Salesforce IoT REST API Getting Started Guide

Version 42.0, Spring '18

@salesforcedocs

Last updated: March 9, 2018

CONTENTS

CHAPTER 1 Introducing Salesforce IoT REST API

Use the Salesforce IoT REST API to create and manage your orchestrations and contexts and retrieve usage data. An orchestration is the business logic that processes incoming events by using states. A context specifies the data that the orchestration can access: one or more platform events, and optionally, data from an associated Salesforce object. With the Salesforce IoT REST API, you can create, retrieve, update, and delete your orchestrations and contexts. You can retrieve an orchestration instance, delete an instance, and retrieve traffic data. Other tasks that you can perform with an orchestration include activation and deactivation. Monitor your event and instance usage by retrieving usage data.

The resources that the REST API provides are based on the following orchestration-related components.

- Contexts
- Orchestrations
- Activations
- Instances
- Traffic
- IoT usage data

To learn about the Salesforce IoT REST API properties, see the following sections.

Salesforce IoT REST API Architecture
To use the Salesforce IoT REST API, understand a few characteristics of its architecture.

Specify Response Sizes
Use filter parameters to return only the data that the application needs. To specify the response size by group (big, medium, or small), use filterGroup as a request parameter or in a request body. To exclude or include specific properties, use either exclude or include as a request parameter or in a request body.

SEE ALSO:

Salesforce Help: Salesforce IoT Explorer Edition

Salesforce IoT REST API Architecture

To use the Salesforce IoT REST API, understand a few characteristics of its architecture.

Authentication
Salesforce IoT REST API uses OAuth 2.0.

Use CORS to Access Salesforce Resources from Web Browsers

Cross-Origin Resource Sharing (CORS) enables web browsers to request resources from origins other than their own (cross-origin). For example, using CORS, JavaScript code at `https://www.example.com` could request a resource from `https://www.salesforce.com`. To access supported Salesforce APIs, Apex REST resources, and Lightning Out from JavaScript code in a web browser, add the origin serving the code to a Salesforce CORS whitelist.

Default HTML Entity Encoding

Salesforce IoT REST API responses are minimally HTML entity-encoded by default. For example, characters such as the > and < operators are escaped as `&gt;` and `&lt;` string values.

JSON Support

The JavaScript Object Notation (JSON) format is supported with UTF-8 in requests and responses. Date-time information is in ISO8601 format.

Localized Names and Times

Salesforce IoT REST API localizes both the names and datetimes to the locale setting of the user. Users set their locale in their personal information. If the user hasn't set a locale, IoT REST API uses the default locale for the org.

Salesforce ID Length

Salesforce IDs in response bodies are always 18-character IDs. In request bodies, you can use either 15-character or 18-character IDs.

API Version

Salesforce IoT REST API is supported by API version 42.0 and later.

OpenAPI Specification

Salesforce IoT REST API is described using the OpenAPI specification, which is a specification for describing, producing, consuming, and visualizing RESTful Web services. The OpenAPI specification file enables you to learn and interact with API elements, including all available endpoints and input and output representations.

Authentication

Salesforce IoT REST API uses OAuth 2.0.

You must access Salesforce securely using HTTPS.

Use CORS to Access Salesforce Resources from Web Browsers

EDITIONS

Available in: Salesforce Classic and Lightning Experience

Available in: **Developer**, **Enterprise**, **Performance**, and **Unlimited**

USER PERMISSIONS

To create, read, update, and delete:

- Modify All Data

Cross-Origin Resource Sharing (CORS) enables web browsers to request resources from origins other than their own (cross-origin). For example, using CORS, JavaScript code at `https://www.example.com` could request a resource from `https://www.salesforce.com`. To access supported Salesforce APIs, Apex REST resources, and Lightning Out from JavaScript code in a web browser, add the origin serving the code to a Salesforce CORS whitelist.

These Salesforce technologies support CORS.

- Analytics REST API
- Bulk API
- Chatter REST API
- Salesforce IoT REST API
- Lightning Out
- REST API

- User Interface API
- Apex REST

In Salesforce, add the origin serving the code to a CORS whitelist. If a browser that supports CORS makes a request to an origin in the whitelist, Salesforce returns the origin in the `Access-Control-Allow-Origin` HTTP header, along with any additional CORS HTTP headers. If the origin is not included in the whitelist, Salesforce returns HTTP status code 403.

1. From Setup, enter *CORS* in the `Quick Find` box, then select **CORS**.
2. Select **New**.
3. Enter an origin URL pattern.

   The origin URL pattern must include the HTTPS protocol (unless you're using your localhost) and a domain name and can include a port. The wildcard character (*) is supported and must be in front of a second-level domain name. For example, `https://*.example.com` adds all subdomains of `example.com` to the whitelist.

   The origin URL pattern can be an IP address. However, an IP address and a domain that resolve to the same address are not the same origin, and you must add them to the CORS whitelist as separate entries.

Important: CORS does not support requests for unauthenticated resources, including OAuth endpoints. You must pass an OAuth token with requests that require it.

Default HTML Entity Encoding

Salesforce IoT REST API responses are minimally HTML entity-encoded by default. For example, characters such as the > and < operators are escaped as `&gt;` and `&lt;` string values.

If an orchestration rule contains this condition:

```
MyEvent.field1 > 10
```

When the orchestration is retrieved with the API, the value of the retrieved condition is:

```
MyEvent.field1 &gt; 10
```

To prevent HTML encoding, set the `X-Chatter-Entity-Encoding` HTTP header in a request to `false`.

For more information, see Response Body Encoding in the *Chatter REST API Developer Guide*.

JSON Support

The JavaScript Object Notation (JSON) format is supported with UTF-8 in requests and responses. Date-time information is in ISO8601 format.

When performing write requests (POST, PUT, PATCH), the REST API client must send the data in JSON format and include the `Content-Type: application/json` header.

Note: The XML format is not supported.

Localized Names and Times

Salesforce IoT REST API localizes both the names and datetimes to the locale setting of the user. Users set their locale in their personal information. If the user hasn't set a locale, IoT REST API uses the default locale for the org.

Clients can use any locale available in the org to override the default locale. To override the locale for an API request, use the Accept-Language HTTP header.

Note: ISO-8601 dates are always in GMT.

Salesforce ID Length

Salesforce IDs in response bodies are always 18-character IDs. In request bodies, you can use either 15-character or 18-character IDs.

API Version

Salesforce IoT REST API is supported by API version 42.0 and later.

The API version is required in the request URI. The request URI format is:

```
https://Salesforce_Instance/services/data/vXX.X/iot/Resource
```

For example, this is a URI with a specific version number.

```
https://Salesforce_Instance/services/data/v42.0/iot/Resource
```

OpenAPI Specification

Salesforce IoT REST API is described using the OpenAPI specification, which is a specification for describing, producing, consuming, and visualizing RESTful Web services. The OpenAPI specification file enables you to learn and interact with API elements, including all available endpoints and input and output representations.

Using the Swagger UI tool, which consumes the OpenAPI specification, you can visualize the API and its documentation, and try out API calls. Or use the Swagger Editor to visualize the API and its documentation.

To access the OpenAPI file for Salesforce IoT REST API, see iot-connect-api-swagger.yaml

SEE ALSO:

- Swagger and OpenAPI Documentation
- Swagger UI
- Swagger Editor

Specify Response Sizes

Use filter parameters to return only the data that the application needs. To specify the response size by group (big, medium, or small), use filterGroup as a request parameter or in a request body. To exclude or include specific properties, use either exclude or include as a request parameter or in a request body.

filterGroup
: Use the `filterGroup` parameter in a request URL or in a request body to specify whether to return a small, medium, or big group of properties.

exclude
: To exclude specific properties from a response body, use the exclude parameter in the request URI. Precede each field with a forward slash (`/`).

include
To limit the response body to specific properties, use the include parameter in the request URI. Precede each field with a forward slash (/).

filterGroup

Use the `filterGroup` parameter in a request URL or in a request body to specify whether to return a small, medium, or big group of properties.

Each property in a response body is assigned a group size:

- `Big`—The default. Includes every property in a response body, including those marked Big, Medium, and Small.
- `Medium`—Includes properties marked Medium and Small.
- `Small`—Includes only properties marked Small.

Only the `totalSize` property, which is the number of records returned in a retrieved collection, uses the `filterGroup` size of `Medium`. Other properties are not filtered. For example, if you perform the following request:

```
GET /iot/orchestrations?filterGroup=Small
```

The returned list of orchestrations doesn't include the `totalSize` property, but it includes all other fields.

exclude

To exclude specific properties from a response body, use the exclude parameter in the request URI. Precede each field with a forward slash (/).

For example:

```
request_uri?exclude=/field1
```

To exclude multiple fields, concatenate the fields with a bar (|), which corresponds to `%7C`.

```
request_uri?exclude=/field1%7C/field2
```

For example, to exclude the orchestration definition in the response, use the following URI.

```
GET /iot/orchestrations/orchestration_id?exclude=/definition
```

include

To limit the response body to specific properties, use the include parameter in the request URI. Precede each field with a forward slash (/).

For example:

```
request_uri?include=/field1
```

To include multiple fields, concatenate the fields with a bar (|), which corresponds to `%7C`.

```
request_uri?include=/field1%7C/field2
```

For example, to include only the orchestration status and name fields in the response, use the following URI.

```
GET /iot/orchestrations/orchestration_id?include=/status%7C/name
```

CHAPTER 2 Quick Start: Create an Orchestration to Monitor Solar Panels

This quick start guides you through the steps of creating a context and an orchestration. The orchestration contains rules that process solar panel events. If a panel has low power output, a rule creates a case for servicing the panel. The quick start also includes instructions for setting up a connected app to authorize the API calls to your Salesforce org and steps for authentication.

Prerequisites

Create a Salesforce Developer Edition

Set up a Developer Edition org for testing your code.

Verify Required Permissions

To access Salesforce IoT resources using the Salesforce IoT REST API, you must have the API Enabled permission, and Salesforce IoT must be enabled.

Define the Solar Panel Platform Event

The orchestration in this quick start consumes solar panel platform events. The context associates the solar panel platform event to the orchestration. Define the platform event named Solar Panel Event with two custom fields.

Making API Calls with the curl Tool

The steps in the quick start use the curl tool to send HTTP requests to access, create, and manipulate IoT REST resources.

Create a Salesforce Developer Edition

Set up a Developer Edition org for testing your code.

If you aren't already a member of the Salesforce developer community, go to developer.salesforce.com/signup and sign up for a Developer Edition org. Even if you already have Enterprise Edition, Unlimited Edition, or Performance Edition, use Developer Edition for developing, staging, and testing your solutions against sample data to protect your org's live data, especially for applications that insert, update, or delete data (as opposed to just reading data).

Verify Required Permissions

To access Salesforce IoT resources using the Salesforce IoT REST API, you must have the API Enabled permission, and Salesforce IoT must be enabled.

API Enabled Permission

The API Enabled permission is enabled by default in a Developer Edition org.

Salesforce IoT Enabled

To enable Salesforce IoT, see Enable Salesforce IoT Explorer Edition in *Salesforce Help*.

Define the Solar Panel Platform Event

The orchestration in this quick start consumes solar panel platform events. The context associates the solar panel platform event to the orchestration. Define the platform event named Solar Panel Event with two custom fields.

1. From Setup, enter `Platform Events` in the Quick Find box, then select **Platform Events**.
2. On the Platform Events page, click **New Platform Event**.
3. For Label, enter `Solar Panel Event`.
4. For Plural Label, enter `Solar Panel Events`.
5. For Description, enter `An event that solar panels send.`
6. Click **Save**.
7. In the Custom Fields & Relationships related list, click **New**.
8. Select Text, and click **Next**.
9. For Field Label, enter `Serial Number`.
10. For Length, enter `20`, and select **Required**.
11. Keep the defaults for the other fields, and click **Save**.
12. Follow steps 7 and 8 to add a field of type Number, and for Field Label, enter `Power Output`.
13. Keep the defaults for the other fields, and click **Save**.

Making API Calls with the curl Tool

The steps in the quick start use the curl tool to send HTTP requests to access, create, and manipulate IoT REST resources.

curl is pre-installed on many Linux and macOS systems. Windows users can download a version at curl.haxx.se. When using HTTPS on Windows, ensure that your system meets the curl requirements for SSL.

Note: curl is an open-source tool and is not supported by Salesforce.

If you prefer a tool with a user interface, you can use a REST API client of your choice, for example, Postman. Follow the steps for authentication first. Then, for each step, provide the URI, the HTTP method, and if needed, request parameters and body.

Authentication

Create a Connected App
Create a connected app in Salesforce, and enable OAuth. The client application uses the connected app to connect to Salesforce.

Create a Connected App

Create a connected app in Salesforce, and enable OAuth. The client application uses the connected app to connect to Salesforce.

1. Use your Developer Edition org to create a new connected app.
   a. In Lightning Experience, use the App Manager to create connected apps. From Setup, enter `App` in the Quick Find box, then select **App Manager**. Click **New Connected App**.
   b. In Salesforce Classic, from Setup, enter `Apps` in the Quick Find box, then select **Apps** (under **Build** > **Create**). Under Connected Apps, click **New**.
2. For the connected app name, enter `IoT API Quick Start`.
3. For Contact Email, enter your email address.
4. Select **Enable OAuth Settings**.
5. For Callback URL, enter `https://`. Depending on which OAuth flow you use, this is typically the URL that a user's browser is redirected to after successful authentication. For this quick start, we don't have a URL to redirect you to because we are using curl to perform the API calls.
6. For OAuth scope, select **Access and manage your data (api)**. Then click **Add**.
7. Click **Save**.

The consumer key is displayed, and a consumer secret is displayed when you click the link to reveal it.

Get an Access Token

To get an access token, make a POST request to the authorization endpoint.

Authorization Endpoint

```
https://login.salesforce.com/services/oauth2/token
```

POST Request Parameters

The POST request must supply these parameters.

Name	Type	Description
grant_type	string	Sets the value of this parameter to password.
client_id	string	The consumer key provided in the connected app.
client_secret	string	The consumer secret provided in the connected app.
username	string	Your Salesforce username.
password	string	Your Salesforce password.

Example curl Command

```
curl -v https://login.salesforce.com/services/oauth2/token
      -d "grant_type=password" -d "client_id=consumer_key"
      -d "client_secret=consumer_secret"
      -d "username=username" -d "password=password"
```

Example JSON Response

```
{
  "access_token": "00DR00000008oBT!AQwAQCPqzc_HBE59c80QmEJD4rQKRRc1GRLvYZEq...",
  "instance_url": "https://na1.salesforce.com",
  "id": "https://login.salesforce.com/id/00DR00000008oBTMAY/005R0000000IUUMIA4",
  "token_type": "Bearer",
  "issued_at": "1513887500425",
  "signature": "3PiFUIioqKkHpHxUiCCDzpvSiM2F6//w2/CslNTuf+o="
}
```

The `access_token` field in the response contains the access token value. API clients pass the access token in the Authorization header (`Authorization: Bearer` ***`access_token`***) of each request.

Use the `instance_url` field value in the response as the Salesforce instance URL in your REST API resource URIs (for example, ***`instance_url`***`/services/data/v42.0/iot/orchestrations`).

SEE ALSO:

REST API Developer Guide : Understanding the User-Agent OAuth Authentication Flow

Steps

Step One: Create a Context

Step Two: Create an Orchestration

Step Three: Update the Orchestration

Step Four: Activate the Orchestration

Step One: Create a Context

HTTP Method and URI

```
POST /iot/contexts
```

Procedure

1. Create a text file, and paste the following context definition into it.

```
{
  "description": "Context for solar panel orchestrations.",
  "events": [
    {
      "keyFields": [
        "Serial_Number__c"
      ],
      "metadata": {
        "name": "Solar_Panel_Event__e",
        "type": "PlatformEvent"
      }
    }
  ],
  "label": "Solar Panel Context",
  "name": "Solar_Panel_Context",
  "referenceData": [ {
    "keyFields" : [ "Id" ],
    "metadata" : {
      "name" : "Asset",
      "type" : "SalesforceObjectReference"
    }
  } ],
  "runtime": "Standard"
}
```

You pass in the context definition as the request body when calling the API. This context references the platform event Solar_Panel_Event__e, which you created in Define the Solar Panel Platform Event. The context also references the Asset Salesforce object. The Id Asset field is used as the key field. Typically, you reference a custom field marked as Unique for the key. Using a custom field ensures that the data that the field stores matches the key value in incoming events. However, for simplicity, we are just using the Id field here.

2. Save the file as `context.txt` in the same folder where you execute the curl command.
3. To create the context, run the following curl command using the saved text file. Replace *`access_token`* with your access token, and replace *`instance_url`* with the instance URL that you obtained after authenticating.

```
curl -H "Authorization: Bearer access_token" -H "Content-Type: application/json"
      -X POST -d @context.txt
      https://instance_url/services/data/v42.0/iot/contexts
```

Note: If the access token value contains an exclamation mark (`!`), the curl command might return an error on macOS and Linux systems. To avoid an error, use single quotes around the Authorization header (`'Authorization: Bearer 00DR00000001hMN!AQ4AQNHlz...'`) or escape the `!` character (`"Authorization: Bearer 00DR00000001hMN\!AQ4AQNHlz..."`).

Response

The response is the JSON definition of the created context, including timestamp fields.

```
{
  "createdBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "createdDate": "2017-12-28T23:21:06.000Z",
  "description": "Context for solar panel orchestrations.",
  "events": [
    {
      "keyFields": [
        "Serial_Number__c"
      ],
      "metadata": {
        "id": "01IR00000000EKAMA2",
        "label": "Solar Panel Event",
        "name": "Solar_Panel_Event__e",
        "properties": {},
        "type": "PlatformEvent"
      }
    }
  ],
  "id": "0MeR00000004C9hKAE",
  "label": "Solar Panel Context",
  "lastModifiedBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "lastModifiedDate": "2017-12-28T23:21:06.000Z",
  "name": "Solar_Panel_Context",
  "orchestrationsUrl":
"/services/data/v42.0/iot/orchestrations?contextId=0MeR00000004C9hKAE",
  "referenceData": [ {
    "keyFields" : [ "Id" ],
    "metadata" : {
      "id" : "Asset",
      "label" : "Asset",
      "name" : "Asset",
      "type" : "SalesforceObjectReference"
    }
  } ],
  "runtime": "Standard",
  "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9hKAE"
}
```

Step Two: Create an Orchestration

HTTP Method and URI

```
POST /iot/orchestrations
```

Procedure

1. Create a text file, and paste the orchestration definition in it.

```
{
  "definition": {
    "global": {
      "id": 0,
      "initial": false,
      "name": "Always",
      "rules": [
        {
          "actions": [],
          "condition": "",
          "description": "",
          "id": 0
        }
      ]
    },
    "states": [
      {
        "id": 1,
        "initial": true,
        "name": "Default",
        "rules": [
          {
            "actions": [
              {
                "stateId": 2,
                "type": "ChangeState"
              }
            ],
            "condition": "Solar_Panel_Event__e.Power_Output__c > 75",
            "id": 0,
            "when": {
              "eventSource": "Solar_Panel_Event__e",
              "type": "Event"
            }
          }
        ]
      },
      {
        "id": 2,
        "initial": false,
        "name": "Low Output",
        "rules": [
          {
            "actions": [
              {
                "fieldAssignments": [
                  {
                    "field": "Subject",
                    "value": "\"Service solar panel\""
                  },
                  {
                    "field": "Description",
                    "value": "\"Schedule service for solar panel.\""
                  }
                ],
                "id": 0,
                "isResponseIngested" : true,
                "label": "Case Output Action",
                "name": "Create",
                "recordFilters": [],
                "responseEventName": "CaseResponseEvent",
                "salesforceObjectApiName": "Case",
                "type": "SalesforceOutputAction"
              }
            ],
            "id": 0,
            "when": {
              "type": "Immediately"
            }
          }
        ]
      }
    ]
  },
  "label": "Solar Panel Orchestration",
  "name": "Solar_Panel_Orchestration",
  "orchestrationContext": {
    "name": "Solar_Panel_Context"
  }
}
```

The orchestration definition corresponds to the OrchestrationInputRepresentation JSON object representation. The orchestration definition references the context you created earlier by its name.

```
  "orchestrationContext": {
    "name": "Solar_Panel_Context"
  }
```

2. Save the file as `orchestration.txt` in the same folder where you execute the curl command.
3. To create the orchestration, run the following curl command. Replace *`access_token`* with your access token, and replace *`instance_url`* with the instance URL that you obtained after authenticating.

```
curl -H "Authorization: Bearer access_token" -H "Content-Type: application/json"
      -X POST -d @orchestration.txt
      https://instance_url/services/data/v42.0/iot/orchestrations
```

Response

The response is the definition of the orchestration that you created, in addition to timestamp fields and validation messages that indicate any errors encountered while saving the orchestration. Note the value of the `id` field that's included after the definition block and before the label (`"id": "0FF..."`). You use this ID in the next step to update the orchestration.

```
{
  "createdBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "createdDate": "2017-12-28T23:34:07.000Z",
  "definition": {
    "global": {
      "id": 0,
      "initial": false,
      "name": "Always",
      "rules": [
        {
          "actions": [],
          "condition": "",
          "description": "",
          "id": 0
        }
      ]
    },
    "states": [
      {
        "id": 1,
        "initial": true,
        "name": "Default",
        "rules": [
          {
            "actions": [
              {
                "stateId": 2,
                "type": "ChangeState"
              }
            ],
            "condition": "Solar_Panel_Event__e.Power_Output__c &gt; 75",
            "id": 0
            "when": {
              "eventSource": "Solar_Panel_Event__e",
              "type": "Event"
            }
          }
        ]
      },
      {
        "id": 2,
        "initial": false,
        "name": "Low Output",
        "rules": [
          {
            "actions": [
              {
                "fieldAssignments": [
                  {
                    "field": "Subject",
                    "value": "&amp;quot;Service solar panel&amp;quot;"
                  },
                  {
                    "field": "Description",
                    "value": "&amp;quot;Schedule service for solar panel.&amp;quot;"
                  }
                ],
                "id": 0,
                "isResponseIngested" : true,
                "label" : "Case Output Action",
                "name": "Create",
                "recordFilters": [],
                "responseEventName": "CaseResponseEvent",
                "salesforceObjectApiName": "Case",
                "type": "SalesforceOutputAction"
              }
            ],
            "id": 0,
            "when": {
              "type": "Immediately"
            }
          }
        ]
      }
    ],
    "variables": []
  },
  "deletedStates": [],
  "id": "0FFR00000004C9wOAE",
  "label": "Solar Panel Orchestration",
  "lastModifiedBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "lastModifiedDate": "2017-12-28T23:34:07.000Z",
  "name": "Solar_Panel_Orchestration",
  "orchestrationContext": {
    "id": "0MeR00000004C9hKAE",
    "label": "Solar Panel Context",
    "name": "Solar_Panel_Context",
    "runtime": "Standard",
    "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9hKAE"
  },
  "status": "Inactive",
  "url": "/services/data/v42.0/iot/orchestrations/0FFR00000004C9wOAE",
  "validationMessages": {
    "messages": []
  }
}
```

Step Three: Update the Orchestration

In this step, you add a rule to the Low Output state. The rule transitions an orchestration instance from the Low Output state to the Default state if the power output is acceptable. For example, if a solar panel had low power output and was fixed through a service case, a new event for this panel moves it to the Default state.

HTTP Method and URI

```
PUT /iot/orchestrations/orchestration_id
```

Procedure

1. To update an orchestration, pass in the entire definition of the new orchestration as the request body. Create a text file, and paste the following request body into it. This definition contains an additional rule at the end of the rules array in the Low Output state, which is highlighted in bold font.

```
{
  "definition": {
    "global": {
      "id": 0,
      "initial": false,
      "name": "Always",
      "rules": [
        {
          "actions": [],
          "condition": "",
          "description": "",
          "id": 0
        }
      ]
    },
    "states": [
      {
        "id": 1,
        "initial": true,
        "name": "Default",
        "rules": [
          {
            "actions": [
              {
                "stateId": 2,
                "type": "ChangeState"
              }
            ],
            "condition": "Solar_Panel_Event__e.Power_Output__c > 75",
            "id": 0,
            "when": {
              "eventSource": "Solar_Panel_Event__e",
              "type": "Event"
            }
          }
        ]
      },
      {
        "id": 2,
        "initial": false,
        "name": "Low Output",
        "rules": [
          {
            "actions": [
              {
                "fieldAssignments": [
                  {
                    "field": "Subject",
                    "value": "\"Service solar panel\""
                  },
                  {
                    "field": "Description",
                    "value": "\"Schedule service for solar panel.\""
                  }
                ],
                "id": 0,
                "isResponseIngested" : true,
                "label": "Case Output Action",
                "name": "Create",
                "recordFilters": [],
                "responseEventName": "CaseResponseEvent",
                "salesforceObjectApiName": "Case",
                "type": "SalesforceOutputAction"
              }
            ],
            "id": 0,
            "when": {
              "type": "Immediately"
            }
          },
          {
            "actions": [
              {
                "stateId": 1,
                "type": "ChangeState"
              }
            ],
            "condition": "Solar_Panel_Event__e.Power_Output__c >= 75",
            "id": 1,
            "when": {
              "eventSource": "Solar_Panel_Event__e",
              "type": "Event"
            }
          }
        ]
      }
    ]
  },
  "label": "Solar Panel Orchestration",
  "name": "Solar_Panel_Orchestration",
  "orchestrationContext": {
    "name": "Solar_Panel_Context"
  }
}
```

2. Save the file as `orchestration_update.txt` in the same folder where you execute the curl command.
3. To update the orchestration, run the following curl command. Replace:
   - *`access_token`* with your access token
   - *`instance_url`* with the instance URL that you obtained after authenticating

- *`orchestration_id`* with the ID of the orchestration that you saved earlier

```
curl -H "Authorization: Bearer access_token" -H "Content-Type: application/json"
      -X PUT -d @orchestration_update.txt
      https://instance_url/services/data/v42.0/iot/orchestrations/orchestration_id
```

Response

The response is the definition of the updated orchestration, in addition to other properties, such as timestamp fields and validation messages.

```
{
  "createdBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "createdDate": "2017-12-28T23:34:07.000Z",
  "definition": {
    "global": {
      "id": 0,
      "initial": false,
      "name": "Always",
      "rules": [
        {
          "actions": [],
          "condition": "",
          "description": "",
          "id": 0,
        }
      ]
    },
    "states": [
      {
        "id": 1,
        "initial": true,
        "name": "Default",
        "rules": [
          {
            "actions": [
              {
                "stateId": 2,
                "type": "ChangeState"
              }
            ],
            "condition": "Solar_Panel_Event__e.Power_Output__c &gt; 75",
            "id": 0,
            "when": {
              "eventSource": "Solar_Panel_Event__e",
              "type": "Event"
            }
          }
        ]
      },
      {
```

```
      "id": 2,
      "initial": false,
      "name": "Low Output",
      "rules": [
        {
          "actions": [
            {
              "fieldAssignments": [
                {
                  "field": "Subject",
                  "value": "&amp;quot;Service solar panel&amp;quot;"
                },
                {
                  "field": "Description",
                  "value": "&amp;quot;Schedule service for solar panel.&amp;quot;"
                }
              ],
              "id": 0,
              "isResponseIngested" : true,
              "label" : "Case Output Action",
              "name": "Create",
              "recordFilters": [],
              "responseEventName": "CaseResponseEvent",
              "salesforceObjectApiName": "Case",
              "type": "SalesforceOutputAction"
            }
          ],
          "id": 0,
          "when": {
            "type": "Immediately"
          }
        },
        {
          "actions": [
            {
              "stateId": 1,
              "type": "ChangeState"
            }
          ],
          "condition": "Solar_Panel_Event__e.Power_Output__c &gt;= 75",
          "id": 1,
          "when": {
            "eventSource": "Solar_Panel_Event__e",
            "type": "Event"
          }
        }
      ]
    }
  ],
  "variables": []
},
"deletedStates": [],
"id": "0FFR00000004C9wOAE",
"label": "Solar Panel Orchestration",
  "lastModifiedBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "lastModifiedDate": "2017-12-28T23:42:16.000Z",
  "name": "Solar_Panel_Orchestration",
  "orchestrationContext": {
    "id": "0MeR00000004C9hKAE",
    "label": "Solar Panel Context",
    "name": "Solar_Panel_Context",
    "runtime": "Standard",
    "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9hKAE"
  },
  "status": "Inactive",
  "url": "/services/data/v42.0/iot/orchestrations/0FFR00000004C9wOAE",
  "validationMessages": {
    "messages": []
  }
}
```

Step Four: Activate the Orchestration

Now that you've saved your orchestration, activate it so that it executes on incoming platform events. You can activate the most recently saved draft, and optionally, pass in options to delete all existing orchestration instances.

HTTP Method and URI

```
POST /iot/orchestrations/orchestration_id/activations
```

Procedure

1. To activate an orchestration, pass in an empty request body (`{ }`) or, activation options. For demonstration purposes,, we pass in the option to delete all running instances, even though there are none. If you don't specify this option, no instances are deleted by default. Create a text file, and paste the following in it.

   ```
   {
     "options": {
       "resetInstancesOnActivation": true
     }
   }
   ```

2. Save the file as `activation_options.txt` in the same folder where you execute the curl command.
3. To update the orchestration, run the following curl command. Replace:
   - *`access_token`* with your access token
   - *`instance_url`* with the instance URL that you obtained after authenticating
   - *`orchestration_id`* with the ID of the orchestration that you saved earlier

   ```
   curl -H "Authorization: Bearer access_token" -H "Content-Type: application/json"
         -X POST -d @activation_options.txt
   ```

```
https://instance_url/services/data/v42.0/iot/orchestrations/orchestration_id/activations
```

Response

The response is the activation representation, which includes the definition of the activated orchestration and activation status information. The orchestration definition included in the response has been omitted for brevity.

```
{
  "createdBy" : {
    "id" : "005R0000000jVvmIAE",
    "name" : "Admin User"
  },
  "createdDate" : "2017-12-29T22:05:58.000Z",
  "definition" : {
    ...
  },
  "id" : "0MlR00000004CA6KAM",
  "lastModifiedBy" : {
    "id" : "005R0000000jVvmIAE",
    "name" : "Admin User"
  },
  "lastModifiedDate" : "2017-12-29T22:05:58.000Z",
  "options" : {
    "resetInstancesOnActivation" : true
  },
  "orchestrationUrl" : "/services/data/v42.0/iot/orchestrations/0FFR00000004C9wOAE",
  "status" : "Activating",
  "url" :
"/services/data/v42.0/iot/orchestrations/0FFR00000004C9wOAE/activations/0MlR00000004CA6KAM"
}
```

CHAPTER 3 Examples: Retrieving Contexts, Orchestrations, and Usage Data

If you've already created some contexts and orchestrations, you can use the IoT REST API to retrieve them. You can also retrieve usage data to monitor your event and orchestration usage.

List All Contexts

HTTP Method and URI

```
GET /iot/contexts
```

curl Command

The following curl command retrieves all contexts. Replace:

- *access_token* with your access token
- *instance_url* with the instance URL that you obtained after authenticating

```
curl -H "Authorization: Bearer access_token" -X GET
      https://instance_url/services/data/v42.0/iot/contexts
```

Response

The request returns all contexts in your org in the following format. Summary information is returned for each context, including the context name and ID. It excludes event and reference data information.

```
{
  "contexts": [ // array of contexts
    {
      …
    },
    {
      …
    },
   // more contexts
  ],
  "totalSize": ...,
  "url": "/services/data/v42.0/iot/contexts"
}
```

For example, this response contains two contexts.

```
{
  "contexts": [
    {
      "createdBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "createdDate": "2017-12-28T23:21:06.000Z",
      "description": "Context for solar panel orchestrations.",
      "id": "0MeR00000004C9hKAE",
      "label": "Solar Panel Context",
      "lastModifiedBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "lastModifiedDate": "2017-12-28T23:21:06.000Z",
      "name": "Solar_Panel_Context",
      "runtime": "Standard",
      "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9hKAE"
    },
    {
      "createdBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "createdDate": "2017-12-28T22:15:18.000Z",
      "id": "0MeR00000004C9cKAE",
      "label": "Test",
      "lastModifiedBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "lastModifiedDate": "2017-12-28T22:15:35.000Z",
      "name": "Test",
      "runtime": "Standard",
      "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9cKAE"
    }
  ],
  "totalSize": 2,
  "url": "/services/data/v42.0/iot/contexts"
}
```

Get a Context by ID

HTTP Method and URI

```
GET /iot/contexts/context_id
```

curl Command

The following curl command retrieves a context. Replace:

- *`access_token`* with your access token
- *`instance_url`* with the instance URL that you obtained after authenticating
- *`context_id`* with the ID of the context to retrieve

```
curl -H "Authorization: Bearer access_token" -X GET
      https://instance_url/services/data/v42.0/iot/contexts/context_id
```

Response

The request returns detailed information for the specified context, including event and reference data information. For example, this context representation is returned for the Solar Panel context.

```
{
  "createdBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "createdDate": "2017-12-28T22:15:18.000Z",
  "events": [
    {
      "keyFields": [
        "Serial_Number__c"
      ],
      "metadata": {
        "id": "01IR00000000EKAMA2",
        "label": "Solar Panel Event",
        "name": "Solar_Panel_Event__e",
        "properties": {},
        "type": "PlatformEvent"
      }
    }
  ],
  "id": "0MeR00000004C9cKAE",
  "label": "Test",
  "lastModifiedBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "lastModifiedDate": "2017-12-28T22:15:35.000Z",
  "name": "Test",
  "orchestrationsUrl":
"/services/data/v42.0/iot/orchestrations?contextId=0MeR00000004C9cKAE",
  "referenceData": [ {
    "keyFields" : [ "Id" ],
    "metadata" : {
      "id" : "Asset",
      "label" : "Asset",
      "name" : "Asset",
      "type" : "SalesforceObjectReference"
    }
  } ],
  "runtime": "Standard",
  "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9cKAE"
}
```

List All Orchestrations

HTTP Method and URI

```
GET /iot/orchestrations
```

curl Command

The following curl command retrieves all orchestrations. Replace:

- *access_token* with your access token
- *instance_url* with the instance URL that you obtained after authenticating

```
curl -H "Authorization: Bearer access_token" -X GET
      https://instance_url/services/data/v42.0/iot/orchestrations
```

Response

The request returns all orchestrations in your org in this format. Summary information is returned for each orchestration, which excludes the orchestration definition.

```
{
  "orchestrations": [
    {
      …
    },
    {
      …
    },
    // more orchestrations
  ],
  "totalSize": ...,
  "url": "/services/data/v42.0/iot/orchestrations"
}
```

For example, this response contains two orchestrations.

```
{
  "orchestrations": [
    {
      "createdBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "createdDate": "2017-12-28T23:34:07.000Z",
      "id": "0FFR00000004C9wOAE",
      "label": "Solar Panel Orchestration",
      "lastModifiedBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "lastModifiedDate": "2017-12-28T23:42:16.000Z",
      "name": "Solar_Panel_Orchestration",
      "orchestrationContext": {
        "id": "0MeR00000004C9hKAE",
        "label": "Solar Panel Context",
        "name": "Solar_Panel_Context",
        "runtime": "Standard",
        "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9hKAE"
      },
      "status": "Inactive",
      "url": "/services/data/v42.0/iot/orchestrations/0FFR00000004C9wOAE"
    },
    {
      "createdBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "createdDate": "2017-12-28T20:13:07.000Z",
      "id": "0FFR00000004C7Tfsr",
      "label": "Solar Panel Orchestration",
      "lastModifiedBy": {
        "id": "005R0000000jVvmIAE",
        "name": "Admin User"
      },
      "lastModifiedDate": "2017-12-28T20:13:07.000Z",
      "name": "Robot_Orchestration",
      "orchestrationContext": {
        "id": "0MeR00000004C9hKAE",
        "label": "Solar Panel Context",
        "name": "Solar_Panel_Context",
        "runtime": "Standard",
        "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9hKAE"
      },
      "status": "Inactive",
      "url": "/services/data/v42.0/iot/orchestrations/0FFR00000004C7Tfsr"
    }
  ],
  "totalSize": 2,
  "url": "/services/data/v42.0/iot/orchestrations"
}
```

Get an Orchestration by ID

HTTP Method and URI

```
GET /iot/orchestrations/orchestration_id
```

curl Command

The following curl command retrieves a specific orchestration. Replace:

- *access_token* with your access token
- *instance_url* with the instance URL that you obtained after authenticating
- *orchestration_id* with the ID of the orchestration to retrieve

```
curl -H "Authorization: Bearer access_token" -X GET
      https://instance_url/services/data/v42.0/iot/orchestrations/orchestration_id
```

Response

The request returns detailed information for the specified orchestration, including the orchestration definition. For example, this orchestration representation is returned for the Solar Panel orchestration.

```
{
  "createdBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "createdDate": "2017-12-28T23:34:07.000Z",
  "definition": {
    "global": {
      "id": 0,
      "initial": false,
      "name": "Always",
      "rules": [
        {
          "actions": [],
          "condition": "",
          "description": "",
          "id": 0
        }
      ]
    },
    "states": [
      {
        "id": 1,
        "initial": true,
        "name": "Default",
        "rules": [
          {
            "actions": [
              {
                "stateId": 2,
                "type": "ChangeState"
              }
            ],
            "condition": "Solar_Panel_Event__e.Power_Output__c &gt; 75",
            "id": 0,
            "when": {
              "eventSource": "Solar_Panel_Event__e",
              "type": "Event"
            }
          }
        ]
      },
      {
        "id": 2,
        "initial": false,
        "name": "Low Output",
        "rules": [
          {
            "actions": [
              {
                "fieldAssignments": [
                  {
                    "field": "Subject",
                    "value": "&amp;quot;Service solar panel&amp;quot;"
                  },
                  {
                    "field": "Description",
                    "value": "&amp;quot;Schedule service for solar panel.&amp;quot;"
                  }
                ],
                "id": 0,
                "name": "Create",
                "recordFilters": [],
                "responseEventName": "CaseResponseEvent",
                "salesforceObjectApiName": "Case",
                "type": "SalesforceOutputAction"
              }
            ],
            "id": 0,
            "when": {
              "type": "Immediately"
            }
          },
          {
            "actions": [
              {
                "stateId": 1,
                "type": "ChangeState"
              }
            ],
            "condition": "Solar_Panel_Event__e.Power_Output__c &gt;= 75",
            "id": 1,
            "when": {
              "eventSource": "Solar_Panel_Event__e",
              "type": "Event"
            }
          }
        ]
      }
    ],
    "variables": []
  },
  "deletedStates": [],
  "id": "0FFR00000004C9wOAE",
  "label": "Solar Panel Orchestration",
  "lastModifiedBy": {
    "id": "005R0000000jVvmIAE",
    "name": "Admin User"
  },
  "lastModifiedDate": "2017-12-28T23:42:16.000Z",
  "name": "Solar_Panel_Orchestration",
  "orchestrationContext": {
    "id": "0MeR00000004C9hKAE",
    "label": "Solar Panel Context",
    "name": "Solar_Panel_Context",
    "runtime": "Standard",
    "url": "/services/data/v42.0/iot/contexts/0MeR00000004C9hKAE"
  },
  "status": "Inactive",
  "url": "/services/data/v42.0/iot/orchestrations/0FFR00000004C9wOAE",
  "validationMessages": {
    "messages": []
  }
}
```

Get IoT Usage Data for the Org

You can get IoT usage information for the entire org. Usage information includes allocations for events and orchestrations, the number of events processed and skipped across all orchestrations, and a link to usage information for orchestration instances.

HTTP Method and URI

```
GET /iot/usage
```

curl Command

The following curl command retrieves all orchestrations. Replace:

- *access_token* with your access token

- *instance_url* with the instance URL that you obtained after authenticating

```
curl -H "Authorization: Bearer access_token" -X GET
      https://instance_url/services/data/v42.0/iot/usage
```

Response

The response looks similar to the following. In this example, the returned allocations are for a Developer Edition org.

```
{
  "allocations": {
    "eventAllocation": {
      "count": 15000,
      "description": "Maximum allowed platform events per day"
    },
    "instanceAllocation": {
      "count": 10,
      "description": "Maximum allowed devices per orchestration"
    }
  },
  "eventUsage": {
    "processedEventCount": 6,
    "rejectedEventCount": 0
  },
  "orchestrationUsageUrl": "/services/data/v42.0/iot/usage/orchestration-usage"
}
```

Get Orchestration Usage Data

You can get usage data for each orchestration in the org, such as the number of instances, and the number of processed and skipped events per orchestration.

HTTP Method and URI

```
GET /iot/usage/orchestration-usage
```

curl Command

The following curl command retrieves all orchestrations. Replace:

- *access_token* with your access token
- *instance_url* with the instance URL that you obtained after authenticating

```
curl -H "Authorization: Bearer access_token" -X GET
      https://instance_url/services/data/v42.0/iot/usage/orchestration-usage
```

Response

The request returns usage information for each orchestration. This example contains usage data for two orchestrations.

```
{
  "orchestrationUsages": [
    {
      "activationStatus": "Active",
      "instanceCount": 3,
      "label": "Solar Panel Orchestration",
      "orchestrationId": "0FFR00000004C9wOAE",
      "orchestrationUrl": "/services/data/v42.0/iot/orchestrations/0FFR00000004C9wOAE",
      "processedEventCount": 4,
      "rejectedEventCount": 0
    },
    {
      "activationStatus": "Active",
      "instanceCount": 2,
      "label": "Robot Orchestration",
      "orchestrationId": "0FFR00000004C7Tfsr",
      "orchestrationUrl": "/services/data/v42.0/iot/orchestrations/0FFR00000004C7Tfsr",
      "processedEventCount": 2,
      "rejectedEventCount": 0
    }
  ],
  "totalSize": 2
}
```