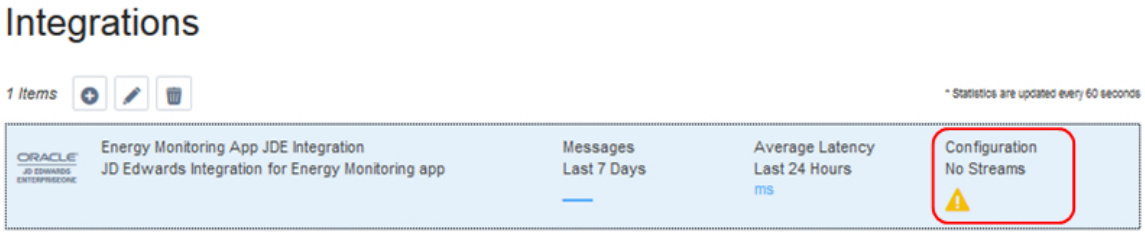
6. (Optional) Click **Verify Connectivity** to test the connection between theOracle Internet of Things Cloud Service instance and the target service or application.

   The connectivity test can include:

   - DNS Resolution Test: Checks whether the server name resolves to an IP address.
   - Connectivity Test: Checks whether the IP address is reachable.
   - IP/Port test: Checks whether you can talk to the target URL on the specified port.
   - SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.
   - Authentication Test: Checks whether you can exchange messages with the target.

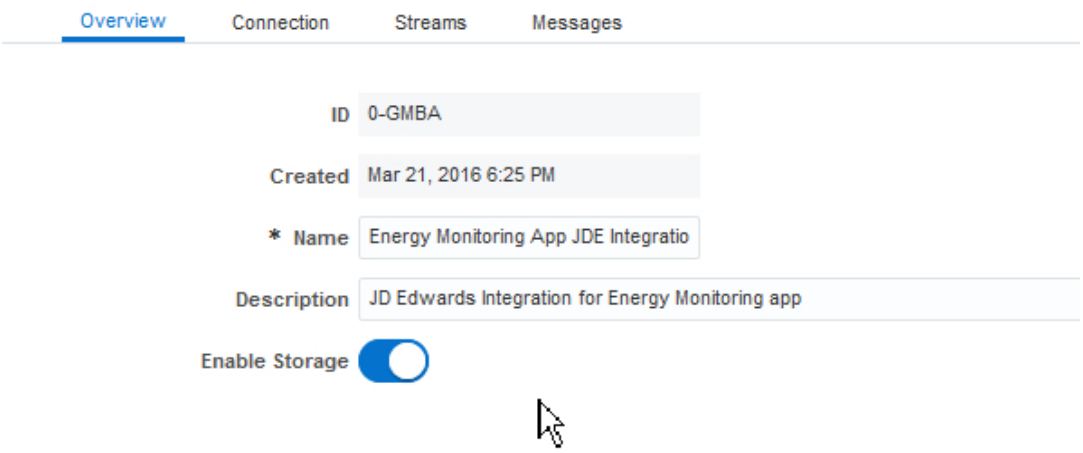
7. Click **Create**.

   The new integration is created and added to the table in the **Integrations** page, as illustrated in the image below.



The leftmost column shows the type of integration, which is JD Edwards EnterpriseOne. The next column shows the name and description, if any, that you assigned to the integration. The next column shows the number of messages received in the last 7 days through this integration and the next column gives the average latency in the last 24 hours. Currently, these columns don't have any numbers since this is a new integration. The last column, which is circled in red in the above image, gives information about the status of the integration. It shows whether the integration is complete or not. In this case, the yellow warning icon indicates that the configuration is incomplete because there were no data streams set up for the integration.

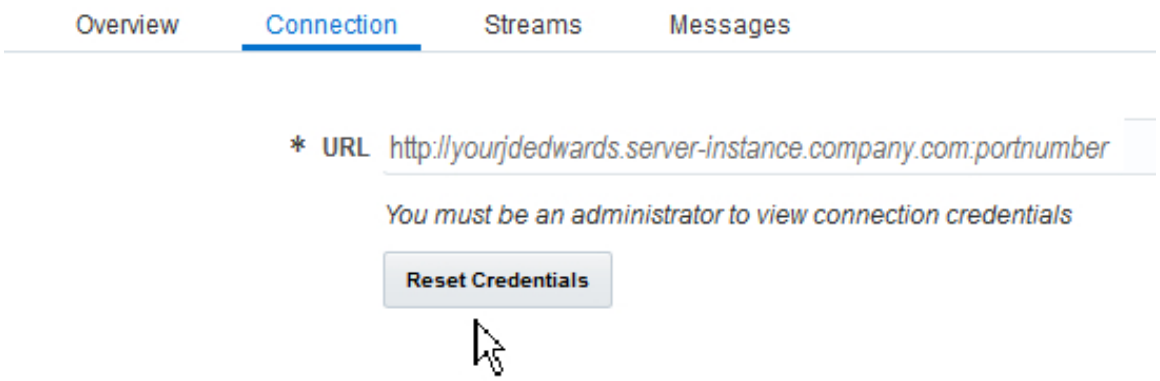
8. Examine the integration you just created.
   a. With the row for the integration selected in the Integrations page, click the pencil icon, located above the table, to edit the integration.
   b. Notice that the upper half of the integration page does not have any data displayed since you just created the integration.
   c. On the bottom half of the page, the **Overview** tab is displayed, as illustrated by the image below:



- **ID** is the unique ID assigned to this JD Edwards EnterpriseOne Tools IoT Orchestrator . integration.
- **Created** is the date when the integration was created..
- **Name** and **Description** are the values you provided when you defined the integration.
- The **Enable Storage**switch is turned on by default. This means that all data messages sent through this integration, via a pre-defined stream, are saved in storage in your Oracle IoT Cloud Service instance. This enables you to see all the data messages that were sent by a particular stream. If you turn the **Enable Storage** switch off, you can no longer ask for messages that were previously sent to your JD Edwards EnterpriseOne instance via the pre-defined stream.

9. Check the connection information.
   a. Click the **Connection** tab.

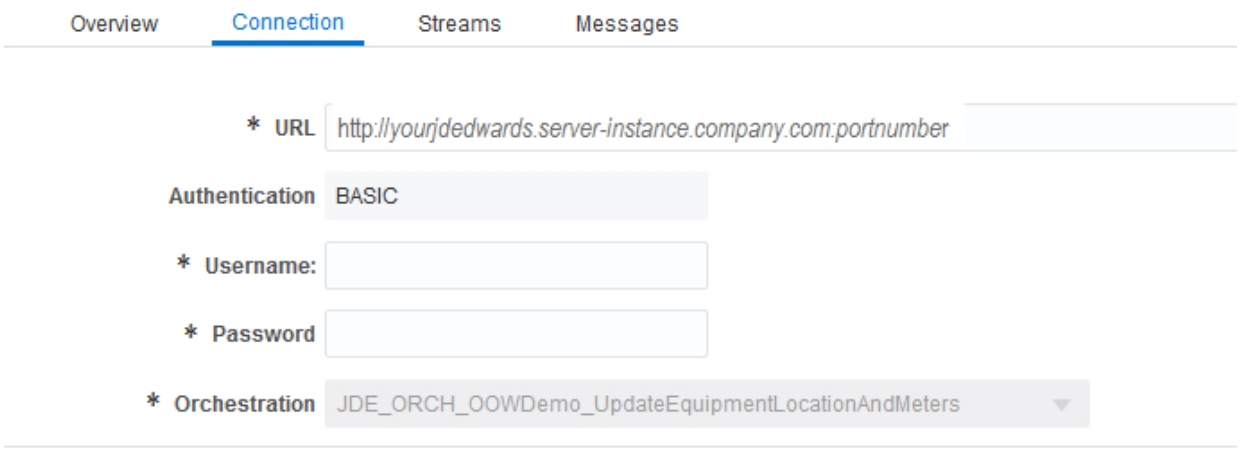
      You will see information similar to the following image:



      The URL is the one you provided when you defined the integration.

   b. Click **Reset Credentials**.

      Additional text fields for Authentication, Username, Password, and Orchestration are displayed, as shown in the image below.



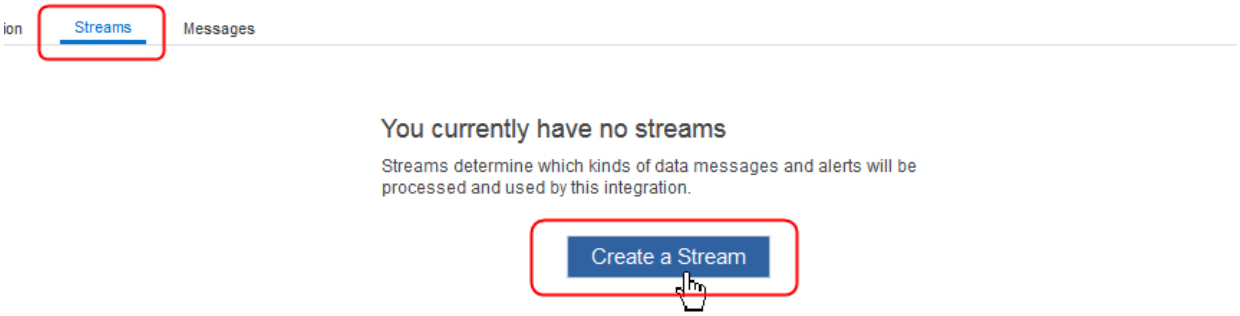
You can modify the values that are currently used, if you want.

c. Re-type the Username and Password values you entered when you defined the integration, or enter new valid values.

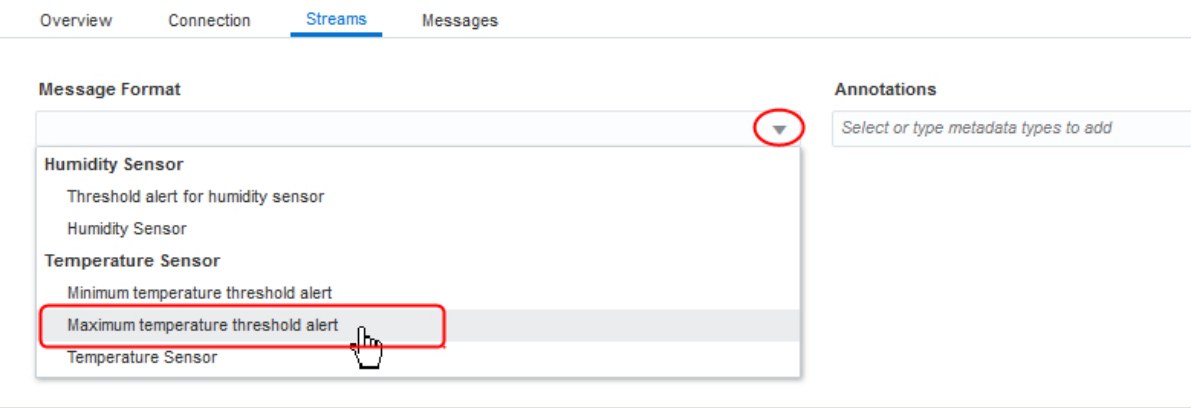
10. Complete the configuration by creating a stream for the integration.

a. Click **Streams**.

b. Click **Create a Stream**, as illustrated in the image below.



c. Click in the **Message Format** text field and select the message format from the list of available displayed, as illustrated in the image below.



d. (Optional) If you want to add annotations to the stream, click in the **Annotations** text field and choose one or more items from the list.

   If you previously created custom metadata keys for your devices and streams, then these also appear in the list of annotations.

   You can also type a new annotation name and select the data type for the annotation.

11. Select **Messages** to view the messages that have been sent through this integration, if any.

    a. Check the **Auto Refresh Interval** check box to refresh the table of messages being sent or received using the integration and pre-defined stream.

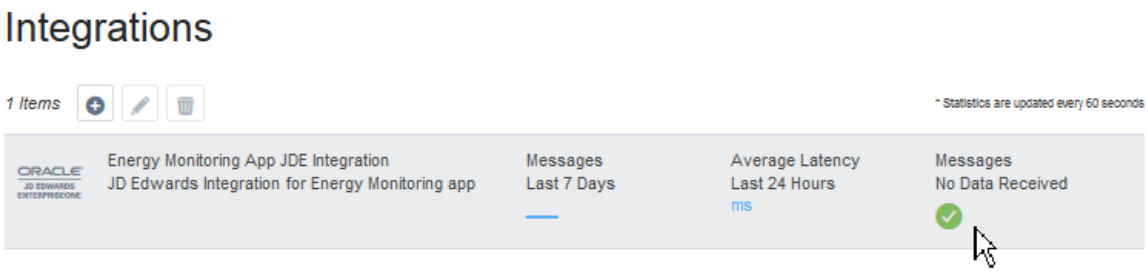
    b. Change the frequency of the refresh, if you want, by clicking the drop list arrow and choosing a new time interval. The default interval is every 5 seconds.

       Notice that the table flashes as it updates.

    c. Deselect **Auto Refresh Interval**.

12. Click **Save**.

    The **Integrations** page is displayed. Notice that the right-most column in the new JD Enterprise integration's row is updated, as shown in the image below. The green circle with the check mark indicates that the integration is now configured properly. In this case, however, no data has been received yet.



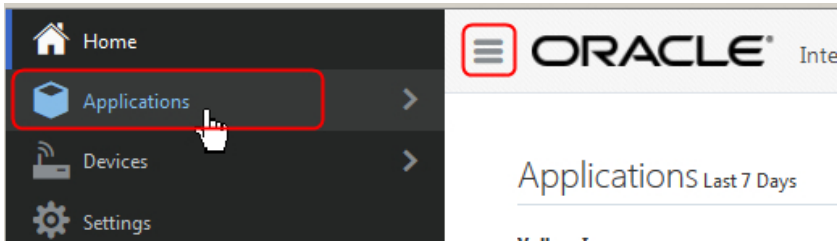
You've completed configuring your JD Edwards EntepriseOne IoT Orchestrator integration.

Integrate Oracle Storage Cloud Service with Oracle IoT Cloud Service

An integration with your Oracle Storage Cloud Service instance enables you to store the streams of device data sent by your devices to your Oracle IoT Cloud Service instance using CSV or JSON formatted files.

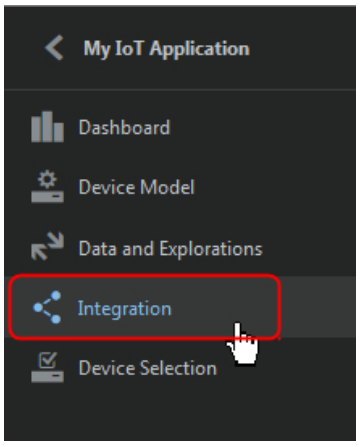
You must already have access to an Oracle Storage Cloud Service instance before you can define an integration with your IoT application.

1. From the Management Console's **Home** menu, click **Applications**.

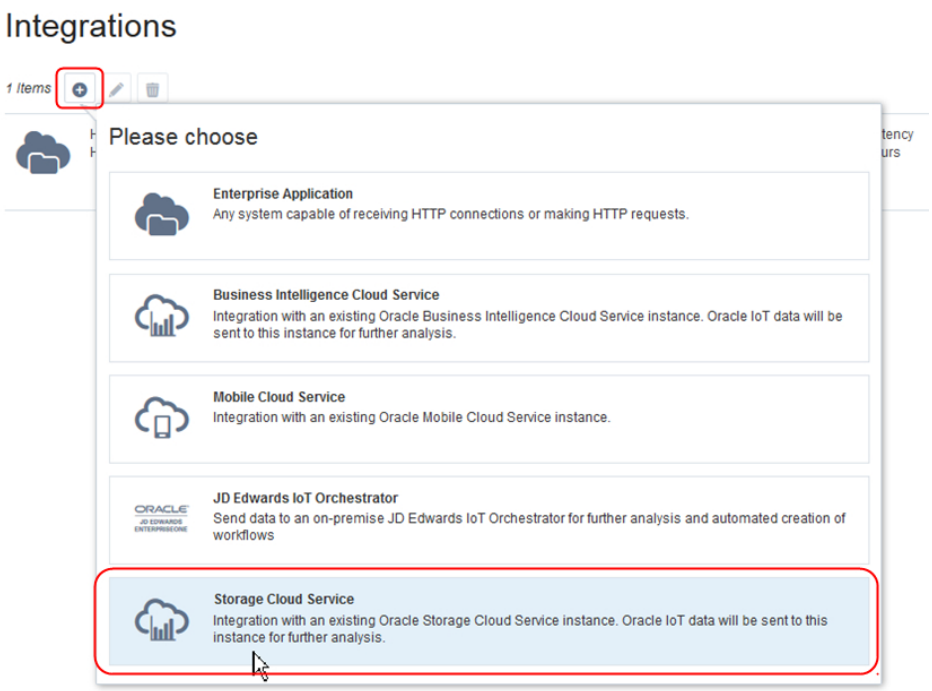


   The Applications menu is displayed and lists all the existing IoT applications.

2. Click the name of the existing IoT application that you want to integrate with your Oracle Storage Cloud Service instance.
3. Create the integration with your Oracle Storage Cloud Service instance.
   a. Click **Integration** on the left navigation area, as shown below.



   b. If you don't have any integration configured yet, click **Create Integration**. If you have at least one integration already set up, click **Add** (**+** icon).
   c. Select **Storage Cloud Service**, as shown in the image below.



d. In the Storage Cloud Service dialog, enter the information required to create the integration with your Oracle Storage Cloud Service instance.

   - **Name** - alphanumeric name of your choice for the integration
   - **URL** - REST API endpoint URL of your Oracle Storage Cloud Service instance.
   - **Identity Domain** - the identity domain for your Oracle Storage Cloud Service instance .
   - **Username** - the username assigned to you for your Oracle Storage Cloud Service subscription.
   - **Password** - the password assigned to you for your Oracle Storage Cloud Service subscription.
   - **Container Name** - alphanumeric name for the folder that will store your data. Clicking in the field provides a default name of `MyStorageCSIntegrationContainer`.
   - **Data Format** - the data format to use when storing the data. Use the drop list to choose either a CSV or a JSON format.

e. (Optional) Click **Verify Connectivity** to test the connection between Oracle Internet of Things Cloud Service instance and the target service or application.

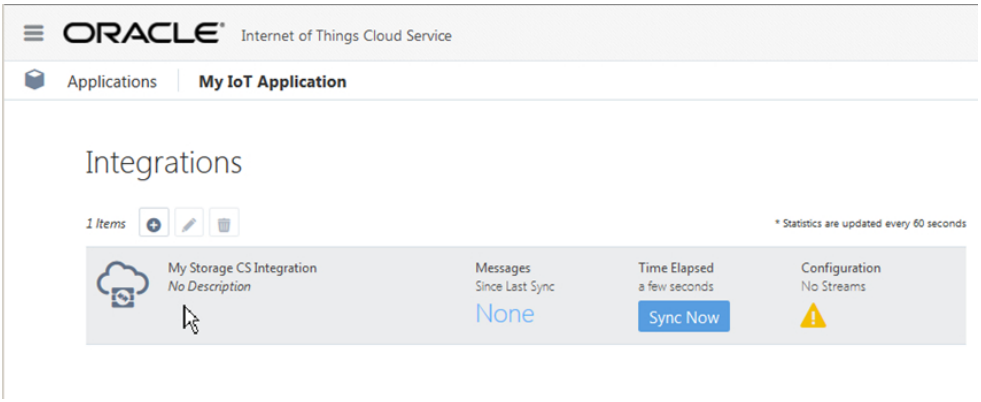
   The connectivity test can include:

   - DNS Resolution Test: Checks whether the server name resolves to an IP address.
   - Connectivity Test: Checks whether the IP address is reachable.
   - IP/Port test: Checks whether you can talk to the target URL on the specified port.
   - SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.

- Authentication Test: Checks whether you can exchange messages with the target.

f. Click **Create**.

The new integration is created and a new row is listed in the **Integrations** page, as shown in the image below.



4. Configure the data streams to use for the new integration
   a. From the Integrations page, select the newly created Oracle Storage Cloud Service integration.
   b. Click **Streams**.
   c. Click **Create Stream**.
   d. From the **Message Format** drop list, select a message format to use for the stream.
   e. In the **Object Base Name** text field, type the base name to use when creating the files that will contain the data that will be sent to Oracle Storage Cloud Service.

      The Object Base Name can be no more than 30 ASCII characters in length.
   f. Click in the **Annotations** text field if you want to add an annotation. Select from the list of metadata types to add to the outgoing stream.

      If you previously created custom metadata keys for your devices and streams, then these also appear in the list of annotations.

      You can also type a new annotation name and select the data type for the annotation.
   g. Click the **+** icon if you want to add another stream.

      An integration can have multiple streams, but only one message format used per stream.
   h. Click **Save**.
5. Back in the **Integrations** page, select the row for the Oracle Storage Cloud Service integration again.
6. Click **Synchronize Now** to start synchronizing the data.

   The synchronization of data from Oracle IoT Cloud Service to Oracle Storage Cloud Service can be done manually or automatically. Automatic synchronization can be set by specifying a synchronization schedule ranging from 1 hour to 1 day.

Once the process has been triggered, Oracle IoT Cloud Service begins to synchronize data that is available for each data stream that was created since the last successful synchronization event in Oracle IoT Cloud Service.

7. Click the **Synchronization** tab to automatically synchronize the data.
8. Click the **Automatic Synchronization** check box to enable automatic batch synchronization. If the check box is unchecked, it indicates that automatic synchronization is disabled.
9. Click the **Synchronization Interval** drop down list and select a value to specify the interval at which data is to be synchronized. There are different time intervals ranging 1 hour to 1 day.

   Note that there is a possibility that even though the entire synchronization process has been completed, there may be certain data streams that don't get updated as part of that process.

   The synchronization operation may take some time to complete. A progress indicator will be displayed to give you status.
10. Click **Messages** to view any messages that are being sent through the integration.

    Once the synchronization is complete, the **Messages** tab is updated with data messages that are streaming from the devices that are online.
11. Click **Save** to store any changes you made in the Oracle Storage Cloud Service integration.

Integrate Oracle Big Data Cloud Service with Oracle Internet of Things Cloud Service

Integrate an Oracle Internet of Things Cloud Service application with Oracle Big Data Cloud Service to send Oracle Internet of Things Cloud Service data to Oracle Big Data Cloud Service. Data is first sent to Oracle Storage Cloud Service and then to Oracle Big Data Cloud Service.

These are the prerequisites for the successful completion of this procedure:

- An active Oracle Big Data Cloud Service instance.
- An active Oracle Internet of Things Cloud Service instance.
- An active Oracle Storage Cloud Service instance.
- An Oracle Internet of Things Cloud Service application.
- A device model associated with the Oracle Internet of Things Cloud Service application.
- The message format associated with the device model is associated with stream.
- WinSCP open source SFTP and FTP client for Microsoft Windows (or similar).
- PuTTY SSH and telnet client (or similar).
- A device simulator.

1. Open the **Oracle Internet of Things Cloud Service** Management Console.
2. Click the **Menu** () icon adjacent to the Oracle Internet of Things Cloud Service title on the Management Console.
3. Click **Applications**.

4. Click an application.
5. Click **Integration**.
6. Select one of these options:
   - If you have not previously created an integration, click **Create Integration** and then select **Big Data Cloud Service**.
   - If you have previously created an integration, click the **Add** (⊕) icon and then select **Big Data Cloud Service**.
7. Complete these fields:
   - **Name**: Enter a unique name for the Oracle Big Data Cloud Service integration.
   - **Description**: Enter an optional description for the Oracle Big Data Cloud Service integration.
   - **URL**: Enter the URL for the Oracle Storage Cloud Service instance.
   - **Identity Domain**: Enter the identity domain for the Oracle Cloud Infrastructure Object Storage Classic instance.
   - **Username**: Enter the user name used to access the Oracle Cloud Infrastructure Object Storage Classic instance.
   - **Password**: Enter the password used to access the Oracle Cloud Infrastructure Object Storage Classic instance.
   - **Container Name**: Accept the default or enter a unique name for the Oracle Cloud Infrastructure Object Storage Classic container that stores Oracle Internet of Things Cloud Service data.
8. (Optional) Click **Verify Connectivity** to test the connection between Oracle IoT Cloud Service and Oracle Storage Cloud Service.

   Common checks include the following:
   - DNS Resolution Test: Checks whether the server name resolves to an IP address.
   - Connectivity Test: Checks whether the IP address is reachable.
   - IP/Port test: Checks whether you can talk to the target URL on the specified port.
   - SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.
   - Authentication Test: Checks whether you can exchange messages with the target.
9. Click **Create**.
10. Add a stream to the integration:
    a. Select the integration in the **Integrations** list and then click the **Edit** ( ) icon.
    b. Click the **Streams** tab.
    c. Click **Create a Stream**.
    d. Select a message format in the **Message Format** list.

The same message format cannot be used by multiple streams. Select a unique message format for each stream you create.

e. Enter an object base name in the **Object Base Name** field. The object base name is the name used to identify the data files in the Oracle Cloud Infrastructure Object Storage Classic container. The naming convention for the data files includes the object base name and the creation date and time. For example, if the object base name is **HVAC**, the data file name is `HVAC-2016-11-21-11-36-12.268.json`.

f. Select annotations in the **Annotations** list. Annotations identify the metadata sent to Oracle Cloud Infrastructure Object Storage Classic.

If you previously created custom metadata keys for your devices and streams, then these also appear in the list of annotations.

You can also type a new annotation name and select the data type for the annotation.

g. Click the **Add** ( ) icon to add another stream or click **Save** to save the stream.

11. Select **Download BDCS Scripts** to download a zip file containing scripts for importing data from Oracle Storage Cloud Service to Oracle Big Data Cloud Service. These scripts are customized to use streams that are configured with the integration and contain HiveQL and ODCP (Oracle Distributed Copy) commands for importing data:

a. Select the integration in the **Integrations** list and then click the **Edit** ( ) icon.

b. Click the **Streams** tab.

c. Click **Download BDCS Scripts**.

d. Browse to a location where you want to save the BDCS scripts and then click **Save**.

12. Send device data to the Oracle Cloud Infrastructure Object Storage Classic instance:

a. Open a client device or simulator and send messages to Oracle Internet of Things Cloud Service.

b. Select the integration in the **Integrations** list and then click the **Edit** ( ) icon.

c. Click **Synchronize Now** to send the message data to the Oracle Cloud Infrastructure Object Storage Classic instance.

13. Configure Oracle Big Data Cloud Service for importing Oracle Cloud Infrastructure Object Storage Classic data:

a. Identify the IP address of the BDCS Cluster node from the Big Data Cloud Service instance.

b. Connect to the Big Data Cloud Service node as **opc** user by using PuTTY on Windows or UNIX as applicable.

c. While connected to the Big Data Cloud Service node as **opc** user, use a `bda-oss-admin add_swift_cred` command to create an association between the Oracle Big Data Cloud Service instance and the Oracle Storage Cloud Service instance.

The syntax for the `bda-oss-admin` command is:

```
bda-oss-admin
    --cm-url    [Cloudera Manager URL]
    --cm-admin  [Cloudera Manager username]
    --cm-passwd [Cloudera Manager password]
    add_swift_cred
    --swift-username   [OSCS username]
```

```
--swift-password   [OSCS pwssword]
--swift-storageurl [OSCS URL]
--swift-provider [provider name to create]
-N
```

14. Add operating system accounts on the BDCS instance for user groups to be accessible through the operating system. While connected to the Big Data Cloud Service node as **opc** user, run the following commands:
    a. Run the `sudo bash` command to assign yourself as a root user.
    b. Run the `dcli -C 'groupadd group_name'` command to create a user group.
    c. Run the `dcli -C 'useradd -G group_name user_name'` to add a new user to the new group.
15. Add the user name to Kerberos KDC and to `bdacloudservice.oracle.com` by creating a principal. The command requires that the user name must match the OS user identity created in step 14.
16. Run the `kdadmin.local` command to open the Kerberos administration console followed by an `addprinc user_name` command to create the principal.

    In the following example, we substitute `my_user_name` for `user_name`. Here is a sample output:

```
bash-4.1# kadmin.local
Authenticating as principal oracle/admin@BDACLOUDSERVICE.ORACLE.COM
with password.
kadmin.local:  addprinc my_user_name
WARNING: no policy specified for
my_user_name@BDACLOUDSERVICE.ORACLE.COM; defaulting to no policy
Enter password for principal
"my_user_name@BDACLOUDSERVICE.ORACLE.COM":
Re-enter password for principal
"my_user_name@BDACLOUDSERVICE.ORACLE.COM":
Principal "my_user_name@BDACLOUDSERVICE.ORACLE.COM" created.
```

    a. Verify that the principal is created before exiting the `kadmin.local` console. Use the `listprincs` command in the `kadmin.local` console and look for the principal in the output:

```
kadmin.local:  listprincs
...(lots of data)
my_user_name@BDACLOUDSERVICE.ORACLE.COM
...(more data)
kadmin.local:  q
bash-4.1#
```

    b. Exit out of the `kadmin.local` console before proceeding to the next step.
    c. Run the `sudo -u user_name kinit` command to obtain a Kerberos ticket by while connected to the Big Data Cloud Service node as **opc** user.
    d. Enter the password for your cluster when the password prompt appears.
17. In a browser window, open Hue and login as an administrator.

18. Create an Oracle IoT Cloud Service user in Hue by entering these details:
    a. First and last names
    b. Email ID.
    c. User name as specified in step 14.
19. Use the Cloudera Manager to add a jar file to the classpath:
    a. Open Cloudera as an administrator.
    b. Click **Hive**.
    c. Click **Configuration** and enter **aux** in the search box.
    d. Note the value of Hive Auxiliary JARs directory.
    e. While connected to the Big Data Cloud Service node using PuTTY or SSH, search for the file hive-hcatalog-core.jar. Copy this file or create a soft-link to this file within the Hive Auxiliary JARs directory noted in the previous step.

    > **Note:**
    >
    > Back in Cloudera Manager, it may be necessary to add the location of hive-hcatalog-core.jar to the three options below the Hive Auxillary JARs Directory option as shown below. These options are named
    >
    > - Hive Service Advanced Snippet (Safety Valve) for hive-site.xml.
    > - Gateway Client Environment Advanced Configuration Snippet (Safety Valve) for hive-esv.sh.
    > - Hive Advanced Configuration snippet (Safety Valve) for hive-site.xml

    f. Restart the HiveServer2, Hive, and Hue services.
20. In a browser window, open **Hue** and login using the user name.
    a. Open File Browser.
    b. Create a destination folder path for incoming data. For example: `/user/user_name/data`.
    c. Create additional folder paths as required.
21. Using WinSCP open source SFTP and FTP client for Microsoft Windows (or similar), copy the zip file containing BDCS scripts to a directory on a Big Data Cloud Service node.
22. While connected to the Big Data Cloud Service node as **opc** user, extract the zip file.
23. Inspect the ODCP script file, which ends in `*_odcp-script.sh` and make any required changes prior to running the script.
24. Run the `*_odcp-script.sh script` as **opc** user to manually move data from Oracle Storage Cloud Service to Oracle Big Data Cloud Service. Provide the following details when running the `*_odcp-script.sh script` command:

    `user_name` — specified in bda-oss-admin command.

    `provider name` - specified in bda-oss-admin command.

    `data directory` – directory created earlier using **Hue**.

Example: `MyApp_MyBDCS_odcp-script.sh my_user_name myprovider /user/my_user_name/data`

25. Open **Hue** and login as `user_name` in a browser window.
    a. Open File Browser.
    b. Navigate to the destination folder path for incoming data.
    c. Verify that the data is present in the folder.
    d. Note the folder name for later use.
26. Create an alias to a `beeline` command to be used by the **oracle** user to query the Oracle Internet of Things Cloud Service data imported into Oracle Big Data Cloud Service.
    a. Connect to a Big Data Cloud Service node as **oracle** user.
    b. Run the command this command to determine the value of `TRUSTSTOREPATH` for your instance:

       ```
       bdacli getinfo cluster_https_truststore_path
       ```

    c. Run the command this command to determine the value of `TRUSTSTOREPWD` for your instance:

       ```
       bdacli getinfo
       ```

    d. Determine the value of `HIVE_NODE` for the Oracle Big Data Cloud Service instance.
    e. Determine the value of `KERBEROS_REALM` for the Oracle Big Data Cloud Service instance.
    f. Create the alias command by replacing the appropriate values shown in this code snippet:

       ```
       alias bee="beeline -u \"jdbc:hive2://<HIVE_NODE>:10000/
       default;principal=hive/
       <HIVE_NODE>@<KERBEROS_REALM>;ssl=true;sslTrustStore=<TRUSTSTOREPA
       TH>;trustStorePassword=<TRUSTSTOREPWD>\" -d
       org.apache.hive.jdbc.HiveDriver"
       ```

27. Obtain a Kerberos ticket as an **oracle** user by invoking the `kinit` command.

    Among the downloaded BDCS scripts is a script named `*_hive-create.sql`. This script contains customized HiveQL commands for creating tables that holds imported Oracle Internet of Things Cloud Service data. Inspect this file to view how the tables will be created and note the table names.
28. Run the `bee -f filename` command to view contents of the `*_hive-create.sql` file. For example, `bee -f MyApp_MyBDCS_hive-create.sql`.
29. Open **Hue** In a browser window and login as `user_name` to access Hive.
30. Run this Hive QL command to associate the table created with the folder location of the imported Oracle Internet of Things Service data:

    ```
    ALTER TABLE <table name> SET LOCATION '<folder location>';
    ```

    For example, `ALTER TABLE myappcontainer_readings SET LOCATION '/user/my_user_name/data';`

Now that the table and folder location are associated, you can now use additional HiveQL commands to query the synchronized data.

31. Go to the Big Data Cloud Service Integration page in Oracle IoT Cloud service to setup Automatic Synchronization.
    a. Select the **Synchronization** tab.
    b. Enable **Auto Sync** and choose a time interval.
32. Connect to the Big Data Cloud Service node as **opc** user by using PuTTY on Windows or UNIX as applicable to setup a cron job within Big Data Cloud Service.
33. Create a script file that contains the following contents:
    a. The export and initialization of the environment variable HADOOP_CLASSPATH.
    b. The invocation of the `_odcp-script.sh` script.

    Here is a sample `runsync.sh` file:

```
#!/bin/bash

# HADOOP_CLASSPATH may not set in the cron environment, so is set
here.
# The value was obtained by "env | grep HADOOP_CLASSPATH".
export HADOOP_CLASSPATH=/opt/oracle/oraloader-3.7.0-h2/jlib/
avro-1.7.3.jar:/opt/oracle/oraloader-3.7.0-h2/jlib/avro-mapred-1.7.3-
hadoop2.jar:/opt/oracle/oraloader-3.7.0-h2/jlib/commons-
math-2.2.jar:/opt/oracle/oraloader-3.7.0-h2/jlib/jackson-core-
asl-1.8.8.jar:/opt/oracle/oraloader-3.7.0-h2/jlib/jackson-mapper-
asl-1.8.8.jar:/opt/oracle/oraloader-3.7.0-h2/jlib/ojdbc6.jar:/opt/
oracle/oraloader-3.7.0-h2/jlib/oraclepki.jar:/opt/oracle/
oraloader-3.7.0-h2/jlib/orai18n.jar:/opt/oracle/oraloader-3.7.0-h2/
jlib/oraloader-examples.jar:/opt/oracle/oraloader-3.7.0-h2/jlib/
oraloader.jar:/opt/oracle/oraloader-3.7.0-h2/jlib/osdt_cert.jar:/opt/
oracle/oraloader-3.7.0-h2/jlib/osdt_core.jar:/opt/oracle/
oraloader-3.7.0-h2/jlib/ora-hadoop-common.jar:/opt/oracle/
oraloader-3.7.0-h2/jlib/orabalancer.jar:/opt/oracle/bda/bdcs/bdcs-
rest-api-app/current/lib-hadoop/*
hadoop classpath

LOG_DIR=/tmp/bdcs-iot-sync-logs
if [ ! -d "$LOG_DIR" ] ; then
  mkdir $LOG_DIR
fi
outputFile=$LOG_DIR/bdcs-iot-sync-`date +%Y%m%d-%H%M`

/home/my_user_name/scripts/MyApp_MyBDCS_odcp-script.sh my_user_name
myprovider /user/my_user_name/data > $outputFile 2>&1
```

34. Run the `crontab -e` command and set the frequency for the `runsync.sh` utility. For example, you can set the `runsync.sh` utility to run at the 45th minute of every hour by using the command line.

```
45 * * * * /home/opc/scripts/runsync.sh
```

35. Open **Hue** in a browser window and login as `user_name`.
36. Open File Browser and navigate to the destination folder path for incoming data.

37. Verify that the data is present in the folder.

Integrate Oracle IoT Cloud Service with Oracle Integration Cloud Service

You can integrate Oracle IoT Cloud Service with an already existing Oracle Integration Cloud Service instance by defining a new ICS integration and using REST to connect it to an integration in your IoT Application.

You must already have access to an Oracle Integration Cloud Service instance before you can integrate it with Oracle IoT Cloud Service.

1. Create an ICS Connection.
    a. From the **Connections** console, click **Create**.
    b. Select the **REST** connection adapter. The **REST Adapter** exposes an Oracle Integration Cloud Service integration flow as a REST service.
    c. Fill in the **New Connection - Information** dialog box, including the **Connection Name** and **Connection Identifier**. For **Connection Role**, use “Trigger and Invoke”. Enter a description.
    d. Click **Create**.
    e. On the **REST Connection** page, modify the appropriate **Connection Properties**. For the **Connection Type**, choose “REST API Base URL”, select an appropriate **TLS Version**, and enter a desired **Connection URL** based on the current ICS server (such as `https://myicsserver.domainname.com/send`).
    f. Enter a desired **Security** policy for the REST connection, such as “Basic Authentication”.
    g. Configure any **Agent** groups if desired.
    h. Click **Test** to test the REST connection.
    i. Once the connection has been successfully tested, **Save** the connection.
    j. Click **Close**.
2. Create an ICS Integration
    a. From the main menu, choose **Integrations**.
    b. Click **Create**.
    c. Select **Map Data**.
    d. Fill in the **Create New Integration** dialog box, including the integration name, identifier, and version. Enter a description and the package information, if applicable. Click **Create**.
3. Create a trigger.
    a. On the right side of the page, find the ICS Connection you created earlier. Drag this connection to the “Drag and Drop a Trigger” icon on the left side of the screen.
    b. In the **Configure Oracle REST Endpoint** dialog box, enter the endpoint name, a description of what it does, a relative URI (for example, `/`

`iotmessages`), and choose the action as **POST**. Check the **Configure a Request Payload** checkbox. Click **Next**.

c. In the **Configure the Request Payload** dialog, configure the request information to match the data format that will be sent to the Oracle Integration Cloud Service. If necessary, select **JSON Sample** in the **Request Payload File** pane and provide a JSON sample file from the Oracle IoT Cloud Server.

d. Click **Next**.

e. Click **Done**.

4. Create an invoke.

a. On the right side of the page, again find the ICS Connection you created earlier. This time, drag this connection to the “Drag and Drop an Invoke” icon on the right side of the screen.

b. In the **Configure Oracle REST Endpoint** dialog box, enter the endpoint name, a description of what it does, a relative URI (for example, `/`), and choose the action as **POST**. Check the **Configure a Request Payload** checkbox. Click **Next**.

c. In the **Configure the Request Payload** dialog, configure the request information to match the data format that will be sent from the Oracle Integration Cloud Service. If necessary, select **JSON Sample** in the **Request Payload File** pane and provide a JSON sample file from the Oracle IoT Cloud Server.

d. Click **Next** and note the URI of the REST service in the resulting dialog pane.

e. Click **Done**.

5. Create the mappings and tracking.

a. Click on the **Click Below to Create Map** icon in the middle of the page, then click the **Plus** (+) icon.

b. Map each of the target parameters on the right side to a source parameter on the left side.

c. Click **Save** and **Close**.

d. From the menu in the upper right, select **Tracking**.

e. In the **Source** pane, under “execute/request-wrapper”, drag up to three fields to the right side of the screen.

f. Click **Done**.

g. Click **Save**.

h. Close the integration by selecting **Close** in the upper right.

6. Activate the ICS integration.

a. Slide on the circular slider near the right side of the integration entry to activate the newly created integration. A confirmation window will appear. Select the checkbox to enable tracing if desired, then click **Activate** to continue.

b. Click on the "i" icon near the right portion of the integration listing to open the details screen, and make note the Endpoint URL in the raised panel. In order for an IoT/ICS Integration to send data, we will need a slightly modified version of this endpoint URL to give us the required values for an IoT/ICS integration.

c. To modify the URL, remove `/metadata` and add the REST endpoint created in Step 3. For example, if the endpoint is listed as `https://icsserver/integration/flowapi/rest/IOTCSINTEGRATION1/v01/metadata`, remove `/metadata` and

add the defined ICS Integration REST Endpoint (for example, `/iotmessages`). The final URL should be of the form `https://host:port/integration/flowapi/rest/INTEGRATION_NAME/VERSION_STRING/RESOURCE_URL`.

7. Configure the IOT settings.

   - On the IoT Server UI, choose **Settings**. Under **Trusted CN**, enter the server name of the desired ICS instance, such as `integration.d12.c1dev1.domainname.com`. Without this, you will not receive messages sent to ICS and the IoT log will consist of messages indicating a hostname verification error.

8. Create the IOT integration.

   a. Select **Integrations**, then **Create Integration** and choose **Integration Cloud Service**.

   b. Present a name and description for the integration. For the URL, use the base URL derived earlier without the integration name, version string, and resource URL. For example, `https://icsserver.integration.dc1.c1dev1.domainname.com/integration/flowapi/rest`.

   c. Enter the proper authentication based on the values entered in Step 1 earlier and press **Create**.

   d. Identify the streams that you wish to add within the **Streams** tab of the IoT/ICS Integration. Each stream entry should have a corresponding pre-made ICS Integration that defines a mapping of the stream's message format to the appropriate message format

Add a Custom Certificate for an Integration

If your external application requires a custom certificate to securely connect to the application from Oracle IoT Cloud Service, you can specify the root certificate to use on the **Connection** tab of the Integration.

To specify a custom root certificate for an Integration:

1. Navigate to the Integrations page for your application.

   If you need help with accessing the Integrations page, use the instructions under Integrate Enterprise Applications with Oracle IoT Cloud Service to navigate to the Integrations page for your application.

2. Double-click an integration to edit it.

3. Select the **Connection** tab.

4. Under the **Connection** tab, select **Enable custom certificate for this URL** to upload a root certificate required to connect to your Integration target.

   The **File Upload** dialog appears.

5. Select the Base64 encoded PEM file to be uploaded from your file system.

   The certificate file must be in PEM format (Base64 encoded).

   The uploaded certificate details are displayed on the page.
   After you have uploaded a custom certificate for an integration, you can click **Download Certificate** to download the certificate at any time.

6. Click **Save** to save your changes on the page.

You have now enabled your IoT Cloud Service server to use a custom certificate for connecting to your integration endpoint.

If you deselect the **Enable custom certificate for this URL** option in future, then you would need to upload an appropriate/current certificate again.

Verify Integration Connectivity

Integration connectivity tests help verify the connection between Oracle IoT Cloud Service and the external application or service.

When creating a new integration, the **Verify Connectivity** feature enables you to test the connection between Oracle IoT Cloud Service and the integration target. Depending on the enterprise application or service that you are connecting to, the applicable tests are run. Common tests include the following:

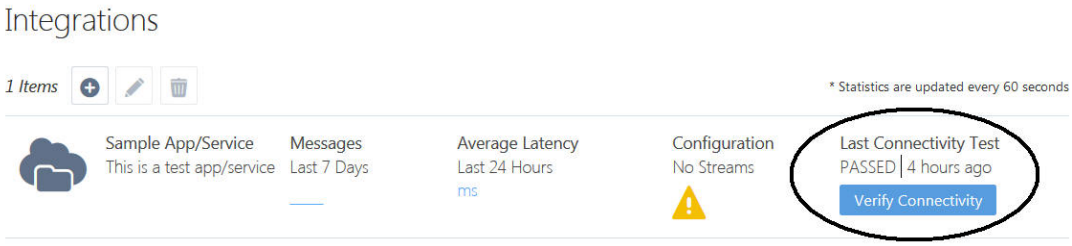
- DNS Resolution Test: Checks whether the server name resolves to an IP address.
- Connectivity Test: Checks whether the IP address is reachable.
- IP/Port test: Checks whether you can talk to the target URL on the specified port.
- SSL Verification Test: Checks whether you have a secured, or trusted, connection to the target. Applies only if the target is an SSL endpoint.
- Authentication Test: Checks whether you can exchange messages with the target.

For more information on the process of creating integrations for specific targets, see the following:

- Integrate Enterprise Applications with Oracle IoT Cloud Service
- Integrate Oracle Analytics Cloud with Oracle Internet of Things Cloud Service
- Integrate Oracle Mobile Cloud Service with Oracle IoT Cloud Service
- Integrate Oracle IoT Cloud Service with JD Edwards EnterpriseOne IoT Orchestrator
- Integrate Oracle Storage Cloud Service with Oracle IoT Cloud Service
- Integrating Oracle Big Data Cloud Service with Oracle Internet of Things Cloud Service

Verifying Connectivity for Your Existing Integrations

You can run the connectivity tests for existing integrations from the Integrations page of your application. The **Verify Connectivity** option appears against each integration target. The result of the last integration test is also displayed.



11

Develop Enterprise Applications That Use Oracle IoT Cloud Service

This section provides information about developing enterprise applications that use Oracle IoT Cloud Service REST APIs.

Topics

- Typical Workflow for Developing Enterprise Applications That Use Oracle IoT Cloud Service
- Receive Incoming Data From Oracle IoT Cloud Service

Typical Workflow for Developing Enterprise Applications That Use Oracle IoT Cloud Service

You can create enterprise applications to query for device data and send commands to your devices that are registered with Oracle IoT Cloud Service.

Use the following table of tasks as a guide when you're developing applications to use with Oracle IoT Cloud Service.

Task	Description	More Information
Register your enterprise application with Oracle IoT Cloud Service.	You must obtain an application ID and shared secret to obtain an access token to use in your enterprise application.	Integrating Enterprise Applications with Oracle IoT Cloud Service
Obtain access token.	Using the application ID and shared secret you received when you registered your enterprise application, use the `POST /iot/api/v1/oauth2/token` REST API call in your enterprise application to obtain the access token to use during your application's session.	REST API Reference for Oracle IoT Cloud Service.
Set up a web listener in your enterprise application.	In order for your enterprise application to be able to receive data stream, it must have a web listener that is accessible from Oracle IoT Cloud Service.	Receive Incoming Data From Oracle IoT Cloud Service
Perform REST API calls within your enterprise application.	Available REST APIs allow you to remotely execute service administration functions, such as querying device data and metadata, or send commands to devices.	REST API for Oracle IoT Cloud Service

Receive Incoming Data From Oracle IoT Cloud Service

After you've successfully registered your enterprise application with Oracle IoT Cloud Service, ensure that it is able to receive the data streams that are being sent from Oracle IoT Cloud Service.

If you haven't registered your enterprise application yet, see Integrating Enterprise Applications with Oracle IoT Cloud Service for information.

Your enterprise application is required to have a Web listener accessible from Oracle IoT Cloud Service in order for your application to receive any data stream being sent by the service.

> **Note:**
>
> If an integration to an Enterprise Application is disconnected from the Oracle IoT Cloud Service instance, messages that were scheduled to be dispatched to that enterprise application will not be delivered. However, they will be persisted if the Enable Message Storage property is selected in the Settings tab of the Management Console. Oracle IoT Cloud Service will withdraw the message dispatch attempt, wait for 10 seconds, and then try to resume normal dispatching again.

The following sections provide some guidelines with code snippets to illustrate what you need to do, and a sample of a simple standalone Java Web application that prints out the data that it receives.

General Steps and Sample Code Snippets

Below is a set of general steps and sample code snippets that you can use as a guideline:

1. Choose a type of Web server where your enterprise application will run as a Web listener. Some Web server examples are Java EE Glassfish, WebLogic, node.js, Apache HTTP Server, or Jetty.
2. Choose the programming model that matches best to the Web server you chose to use in step 1 above. For example, you can use Java using JAX-RS for Java EE Glassfish, JavaScript for node.js, PHP and JavaScript for Apache HTTP Server.
3. Create an HTTP POST call handler according to the programming model you chose to use in step 2 above.

   Below is an example of a Java source code using JAX-RS for Glassfish.

```
 /**
  * POST method for creating an instance of
DataListenerResource
  * @param content representation for the new resource
  * @return an HTTP response with content of the created
resource
  */
 @POST
 @Consumes("application/json")
```

```
@Produces("application/json")
public Response postJson(String content) throws IOException {
   /*
    * PARSE AND PROCESS THE JSON MESSAGE BODY HERE
    */

   /* Ex. JSONArray json = new JSONArray(context)... */

    /* Return HTTP OK */
    String returnStr = "{" +
                " \"type\": \"HttpRequestMessage\",\n" +
                " \"method\": \"POST\",\n" +
                " \"body\": \"OK\"\n" +
                " \n" +
                " }";
        return Response.ok((Object) returnStr).build();
}
```

4. In that HTTP POST call handler in step 3 above, process the JSON body of the data stream that is sent from Oracle IoT Cloud Service as a result of subscribing to receive data stream from the service.

   Below is an example of the JSON body of a message:

```
{    "id": "fbba219d-f30a-4faf-bfd9-a8777961f861",
     "clientId": "402d2487-9c25-435b-8f6f-464b7e1c512e",
     "source": "0-FM",
     "destination": "",
     "priority": "LOW",
     "reliability": "BEST_EFFORT",
     "eventTime": 1444438809722,
     "eventTimeAsString": "2015-10-10T01:00:09Z",
     "sender": "",
     "type": "DATA",
     "properties": {},
     "direction": "FROM_DEVICE",
     "receivedTime": 1444438809827,
     "receivedTimeAsString": "2015-10-10T01:00:09Z",
     "payload": {
         "format": "urn:oracle-ebsdemo:iot:device:data:3dprintersensors",
         "data": {
             "temperature": 25.21332000000001,
             "humidity": 45.0912,
             "vibrations": 255.60516426707815,
             "led": true
         }
     }
}
```

5. Return the correct HTTP Response according to how the processing worked in step 4 above.

Below is an example code snippet:

```
/* Return HTTP 200 OK */
    String returnStr = "{" +
            " \"type\": \"HttpRequestMessage\",\n" +
            " \"method\": \"POST\",\n" +
            " \"body\": \"OK\"\n" +
            " \n" +
            " }";

     return Response.ok((Object) returnStr).build();
```

Sample of a Standalone Java Web Application

Below is an example of a simple standalone Java Web application that prints the data it receives. In this example, you need to use `http://YOUR_MACHINES_IP:9000/httpConnector` that is used in the sample code below, as the value to enter in the **URL** field when setting up the enterprise application integration via the Oracle IoT Cloud Service Management Console. Where this sample application simply prints the body of the received data stream, your application will most likely make use of a JSON parser library to process the data.

```
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetSocketAddress;

public class IoTDataReceiver {

    public static void main(String[] args) throws IOException {
         System.out.println("Listening on http://localhost:9000/
httpConnector");
         HttpServer server = HttpServer.create(new
InetSocketAddress(9000), 0);
         server.createContext("/httpConnector",
IoTDataReceiver::handle);
         server.setExecutor(null); // creates a default executor
         server.start();
     }

      public static void handle(HttpExchange t) {
         System.out.println("HvacDataReceiver.handle(" + "t = [" + t +
"]" + ")");

          try {

System.out.println("================================================
=");
             BufferedReader bodyReader = new BufferedReader(new
InputStreamReader(t.getRequestBody()));
             String line;
```

```
                while((line = bodyReader.readLine()) != null) {
                    System.out.println(line);
                }
                // Send 200
                t.sendResponseHeaders(200, 0);
                t.close();
            } catch (Throwable e) {
                e.printStackTrace();
            }
        }
    }
```

About the Oracle Internet of Things Cloud Service REST API

REST APIs give your applications access to the different capabilities of Oracle Internet of Things Cloud Service.

REST APIs are available for the application, device model, device resource, and messages components of the Oracle Internet of Things Cloud Service. Requests to these REST APIs are protected through OAuth 2.0 protocol.

You are able to manage your Oracle Internet of Things Cloud Service devices and applications using REST APIs. You can enroll and search for your gateway devices, gateway applications, and enterprise applications. You can also group your devices, specify metadata, or assign software artifacts using REST APIs.

The Oracle Internet of Things Cloud Service messaging proxy have REST APIs that are used to send and receive messages between devices and applications. The gateway devices running the Oracle Internet of Things Cloud Service gateway and non-Java devices can use REST APIs to send and receive message from enterprise resources. Enterprise applications can use REST APIs to communicate with devices and distributed applications running onOracle Internet of Things Cloud Service gateway devices, even if those devices are only occasionally connected.

The Oracle Internet of Things Cloud Service repository also exposes a set of REST APIs that allow interaction with devices and enterprise applications. REST APIs can also be used to automate bulk registration requests. Message connectors and enterprise applications can be activated using Repository REST APIs. Software can be deployed to the artifact repository, then updated, deleted, and viewed using REST APIs.

There are also REST APIs that allow you to perform server configuration tasks. Access to the Oracle Internet of Things Cloud Service database can only been done through REST APIs.

See the REST API Reference for Oracle IoT Cloud Service documentation in the Oracle Help Center.

12

Troubleshoot

This topic contains information about known issues and any known workarounds.

Topics

- Troubleshoot Integrations
- Troubleshoot Management Console Issues
- Troubleshoot REST API Issues
- Troubleshoot Stream Explorer Issues

Troubleshoot Integrations

Use the information in this topic to resolve known issues that you may encounter while working with integrations.

- **Issue:** New messages do not seem to appear under the Messages tab for the Integration.

  Description: Messages are listed under the **Messages** tab of the Integration page for an IoT Cloud Service Integration. The messages appear in descending order by `sentTime`. This means that the latest messages appear near the top. However, the messages that are not dispatched successfully have a `sentTime` value of 0. This results in these failed messages appearing at the bottom of the list.

  Workaround: If you have a lot of dispatched messages in the list, you need to scroll down to the earliest messages in order to see the non-dispatched messages.

  You can also use a REST command to obtain the list of non-dispatched messages. An example command follows. Notice that the `since` and `until` parameters are set to `0` and `1` respectively for non-dispatched messages. You can additionally adjust the values of `offset` and `limit`.

  ```
  GET https://iotserver/iot/api/v2/apps/AAAAAARUNC8A-5EBQ/integrations/
  AAAAAARUNC8A-CMEA/messages?limit=100&offset=0&type=DATA&since=0&until=1
  ```

  See *REST API Reference for Oracle Internet of Things Cloud Service* for more details on the REST command.

  See Verifying Integration Connectivity for verifying connectivity to the integration target.

- **Issue:** Messages show up only sporadically or randomly.

  Description: If messages show up sporadically, or at random intervals, you might be experiencing intermittent dispatch failures for one or more of your integrations. When an IoT integration is unable to dispatch a message, the IoT server waits 10 seconds before attempting to dispatch to the integration URL again. If the dispatch fails again, the IoT server waits 20 seconds before the next dispatch. For each additional failure, the IoT server adds an additional 10 seconds to the wait-time. This behavior ensures that the IoT

server doesn't spend too many cycles on repeated failures. The wait-time can increment up to a maximum value of one hour.

A message dispatch can fail for a variety of reasons. For example:

- Network connectivity issue. See Verifying Integration Connectivity for verifying connectivity to the integration target.
- Incorrect URL.
- Application at the other end of the Integration URL returns an error code.
- Application at the other end of the Integration URL takes too long to return a response.

Workaround:

1. Verify that the application at the other end of the integration URL accepts messages successfully.
   a. Use a `curl` command to send a message directly to the application and verify a successful response.
   b. Make sure that the message response from the application does not take too long. A value of 20 seconds, for example, would be too long.
   c. Make adjustments, as necessary, to the external system to ensure that a successful response is received within the time that you would expect.
2. Send messages to the integration, and the messages should be received correctly.
3. Repeat the previous steps if the messages show up sporadically or randomly.

Troubleshoot Management Console Issues

Use the information in this topic to resolve known issues that you may have encountered while working with the Oracle IoT Cloud Service Management Console.

- **Issue:** A status code of `502 Proxy Error` is received while a large batch registration is being processed.

  Description: If you are registering a large number of devices using the **Batch Registration** feature under the **Devices** menu, you may receive a status code of `502 Proxy Error` while you are waiting for the batch registration to complete. If the progress indicator is still spinning when this error is displayed, it means your batch registration information has already been received and the request is still being processed by the Oracle IoT Cloud Service.

  Workaround: Continue to wait until the progress indicator has stopped spinning to get the actual results of your batch registration request.

- **Issue:** If you are using a Firefox browser while scrolling the tables in the **Messages** or **Alerts** tabs of the **Data** page, you may not actually see all of the alerts and messages.

  Description: If you are using a Firefox browser to view the tables in the **Alerts** or **Messages** tabs in the **Data** page and you are scrolling using the mouse wheel, scroll up button, or scroll down button, the **Alerts** or **Messages** table will not display all of the actual data.

  Workaround: Grab the scroll bar for **Alerts** or **Messages** table and use it to scroll down or up to view all of data. You can also try using a different web browser.

- **Issue:** The filter for Logging Level is not working correctly.

  Description: If you are viewing a gateway device's log messages using the **View/Edit** page for **Management** tab under **Devices**, and you set the **Logging Level** in **Logs** tab to `Warning`, only the Warning and Severe messages should be displayed. Instead, `Fine` level messages are also displayed.

  Workaround: None known.

- **Issue:** Exporting an IoT Application does not work when using a Safari browser.

  Description: The feature to export an IoT Application fails if you're using a Safari browser on a Mac OS X platform. This issue is due to Safari bug https://bugs.webkit.org/show_bug.cgi?id=102914.

  Workaround: Use another browser, such as Chrome, when using the Export IoT Application feature.

- **Issue:** An uploaded software that's been installed on a device appears to have been deleted from that device after the uploaded software is deleted from Oracle IoT Cloud Service.

  Description: Using the **Software** page in the **Devices** tab of the Oracle IoT Cloud Service Management Console, if you upload a device software to the Oracle IoT Cloud Service software repository, install it on a gateway device, and then delete the uploaded software, the device software appears to have been deleted from the gateway device. In fact, the device software is still installed in the gateway device, but the **Software** tab in the gateway device's details page no longer displays the information about the installed software.

  Workaround: None.

- **Issue:** After you upgrade to Oracle IoT Cloud Service version 16.1.5 from an older version, any previous alert message counts are lost.

  Description: When you upgrade from a version of Oracle IoT Cloud Service prior to version 16.1.5, the history of the total number of alert messages displayed is lost and the number displayed in the **Alerts and Messages** page is zero. To view this number, click **Devices** at the top right of the Management Console UI and click **Alerts and Messages** on the left navigation area. In the **Alerts** tab, any alert messages received prior to the upgrade are no longer accounted for in the total number displayed in this **Alerts** tab.

  Workaround: None.

- **Issue:** If you're using a non-English locale, the time format used in the date selector fields of the **Alerts and Messages** page resets to the English format when the page is refreshed.

  Description: Choose a different language to use for your web browser and log in to your Oracle IoT Cloud Service instance. Click **Devices** from the **Home** page and then **Alerts and Messages** on the left navigation area. View the **Alerts** section. Notice the date format in the date fields at the top of the page. Refresh the current page using F5 and you will see that the date is changed to use the English format. The same occurs in the Messages section.

  Workaround: Alternately select the **Alerts and Messages** tab and the format displays back to the correct locale that your web browser is set to use.

- **Issue:** When using a Firefox ESR browser version 38.x, the Unresponsive script error dialog may appear when viewing the detailed information about the devices.

Description: If you're using Firefox ESR browser version 38.x, click **Devices** and then select **Management** on the left navigation area. The Unresponsibe script error dialog will appear when you have more than 30 devices to display.

Workaround: Upgrade to Firefox version 43 or later, or use another web browser, such as Chrome or Internet Explorer.

- **Issue:** The graphic diagram that's supposed to appear on the left-hand side of the **Overview - All Devices** section is blank when you use a Firefox ESR browser version 38.x.

  Description: Click **Devices** and then **Management** on the left navigation area. Expand the **Overview - All Devices** section. A diagram showing the states of the devices is supposed to appear on the left side. If you're using Firefox ESR browser version 38.x, the diagram is not displayed.

  Workaround: Upgrade to Firefox version 43 or later. Or use another browser, such as Chrome or Internet Explorer.

Troubleshoot REST API Issues

Use the information in this topic to resolve Oracle Internet of Things Cloud Service REST API known issues.

- **Issue:**Oracle Internet of Things Cloud Service 16.1.5 does not return an appropriate message when you try to access any REST APIs that are no longer supported.

  Description: Several REST APIs that were available prior to the Oracle Internet of Things Cloud Service 16.1.5 release are no longer supported in the Oracle Internet of Things Cloud Service 16.1.5 version. When you try to access any of these unsupported REST APIs, Oracle Internet of Things Cloud Service returns a `404 (Not Found)` status, but does not give information on what new APIs, if any, should be used.

  Workaround: Use the following table to get information about the REST APIs that are no longer supported in the Oracle Internet of Things Cloud Service 16.1.5 version and what the REST APIs, if any, that replaced them.

REST APIs that are no longer supported in version 16.1.5	Replaced by the following REST APIs	Notes
`/iot/api/v1/endpoints/{connector-endpoint-id}`	`/iot/api/v2/apps/{app-id}/integrations`	See the latest REST API documentation.
`/iot/api/v1/endpoints/{configurations}`	None.	Completely removed in version 16.1.5 of Oracle Internet of Things Cloud Service.

REST APIs that are no longer supported in version 16.1.5	Replaced by the following REST APIs	Notes
`/iot/api/v1/annotators`	None	Completely removed in Oracle Internet of Things Cloud Service 16.1.5 version. When you're configuring an integration in Oracle Internet of Things Cloud Service 16.1.5, the annotation references are explicitly defined.
`/iot/api/v1/formats`	`/iot/api/v2/apps/{app-id}/formats`	Beginning with Oracle Internet of Things Cloud Service 16.1.5, you define message formats within the context of a device model.

- **Issue:** A **403 Forbidden** error is returned when a web application running on a particular host tries to call an Oracle Internet of Things Cloud Service REST API.

  Description: If a web application residing on a host tries to access an Oracle Internet of Things Cloud Service resource, then access is denied by default for a host that does not have a `.oraclecloud.com` or `.oraclecloudapps.com` domain suffix. You might notice the following error in the log file:

  ```
  <Error> <oracle.IoT.Security> <BEA-000000> <CORS : Invalid CORS request; Origin= ...
  ```

  This means that the host name has not been explicitly added to the CORS (Cross-Origin Resource Sharing) list. CORS specifies if the system that is hosting the web application is allowed to access Oracle Internet of Things Cloud Service resources.

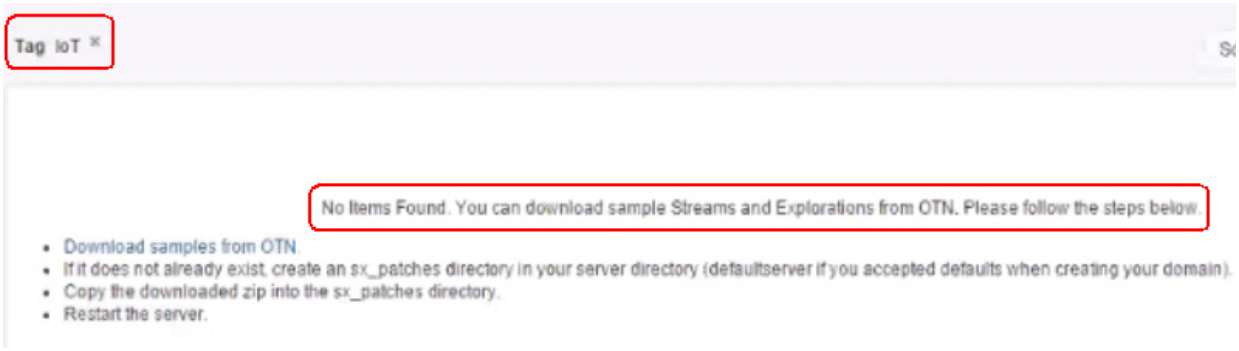
  Workaround: You need to add the host name of the system hosting the web application to the CORS list. You can do this in the **Allowed Hosts for Cross-Origin Resource Sharing** field in the **Security** section of the **Settings** page of Oracle Internet of Things Cloud Service.

Troubleshoot Stream Explorer Issues

Use the information in this topic to resolve known issues that you may have encountered while working with the Oracle IoT Cloud Service Stream Explorer.

- **Issue**: The initial data streams created using the Management Console may not display when you try to view it using the Oracle Stream Explorer user interface.

  Description: After you create the initial data streams using the Data Analytics page in the Oracle IoT Cloud Service Management Console and you click **View** , the Oracle Stream Explorer UI may display a screen stating that no items were found, as illustrated below:



Workaround: Dismiss **Tag IoT** at the top left corner of the pane and the list of streams should be displayed.

13
Glossary

This is a list of common terms used in the Oracle Internet of Things Cloud Service documentation and user interface.

Term / Concept / Standard Name	Description
Activation	The process when an endpoint of transitions from Registered to Active status. The point at which the ID and shared secret assigned during registration are used to create the secure association between endpoint and Oracle IoT Cloud Service.
Activity	Refers to the overall data exchange activity performed by the Oracle IoT Cloud Service, including data and alerts exchanged with devices, events triggered by the event processing component, data and actions exchanged with enterprise applications, and events related to device and software management.
Alert Message	An Oracle IoT Cloud Service message type that is used to indicate an alert condition has been detected by the device and requires further analysis by the Oracle IoT Cloud Service, or an action by a user or operator.
Annotation	Specifies a set of key-value pairs of metadata that should be included in the data stream output generated by connectors to enhance messages with additional context.
Artifact Repository	The repository within the Oracle IoT Cloud Service where Device Software is stored prior to being installed on devices running the Oracle IoT Cloud Service Gateway.
Client Software	Software provided to connect edge devices or external applications with Oracle IoT Cloud Service. Client Software refers to either the Device Libraries, the Enterprise Libraries, or the Gateway that are delivered with Oracle IoT Cloud Service Client Software package.
Client Software Library	A software library used to develop device or enterprise applications that communicate with Oracle IoT Cloud Service. Client Software Libraries are provided to support a variety of languages and platforms, including Java, C, JavaScript, and Android.
Gateway	An OSGi-based software framework provided by Oracle, designed to function as a gateway for devices that are unable to directly connect with Oracle IoT Cloud Service.

Term / Concept / Standard Name	Description
Connector	An internal construct used in Oracle IoT Cloud Service to associate data streams with external consumers.
Data Message	An Oracle IoT Cloud Service Message type used to transmit data from a device to the cloud service. Data Messages contain key-value pairs representing the attributes and values reported by a device.
Data Stream	A continuous sequence of messages delivering information from an endpoint to Oracle IoT Cloud Service. Data Streams typically originate on devices and are delivered via an Oracle IoT Cloud Service Client Software Library, the Oracle IoT Cloud Service Gateway, or through the Oracle IoT Cloud Service REST APIs.
Deactivated Device	A device that is no longer an active participant in the Oracle IoT Cloud Service network. Its previously assigned authentication credentials have been revoked. The device is still trusted and can request to be activated again.
Decommission	A permanent condition that occurs when a device is taken out of commission.
Device	Hardware component with communication capabilities and is able to connect to Oracle IoT Cloud Service either directly or via a Gateway Device. .
Device Adapter	A piece of code that runs on the Oracle IoT Cloud Service Gateway to provide protocol support for connecting Indirectly Connected Devices.
Device Application	A software component that is running on an edge device and uses an Oracle IoT Cloud Service Client Software Library to connect to Oracle IoT Cloud Service, either directly or through a third party Gateway Device.
Device ID	An identifier that uniquely identifies a device.
Device Library	An Oracle IoT Cloud Service Client Software Library used to develop Device Applications that communicate with Oracle IoT Cloud Service. Device Libraries are typically used to expose device functionality to Oracle IoT Cloud Service.
Device Metadata	A set of attribute/value pairs that describe an entry in the device repository.
Device Model	Describes the device metadata, message formats, and resources associated with a specific type of device. A device may support multiple device models.
Device Model Repository	The set of Device Models that have been loaded into and are known by an instance of Oracle IoT Cloud Service

Term / Concept / Standard Name	Description
Device Selection Group	Logical grouping of devices based on common parameters, such as device model, location, and manufacturer, that are associated with an IoT Application
Device Software	Applications, device adapters, and other software packaged as OSGi bundles that run on the Oracle IoT Cloud Service Gateway, and whose life cycle can be managed through the Oracle IoT Cloud Service Management Console.
Directly Connected Device (DCD)	A device that is capable of communicating directly with Oracle IoT Cloud Service by running an application that uses an Oracle IoT Cloud Service Client Software Library or by calling the Oracle IoT Cloud Service REST APIs.. This device type cannot register any indirectly connected devices. The only supported direct connection protocol is HTTPS over TCP/IP.
Endpoint	An entity that is addressable in the Oracle IoT Cloud Service system. It has a unique ID and can be communicated with using a controlled security policy.
Endpoint ID	The globally unique identifier used to reference a specific Endpoint.
Endpoint Lifecycle	The set of state changes that occur for an endpoint in Oracle IoT Cloud Service. Supported lifecycle states are Registered, Activated, Deactivated, and Decommissioned.
Endpoint Repository	A database that contains information about all the endpoints known to Oracle IoT Cloud Service. The database is internal to Oracle IoT Cloud Service and accessible via Oracle IoT Cloud Service REST APIs requests.
Endpoint State	The operational state of an endpoint in Oracle IoT Cloud Service. The endpoint state determines whether an endpoint can actively participate in Oracle IoT Cloud Service.
Enterprise Library	A Client Software Library used to develop Enterprise Applications that communicate with IoT Cloud Service. Enterprise Libraries are typically used to inspect and control a Device.
Enterprise Application	A software component created for a specific business problem and runs externally to Oracle IoT Cloud Service. It integrates and communicates with Oracle IoT Cloud Service using one of the Oracle IoT Cloud Service connectors or by invoking Oracle IoT Cloud Service REST API.
Enterprise Application Integration	The configuration required to connect streams of incoming data with an Enterprise Application.
Gateway Device	A device that communicates with Oracle IoT Cloud Service on behalf of other devices that are incapable of direct communication with Oracle IoT Cloud Service.

Term / Concept / Standard Name	Description
IoT Application	A set of Oracle IoT Cloud Service definitions that comprise an end-to-end IoT solution. These definitions include Device Models, Explorations, and Integrations.
Indirectly Connected Device (ICD)	A device that communicates with Oracle IoT Cloud Service through a Gateway Device. These devices communicate with the Gateway Device over a non-HTTPS TCP/IP protocol or interface, such as Bluetooth, Zigbee, I2C, or GPIO. For this device to indirectly communicate with Oracle IoT Cloud Service, it is required to be connected to a Gateway Device that has already been activated with Oracle IoT Cloud Service.
Integration	The association of an incoming data stream with a data sink. The data sink may be an integrated analytics tool, such as Oracle Stream Explorer, or an external enterprise application, or PaaS. Oracle BI Cloud Service is an example of a pre-defined PaaS integration. Integrations may also contain Annotations.
Log Message	An Oracle IoT Cloud Service Message type used to transmit application or device logging information to the Oracle IoT Cloud Service.
Managed Device	A device configured with the software that supports secure bi-directional messaging, and remote provisioning and lifecycle management of applications from Oracle IoT Cloud Service. This device is configured with the installation of the Oracle IoT Cloud Service Gateway.
Management Console	The web-based user interface (UI) component of Oracle IoT Cloud Service. It is used for performing management and configuration tasks.
Message	A single unit of data stream at a given point in time. It may be any one of the valid Message Types. The payload of a message is dependent on the message type.
Message Format	Specifies a set of key-value pairs of data that are included in an Oracle IoT Cloud Service Message payload. Each Message Format must have a unique identifier using a Uniform Resource Name (URN) value.
Message Type	The specific class of an Oracle IoT Cloud Service Message. The type of the message may be: Data, Alert or Log. The message type dictates the content of the message payload field.
Offline Device	A device that has not communicated with the Oracle IoT Cloud Service for over 60 minutes.
Online Device	A device that has successfully communicated with Oracle IoT Cloud Service within the last 60 minutes.

Term / Concept / Standard Name	Description
Registered Device	A Device that has been provisioned within the Oracle IoT Cloud Service. It has a unique identifier and shared secret that can be used to complete the Activation process.
Resource	A logical or physical entity managed by a device and registered within Oracle IoT Cloud Service. Resource values can be accessed by applications through communication with Oracle IoT Cloud Service.
Service	A general term used to refer to the Oracle IoT Cloud Service.
Service Administrator	A person who configures and controls overall access to Oracle IoT Cloud Service, including the configuration and integration with other Oracle Cloud Services. This is person is the primary user of the Oracle IoT Cloud Service Management Console.
Service Operator	An individual who is responsible for the day-to-day Oracle IoT Cloud Services operations, such as devices' lifecycle management and monitoring of messages.
Service User	A person who views data and reports generated by Oracle IoT Cloud Service. This person uses either the Management Console or external applications integrated with Oracle IoT Cloud Service.
Shared Secret	A symmetric key that is required to be available on both the device and Oracle IoT Cloud Service for a successful device activation. The shared secret can be created automatically by Oracle IoT Cloud Service during the registration process or provided by the device or the service administrator.
Software Repository	Built-in storage and retrieval mechanism for software packages intended to be deployed as device applications on Managed Devices.
Storage	Persistence of message to the internal data store in Oracle IoT Cloud Service.
Unmanaged Device	A device configured with software that only supports a secure bi-directional messaging with Oracle IoT Cloud Service and does not support software installation or application lifecycle management.