Oracle

Field Service Cloud
Mobile Plug-in Framework

18C
# Contents

## Preface

1. **About Plug-Ins**
   - Mobile Plug-In Framework Glossary

2. **The Plug-In Framework**
   - Basics of the Plug-In Framework
   - Requirements to Use a Plug-In API
   - Using HTTPS
   - Basic HTTP Authentication
   - HMAC Authentication
   - Plug-In API Specification
   - Available Methods
   - Initialization
   - `ready` Method
   - Plug-In Action Flow
   - Available Entities and Data Collections
   - File Properties Support
   - Order of Execution of Actions
   - Action Parameters
   - Creating and Deleting Inventory
   - Returning from the Plug-In
   - Storing and Passing Sensitive Information
   - Error Handling
   - Error Codes for Entities
   - Error Codes for Actions
   - Notifying When Offline or Online
   - Property Value Length Limits
   - JSON Schema for Message Data
   - JSON Example
   - Barcode Scanner Method
   - A Sample Plug-In
### 3 Hosting Plug-Ins

- How to Host a Plug-In
- Preparing a Plug-In for Upload
- Working Offline
- Upload a Plug-In
- Modify, Download, or Delete an Archive
- Use in an Action Link

### 4 Using plug-ins

- Configure the Application to Use a Plug-In
- Using the Plug-In
- Possible Transitions
Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center.

Documentation Accessibility

For information about Oracle’s commitment to accessibility, visit the Oracle Accessibility Program website. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Contacting Oracle

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit My Oracle Support or visit Accessible Oracle Support if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides. Please take one of the following surveys:

- For web-based user guide, Web-based User Guide Survey
- For tutorial feedback, Tutorial Survey
1 About Plug-Ins

Mobile Plug-In Framework Glossary

The following table lists the terms that are used in the context of a plug-in for mobile devices that run Oracle Field Service Cloud:

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Field Service Cloud Plug-in</td>
<td>An external application that is used to extend Oracle Field Service Cloud capabilities with custom features. The external application can be an HTML 5 application, an offline HTML 5 application, or a mobile application. The application must correspond to certain requirements to be used as a Oracle Field Service Cloud plug-in.</td>
</tr>
<tr>
<td>Mobile Plug-in Extension Framework</td>
<td>General guidelines and communication protocol that an external offline supported HTML 5 application must follow to be used with Oracle Field Service Cloud as an offline capable plug-in.</td>
</tr>
<tr>
<td>Oracle Field Service Cloud Plug-in API</td>
<td>Protocol of communication between a plug-in and Oracle Field Service Cloud, which is used in the Mobile Plug-in Extension Framework.</td>
</tr>
<tr>
<td>Oracle Field Service Cloud Action link</td>
<td>Oracle Field Service Cloud configuration entity that includes all the information necessary for Oracle Field Service Cloud to invoke a plug-in or a standard form.</td>
</tr>
<tr>
<td>Oracle Field Service Cloud Action</td>
<td>A button or a link placed on an Oracle Field Service Cloud screen that invokes a plug-in or a standard Oracle Field Service Cloud action, using settings configured in the Action Link.</td>
</tr>
<tr>
<td>HTML5 Web Messaging</td>
<td>W3C defined technology that allows HTML 5 applications opened in the same browser communicate directly, without involving any server. This is the technology on top of which the Oracle Field Service Cloud plug-in API is built. Official description can be found on the W3C site <a href="https://www.w3.org/TR/webmessaging/">https://www.w3.org/TR/webmessaging/</a>. The browser that is used to work with Oracle Field Service Cloud must be compatible with this technology.</td>
</tr>
</tbody>
</table>
2 The Plug-In Framework

Basics of the Plug-In Framework

Oracle Field Service Cloud is a highly developed application that can be customized for the unique purposes and specialized business needs of organizations. That extensibility is achieved in part through the use of plug-ins, which can perform actions not found in the standard solution. Plug-ins appear as selectable links on the application. They open a new window, tab, or frame in a browser where an external HTML5 application is executed.

Plug-ins can be internal or external. Internal plug-ins affect the behavior of the Oracle Field Service Cloud platform and can modify objects stored in the database. They use the internal structures of the application and can work in the offline mode with Mobility Cloud Service. They can be created only by Oracle developers. External plug-ins, however, can be developed by anyone; they use externally stored data and communicate with the application by HTTP requests. The external plug-ins use HTML5 features such as offline work and persistence storage. The plug-in framework also allows these applications to exchange data with Oracle Field Service Cloud, in two ways:

- Traditional one-way communication when the plug-in receives data from OFSC via HTTP GET and POST parameters.
- Two-way communication using Oracle Field Service Cloud Plug-in API

The plug-in framework offers the following features:

- Integration with Oracle Field Service Cloud through an API and, therefore, the ability to perform complex tasks which could previously be performed only by internal plug-ins
- Ability to work offline with Mobility Cloud Service
- Plug-in development by your organization or third-party developers without requiring Oracle developers

The plug-ins work in Oracle Field Service Mobility Cloud Service and can manipulate the following Oracle Field Service Cloud entities:

- Resource
- Activity
- Inventory (does not include Parts Catalog)
- Activity list
- Inventory list

Consequently, the plug-ins can be added to the following Mobility Cloud Service contexts:

- Activity list
- Edit/View activity
- Inventory grid
- Add/Details inventory
Requirements to Use a Plug-In API

This topic provides the requirements to develop a plug-in for Oracle Field Service Cloud.

You must meet these requirements to develop a plug-in using the plug-in API:

- The plug-in URL must point to the main page of the plug-in source and use the HTTPS protocol. The main page must be accessible through the configured plug-in URL.
- The main page must be a valid HTML/XHTML page that can load JavaScript code sources, static resources (images, .css style sheets), or contain them directly.
- A valid offline plug-in must run JavaScript code in the main page interacting with Mobility Cloud Service.
- To make the plug-in accessible in the offline mode, the <html> tag should contain a manifest attribute whose value points to a file of the special offline manifest format describing what resource files must be cached by the browser for offline. The offline plug-in is processed entirely outside of Oracle Field Service Cloud.
- You can also load resources for offline plug-in using Service Workers. However, you must be aware of their availability in different versions of browsers.
- The plug-in can save data for offline use locally on the user’s device by means of cookies, local storage, or indexed DB.
- The plug-in is loaded into an iframe. The URL points outside the Oracle Field Service Cloud domain, therefore its application cache, cookies, local storage, and indexed DB are separated from those of the application and cannot interfere with them according to the Same origin policy. Most properties of the parent window are also unavailable for plug-in JavaScript code. As a result, the only way to interact with Mobility Cloud Service is through the plug-in API.
- It’s required to use a valid certificate, not a self-signed certificate.
- The ability to work offline must be maintained by the plug-in developer. Oracle Field Service Cloud only provides the ability to load all plug-in files on start, by opening the plug-in in an invisible iframe. The plug-in developer must take care of the application cache state and use the plug-in properly in offline mode.

Placeholders used in the URL

You can include the following placeholders in the URL to send additional parameters to the plug-in:

- {user_id}, {uid}: ID of the current user
- {date}: Current date
- {timestamp}: Current timestamp in ISO format
- {uname}: User name
- {ulanguage}: ID of the user ‘s language
- {ulogin}: User login
- {su_zid}: User timezone
- {allow_desktop_notifications}: Parameter defining whether the user allows HTML5 notifications
- {allow_vibration}: Parameter defining whether the user allows vibration alerts
Using HTTPS

The plug-in must be hosted on an HTTPS server. If your web server is not configured to use HTTPS, follow your web server documentation and configure it. Further, ensure that the plug-in is hosted on the same port as Oracle Field Service Cloud, which is port 443.

Basic HTTP Authentication

The basic HTTP authentication method is a standard method, which is part of the HTTP 1.0 standard (RFC 1945) called Basic Access Authentication. It works over HTTPS as well.

Ensure that the following conditions are met:

- **Oracle Field Service Cloud**: In the Add action link and Edit action link windows, select HTTP Basic authentication type, and fill up the Login and Password fields with valid values. These credentials are encrypted and saved to the Oracle Field Service Cloud database.

- **Server Side**: Configure the web server on which the plug-in sources are hosted to return the HTTP 401 Unauthorized status, if you are requesting the configured plug-in URL without the credentials. See the NGINX and Apache documents for details. The server must return the plug-in content if its URL is requested with the HTTP header. Authorization: Basic bXlsb2dpbjpteXBhc3M= Where bXls ... is a valid Base64-encoded pair of login:password. The credentials configured for the plug-in in the Add action link and Edit action link windows must be accepted as valid.

- **Client Side**: When the user logs in to Mobility Cloud Service, Oracle Field Service Cloud reads the credentials from the database and loads the plug-in URL into the hidden iframe as follows: <iframe src="https://mylogin:mypass@example.com/myPlugin.php"/> This way, the browser loads the plug-in sources over HTTPS using HTTP Basic Authentication:

```plaintext
GET /myPlugin.php HTTP/1.1
Host: example.com
Authorization: Basic bXlsb2dpbjpteXBhc3M=
```

**Note**: We recommend that you use HMAC authentication instead of basic HTTP authentication. This is because, Google Chrome doesn’t support the use of Basic HTTP authentication in sub-resources starting from release 59.

HMAC Authentication

HMAC (Hash-based message authentication code) lets you sign HTTP requests and their GET parameters. It authenticates to see that the data is not forged and is not received from an unauthorized source.

The MAC signature (digest) is added as an additional GET parameter at the end of query string: <!CDATA[[http://www.example.com/path?user=test&section=D%26G&activity=33&hmac=D2BJn9P1EcLhaFrNhbbAzCQTVQCCwCEQsrg8V6h4YoU%3D]]>.
HMAC function algorithm

The algorithm is defined in RFC 2104, and can be very roughly described as: $hmac = BASE64(HMAC - SHA - 256(data, SHA256(SecretKey)))$. SHA - 256 accepts SecretKey as a string and returns the hash string. The secret key is configured per plug-in in the Add action link and Edit action link windows in Core Manage Cloud Service, hashed by SHA256, encrypted and stored in the database. HMAC-SHA-256 accepts data and key as strings and returns a binary array of HMAC signature. BASE64 accepts the binary array and returns BASE64 encoded string. Data for HMAC generation is query resource location with query parameters sorted lexicographically:

- Remove the protocol identifier from the URL together with colon and slashes (http:// or https://).
- Remove the resource name and port from the URL.
- Append query location to the output string.
- If there are query parameters append the character ? to the output string.
- Decode every name and value for URL parameters.
- Sort the list of parameters alphabetically by name.
- For each name/value pair:
  - Append the encoded name to the output string.
  - Append the ‘=’ character to the output string.
  - Append the encoded value to the output string.
- If there are more key/value pairs remaining, append an & character to the output string.

Example: Request URL: http://www.example.com/path?user=test&section=D%26G&activity=33

SecretKey: 'mysecret'

2. www.example.com/path?user=test&section=D%26G&activity=33 => /path?user=test&section=D%26G&activity=33
3. data = '/path'
4. data = '/path/'
5. ['user'='test', 'section'='D&G', 'activity'=33]
6. ['activity'=33, 'section'='D&G', 'user'='test']
7. ['activity'=33, 'section'='D&G', 'user'='test'] => data
8. data = '/path? activity'
9. data = '/path? activity='
10. data = '/path? activity=33'
11. data = '/path? activity=33&'
12. data = '/path? activity=33&section=D%26G&user=test'

$hmac = BASE64(HMAC-SHA-256('/path?activity=33&section=D%26G&user=test',SHA256('mysecret'))) = BASE64(HMAC-SHA-256('/path?activity=33&section=D%26G&user=test', '652c7dc687d98c9889304ed2e408c74b611e86a40c51c4b43f1dd5913c5cd0')) = BASE64(['0f,60,49,9f,d3,f5,11,c2,e1,68,5a,cd,85,b0,33,09,04,d5,41,70,82,c0,20,50,b2,b8,3c,57,a8,78,62,85]) = 'D2BJn9P1EcLhaFrNhbaZCQTVQCvCBQsrg8V6h4YoU='

The full signed URL is http://www.example.com/path?user=test&section=D%26G&activity=33&hmac=D2BJn9P1EcLhaFrNhbaZCQTVQCvCBQsrg8V6h4YoU%3D
Plug-In API Specification

The plug-in API is based on cross-window messaging with postMessages, which can be sent by Oracle Field Service Cloud and received by the plug-in and vice-versa. Messages are sent by JavaScript code using `window.postMessage()` method, and received by subscribing to messages using `window.addEventListener()`.

Message format

The following message format is supported:

```
window.frames['my_plugin'].postMessage({
  "apiVersion": 1,
  "method": "open",
  "entity": "activity",
  "resource": {
    "pid": 5000038
  }
}, targetOrigin);
```

Message processed by the plug-in:

```javascript
function getPostMessageData(event) {
  var data = JSON.parse(event.data);
  switch (data.method) {
    case 'open':
      pluginOpen(data);
      break;
    default:
      showError();
  }
}

window.addEventListener("message", _getPostMessageData, false);
```

JSON data is an object (hash) of defined format containing special fields (which describe the message itself) and data fields (which contain the data of Oracle Field Service Cloud entities), for example:

```json
{
  "apiVersion": 1,
  "method": "open",
  "entity": "activity",
  "resource": {
    "pid": 5000038
  },
  "inventories": {
    "20997919": {
      "invid": 20997919,
      "inv_pid": 5000038
    }
  }
}
```

where:

- `apiVersion`, `method`, `entity` – special fields
- `resource`, `inventories` – entity data collections available only for the ‘open’ and ‘close’ methods

Special fields:

- `apiVersion` – version of the plug-in API used for interaction between Mobility Cloud Service and the plug-in. It defines the available methods and data
- `method` – describes the action initiated by Oracle Field Service Cloud or the plug-in and the actions to be performed on the other side
Available Methods

Methods initiated by Oracle Field Service Cloud

The methods initiated by Oracle Field Service Cloud are:

- **init**: Notifies the plug-in that it should perform initialization. For example, load metadata or offline files from the server.
- **open**: The plug-in content is to be shown on the Mobility Cloud Service screen.
- **error**: The data submitted by the plug-in is invalid or internal errors have occurred.
- **wakeup**: Notifies the plug-in about the presence of network connectivity. The plug-in can perform its own synchronization after receiving this message.

Methods initiated by the plug-in

The methods initiated by the plug-in are:

- **ready**: The plug-in has been loaded and is ready to receive messages.
- **initEnd**: The plug-in notifies Oracle Field Service Cloud that it has finished initialization and can be suspended until opened by the user.
- **close**: The plug-in submits data after which its window will be closed if the data is valid.
- **sleep**: If the plug-in is not able to synchronize with its own server due to absence of network connectivity, it can notify Oracle Field Service Cloud to wake it up in the background when connectivity is available.

Initialization

The following figure shows the flow of steps in initializing a plug-in:

- Create an iframe, if required
- Execute the ready method
- Execute the init method
- Execute the initEnd method; here, the wakeupNeeded parameter is set to ‘true’
- Destroy the iframe, if it is created
ready Method

These messages are always sent in the following format:

```json
{
   "apiVersion": 1,
   "method": "ready"
}
```

The optional “dataItems” field can be included in the message to limit the amount of data sent to plug-in in open method. See “Limit the amount of data sent to the plug-in” for more details.

Limit the Amount of Data Sent to the Plug-In

Mobility sends all data available for entity collections, with the “open” message. The data is sent according to the layout, where the plug-in’s action link is placed and visibility of properties on plug-in API layouts. This means that when plug-in is opened from Activity List screen, the data of all activities for selected day’s queue will be sent with the data of all non-scheduled activities of selected resource. To reduce the amount of data collected and serialized by Mobility and un-serialized by a plug-in, the new optional parameter dataItems is added for the “ready” message. This decreases loading time of the plug-in. The value of this parameter defines which items are present in the entity collections. Using this parameter the plug-in can prevent Mobility from sending certain items in available entity collections, but it can’t broaden the set of entity collections sent to the plug-in. This set is predefined and depends on the screen from which the plug-in is opened.

Hiding the Header When the Plug-In is Open

When a user opens a plug-in, it is shown in Oracle Field Service Mobility Cloud Service and the header of the screen is rendered by Oracle Field Service Cloud. The user can exit the plug-in by clicking **Back** in the header. The plug-in will be
closed without sending any data to Oracle Field Service Cloud. In some business flows, it is preferred to hide the header of the screen so that the plug-in can:

- Render the header itself
- Based on the business flow, decide whether the user can leave the plug-in

To support this function, the "ready" method is extended, and a new flag, "showHeader" is added:

- If flag value is set to TRUE, then the Oracle Field Service Mobility Cloud Service header is shown.
- If flag value is set to FALSE, then the Oracle Field Service Mobility Cloud Service header is hidden.

Preventing the User from Using Back

When a user opens a plug-in, it is shown in Oracle Field Service Mobility Cloud Service and the header of the screen is rendered by Oracle Field Service Cloud. The user can exit the plug-in by clicking Back in the header or the browser. The plug-in will be closed without sending any data to Oracle Field Service Cloud. In some business flows, it is preferred to prevent the user from returning using any of the options. Based on the business flow, the plug-in decides whether the user must exit the plug-in or not. To support this function, the "ready" method is extended, and a new flag, "enableBackButton" is added:

- If flag value is set to TRUE, then the Oracle Field Service Mobility Cloud Service Back button is shown and the navigation is not locked.
- If flag value is set to FALSE, then the Oracle Field Service Mobility Cloud Service Back button is hidden and the navigation is locked. This means, the browser’s native Back and Forward buttons are disabled.

Example for ready Method

```javascript
// the header is shown but the "back" button is hidden:
{
  "apiVersion": 1,
  "method": "ready",
  "showHeader": true,
  "enableBackButton": false
}

// the header is hidden but the user can go back using browser's back button:
{
  "apiVersion": 1,
  "method": "ready",
  "showHeader": false,
  "enableBackButton": true
}

// the header is hidden and the user can leave the plugin only when the plugin sends the "close" message via Plugin API (the browser's back button does not work):
{
  "apiVersion": 1,
  "method": "ready",
  "showHeader": false,
  "enableBackButton": false
}
```
Plug-In Action Flow

This topic describes the flow of the plug-in, when a user opens it to perform an action.

The following figure describes the flow of the plug-in, when a user opens it to perform an action:

The flow of steps is as follows:

- The user clicks the action link in Oracle Field Service Mobility Cloud Service.
- The iframe is created, if required.
- The Ready method is executed.
- The Open method is executed.
- The Close method is executed; here, the wakeupNeeded parameter is set to ‘true’.
- The iframe is destroyed, if it is created.

Available Entities and Data Collections

The ‘entity’ field and entity data collections are available only for methods ‘open’ and ‘close’. The value of the special ‘entity’ field depends on the Oracle Field Service Mobility Cloud Service screen on which the plug-in is used. The availability of entity data collections sent within the message data depends on the value of ‘entity’.
The following table describes the entities and the data collections available for them:

<table>
<thead>
<tr>
<th>Screen</th>
<th>Entity Field Value</th>
<th>Available Collections</th>
</tr>
</thead>
</table>
| Activity List | activityList | • resource
| | | • activityList
| Activity List > Inventory List | inventoryList | • inventoryList
| Activity List > Activity Details | activity | • resource
| Activity List > Activity Details> Inventory List | activityInventoryList | • activity
| | | • inventoryList
| Activity List > Inventory List > Inventory Details | inventory | • resource
| | | • activityList
| | | • inventory
| Activity List > Activity Details > Inventory List > Inventory Details | activityInventory | • resource
| | | • activity
| | | • inventory

Note: Resource properties cannot be set or updated through the plug-in API. Resource collection is intended to show additional information in the plug-in.

Entity data collections
- resource – element in the resource tree representing a defined company asset
- activity – entity of Oracle Field Service Cloud that represents any time-consuming activity of the resource
- activities – activity list
- inventory – equipment that can be installed or de-installed during an activity
- inventories – inventory list

File Properties Support

The "close" method is extended with the support of file properties. The file properties can be sent both through data collections and through actions. Due to performance limitations, it's not rational to send the file contents using JSON strings, so the plug-in API now accepts raw JS objects as values for PostMessage data. The original JSON strings are still supported, so backward compatibility with existing plug-ins framework implemented in release 17.2 is retained, but the file properties cannot be updated in this case.

The value of each file property must be an object that has two fields:
- fileName - name of the file, which is shown on the user interface
- fileContents - Blob object, which contains the file contents. It can be constructed and filled with the generated data by JS code in runtime, or just obtained from file input and sent to Oracle Field Service Mobility Cloud Service without any transformation, as the File object inherits the Blob.
The file can be deleted by setting the appropriate property to empty string or null.

**Note:** The Image property lets you to set the coordinates through the Oracle Field Service Mobility Cloud Service interface, but there is no support for setting the coordinates in the image property in the plug-in API.

Example: How to Send the Uploaded File

```javascript
var file = document.querySelector('input[type=file]').files[0];

window.parent.postMessage
{
{
apiVersion: 1,
method: 'close',
activity:
{
ccity: 'Cleveland',
door_photo:
{
fileName: 'DCIM_20170425_203115.jpg',
fileContents: file
}
},
targetOrigin
});
```

Example: How to Send the Generated File Contents

```javascript
var text = '<?xml version="1.0" encoding="UTF-8"?>' + '<test></test>';

var blob = new Blob([text], { type: 'text/xml' });

window.parent.postMessage
{
{
apiVersion: 1,
method: 'close',
activity:
{
ccity: 'Cleveland',
XML_DATA_PROP:
{
fileName: 'test_data.xml',
fileContents: blob
}
},
targetOrigin
});
```

Example: How to Delete a File

```javascript
window.parent.postMessage
{
{
apiVersion: 1,
method: 'close',
activity:
```
Order of Execution of Actions

Actions are applied in the same order, as sent in the "actions" array of the plug-in. Actions are executed after applying all data collection updates that are sent by plug-in in the same "close" message. No actions are applied if errors occur during the validation of data collections and the actions that are sent by the plug-in. All validation errors are sent to the plug-in within the "error" message. If some actions fail to execute, the remaining actions are applied, and the execution errors are sent to the plug-in within the "error" message.

Each error contains the ID of the action that has failed or that doesn’t pass validation. ID is the order number of the action in the actions list, sent by plug-in. So, Oracle Field Service Mobility Cloud Service processes the "close" message as follows:

1. Validate entity data collections.
2. Validate actions.
3. If there are any validation errors, send an "error" message to the plug-in; otherwise proceed to the next step.
4. Apply data collections update.
5. Apply actions.
6. If there are any errors, occurred during update of data collections or execution of actions, send "error" message to a plug-in; otherwise proceed to the next step.
7. Close the plug-in window.

Action Parameters

Each action is an object, whose fields are the action parameters. Every action must contain at least two fields (parameters):

- entity - must be equal to "inventory"
- action - must be equal to one of the supported inventory actions (e.g. "install", "create")

Parameters that are specific to each action are described in the section Supported inventory actions.

⚠️ Note: Parameters that contain the ID of entities (inv_id, inv_aid, inv_pid) are of the type "string" and not "number". This is because, entities created on the client side have the IDs similar to "1234567890-1234" before they’re synchronized with the server.

Labels and values of all parameters are case-sensitive, for example, all these parameters are invalid:

```json
{
    ACTION: "INSTALL",
    entity: "Inventory",
    Inv_Aid: ""
}
```
Creating and Deleting Inventory

Usually inventory data is updated by a plug-in using data collections such as "inventoryList" and "inventory" of the Close method. It is not possible to create or delete inventories using data collections. You can do it using the optional field Actions added to the Close method. The plug-ins support the following actions:

- Install
- Deinstall
- Undo install
- Undo deinstall
- Create
- Delete

Each action is an object, and its fields are the action parameters. Every action must contain at least two fields (parameters)—entity and action. Entity must be inventory and action must include one of the actions mentioned earlier.

Format of the dataItems parameter: The dataItems parameter is an array where each item is a label of a certain data subset. If the item with the label of a subset is absent in the array, then Mobility will not send the corresponding items in the entity collections of the "open" message. If the dataItems parameter is not set in the "ready" message, no filtering is applied and full data set is sent to the plug-in. The following table provides the available keys and data subsets:

<table>
<thead>
<tr>
<th>Key</th>
<th>Affected Collections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource</td>
<td>resource</td>
<td>Properties of the currently selected resource</td>
</tr>
<tr>
<td>scheduledActivities</td>
<td>activityList</td>
<td>Activities, scheduled (belongs to the queue) for the selected date</td>
</tr>
<tr>
<td>nonScheduledActivities</td>
<td>activityList</td>
<td>Non-scheduled activities, that do not belong to any date's queue</td>
</tr>
<tr>
<td>resourceInventories</td>
<td>inventoryList</td>
<td>Inventories in the &quot;provider&quot; pool</td>
</tr>
<tr>
<td>installedInventories</td>
<td>inventoryList</td>
<td>Inventories in the &quot;install&quot; pool</td>
</tr>
<tr>
<td>deinstalledInventories</td>
<td>inventoryList</td>
<td>Inventories in the &quot;deinstall&quot; pool</td>
</tr>
<tr>
<td>customerInventories</td>
<td>inventoryList</td>
<td>Inventories in the &quot;customer&quot; pool</td>
</tr>
</tbody>
</table>

Support for Non-Serialized Inventory

You can install or de-install non-serialized inventory such as cable or faceplates through plug-in APIs. You can perform the following actions with non-serialized inventory:

- Create an inventory assigned to a resource
- Create a customer inventory linked with a specific activity
- Create an inventory in the "installed" pool
- Create an inventory in the "deinstalled" pool
- Delete inventory
When non-serialized inventory such as 10 feet (or 10 meters) of cable is installed, depending on whether the inventory of the same type has been installed before, the plug-in API performs the following steps:

- If there is no inventory of the same type (or type and model, depending on the configuration) in the "installed" pool, then it is created and the quantity is set to the amount installed, for example, 10.

- If installed inventory exists, it is updated and it’s quantity is increased by the amount installed. For example, there is 20 feet (or 20 meter) of cable already installed, then the total amount is set to 30.

The same steps are performed when you de-install inventory.

**Supported Inventory Actions**

**Install:** The following table describes the parameters for the install inventory action:

<table>
<thead>
<tr>
<th>Param Name</th>
<th>Mandatory</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invid</td>
<td>Yes</td>
<td>String</td>
<td>ID of the existing inventory that is in the &quot;provider&quot; pool of the current resource or their teammates.</td>
</tr>
<tr>
<td>inv_aid</td>
<td>Yes</td>
<td>String</td>
<td>ID of the started activity. Inventory will be installed to its &quot;install&quot; pool. Must contain the ID of the started segment for multi-day activities.</td>
</tr>
</tbody>
</table>
| quantity   | Yes/No    | Number | • Is mandatory and must be > 0 for non-serialized inventory types.  
          |           |                                  | • Is forbidden for serialized inventory types. |
| properties | No        | Object | • Is a key-value object, where keys are the labels of inventory properties to be updated.  
          |           |                                                                  | • Properties are validated and processed according to the Inventory properties for Mobility layout of the logged in user’s User Type. So, properties that you want to update must be added to this layout and the Read/Write or Mandatory visibility must be set. |

**De-install:** The following table describes the parameters for the de-install inventory action:

<table>
<thead>
<tr>
<th>Param name</th>
<th>Mandatory</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invid</td>
<td>Yes</td>
<td>String</td>
<td>ID of the existing inventory that is in the &quot;customer&quot; pool of the current resource or their teammates.</td>
</tr>
<tr>
<td>inv_pid</td>
<td>Yes</td>
<td>String</td>
<td>ID of the current resource or their teammates. Inventory will be de-installed to its &quot;deinstall&quot; pool.</td>
</tr>
</tbody>
</table>
| quantity    | Yes/No    | Number | • Is mandatory and must be > 0 for non-serialized inventory types  
          |           |                                                                  | • Is forbidden for serialized inventory types |
| properties  | No        | Object | • Is a key-value object, where keys are the labels of inventory properties to be updated.  
          |           |                                                                  | • Properties are validated and processed according to the Inventory properties for Mobility layout of the logged in user’s User Type. So, properties that you want to update must be added to this layout and the Read/Write or Mandatory visibility must be set. |

**Undo-install:** The following table describes the parameters for the undo install inventory action:
## Undo-deinstall

The following table describes the parameters for the undo-deinstall inventory action:

<table>
<thead>
<tr>
<th>Param name</th>
<th>Mandatory</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invid</td>
<td>Yes</td>
<td>String</td>
<td>ID of the existing inventory that is in the &quot;de-install&quot; pool of the current resource or their teammates.</td>
</tr>
</tbody>
</table>
| quantity   | Yes/No    | Number | • Is mandatory and must be > 0 for non-serialized inventory types.  
• Is forbidden for serialized inventory types |
| properties | No        | Object | • Is a key-value object, where keys are the labels of inventory properties to be updated.  
• Properties are validated and processed according to the Inventory properties for Mobility layout of the logged in user’s User Type. So, properties that you want to update must be added to this layout and the Read/Write or Mandatory visibility must be set. |

## Create

The following table describes the parameters for the create inventory action:

<table>
<thead>
<tr>
<th>Param name</th>
<th>Mandatory</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| invtype    | Yes       | String | Label of one of the configured Inventory Types, for example "NT".  

| invpool    | Yes       | String | Inventory pool in which the inventory will be created. It is one of: "customer", "install", "deinstall", and "provider". |
| inv_aid    | Yes/No    | String | • ID of the started activity. Inventory will be created in its pool. Must contain the id of started segment for multi-day activities.  
• Is mandatory if invpool is one of: "customer", "install", "deinstall".  
• Is forbidden for invpool equal to "provider". |
| inv_pid    | Yes/No    | String | • ID of the current resource or their teammates. Inventory will be created in the resource’s pool.  
• Is mandatory if invpool is one of: "provider", "install", "deinstall".  
• Is forbidden for invpool equal to "customer". |
| quantity   | Yes/No    | Number | • Is mandatory and must be > 0 for non-serialized inventory types  
• Is forbidden for serialized inventory types |
Chapter 2

The Plug-In Framework

### Param name | Mandatory | Type | Description
---|---|---|---
properties | No | Object | • Is a key-value object, where keys are the labels of inventory properties to be updated.

*Note:* When quantity is not present in the screen configuration Plugin API > Inventory properties for Mobility or is present and set to Read only, then it is set to "1" for non-serialized inventory by Oracle Field Service Mobility Cloud Service.

Delete: The following table describes the parameters for the delete inventory action:

### Param name | Mandatory | Type | Description
---|---|---|---
invid | Yes | String | Id of the existing inventory that is in the "provider" pool of the current resource or their teammates or in "install", "deinstall" or "customer" pool of the started activity. The quantity parameter is not available for the Delete action. The entire record with any quantity will be deleted from the selected pool.

Returning from the Plug-In

The "close" method allows a plug-in to define the screen to which the user is redirected to, after executing this method. The behavior is controlled by the "backScreen" parameter.

Previously, there were four possible values for this parameter:

- **default** - the user is redirected to the screen that was opened before the plug-in was opened. For example, if plug-in was open from the activity list then the user is redirected back to the activity list
- **activity_by_id** - the user is redirected to the activity details of the activity that is identified by the id set in the parameter "backActivityId". If it is not possible to do then the user is redirected to the activity list
- **next_activity** - the user is redirected to the activity details of the next pending activity in the list. If it is not possible to do then the user is redirected to the activity list
- **activity_list** - the user is redirected to the activity list

Now, the list of possible values has been extended to:

- **start_activity** - the user is redirected to the "start activity" screen of the activity that is identified by the id set in the parameter "backActivityId". If it is not possible to do then the user is redirected to the activity list
- **end_activity** - the user is redirected to the "complete activity" screen of the activity that is identified by the id set in the parameter "backActivityId". If it is not possible to do then the user is redirected to the activity list
- **cancel_activity** - the user is redirected to the "cancel activity" screen of the activity that is identified by the id set in the parameter "backActivityId". If it is not possible to do then the user is redirected to the activity list
• notdone_activity - the user is redirected to the "not done" screen of the activity that is identified by the id set in the parameter "backActivityId". If it is not possible to do then the user is redirected to the activity list.

• suspend_activity - the user is redirected to the "suspend activity" screen of the activity that is identified by the id set in the parameter "backActivityId". If it is not possible to do then the user is redirected to the activity list.

• delay_activity - the user is redirected to the "suspend activity" screen of the activity that is identified by the id set in the parameter "backActivityId". If it is not possible to do then the user is redirected to the activity list.

⚠️ **CAUTION:** Both Oracle Field Service Mobility Cloud Service and the plug-in use the same global browser history object. If the plug-in accidentally corrupts the history object, then it will not be possible to return from the plug-in to Oracle Field Service Mobility Cloud Service using the Back button. Work carefully with the browser history in your plug-ins.

Example of "close" Method with Different "backScreen" Cases

```json
// start_activity
{
  "apiVersion": 1,
  "method": "close",
  "backScreen": "start_activity",
  "backActivityId": "4225473"
}

// end_activity
{
  "apiVersion": 1,
  "method": "close",
  "backScreen": "end_activity",
  "backActivityId": "4225473"
}

// cancel_activity
{
  "apiVersion": 1,
  "method": "close",
  "backScreen": "cancel_activity",
  "backActivityId": "4225473"
}

// notdone_cancel
{
  "apiVersion": 1,
  "method": "close",
  "backScreen": "notdone_activity",
  "backActivityId": "4225473"
}

// suspend_activity
{
  "apiVersion": 1,
  "method": "close",
  "backScreen": "suspend_activity",
  "backActivityId": "4225473"
}

// delay_activity
{
  "apiVersion": 1,
  "method": "close",
  "backScreen": "delay_activity",
  "backActivityId": "4225473"
}
```
Storing and Passing Sensitive Information

You can store and pass sensitive data such as login credentials in the plug-in.

You add this information in the Secure parameters section in the Add action link or Edit action link screens, when you configure a plug-in. The Secure parameters section includes the Parameter name and Parameter value fields. The data entered in these fields is encrypted and stored in the database. Changes to this data are sent to Oracle Field Service Mobility Cloud Service during the next synchronization. This data is sent to the plug-in when the next message is sent. The plug-in also receives the up-to-date data with every message.

When sensitive data is sent to the plug-in, the message contains the new field securedData. securedData is included only if at least one key-value pair is configured on the Add action link screen. The init, open, and wakeup methods support the securedData field.

Format of securedData: securedData field is an object, where:

- Each key is a String, which equals to the contents of "key" text input on the Add action link or the Edit action link screen.
- Each value is a String, which equals to the contents of "value" text input for the corresponding key on the Add action link or the Edit action link screen.
- Order of entries may not be identical to the order of key-value pairs on the Add action link or the Edit action link screen.

Example of the ‘open’ method data for Supervisor plug-in: Consider the following configuration:

```json
{
  "apiVersion": 1,
  "method": "open",
  "entity": "activityList",
  "resource": {
    "external_id": "33001",
    "manager": "admin"
  },
  "activityList": {
    "4224031": {
      "aid": "4224031"
    }
  }
}
```
Error Handling

When Oracle Field Service Cloud receives a message from a plug-in with the close method, it validates all entity properties and their values. It applies the updates only if no rules are violated. After updating, the plug-in is closed. Otherwise, the application sends a message with the error method containing the list of found errors.

Error method

The error message contains no entity collections, instead, it includes the ‘errors’ field containing the list of errors:

```json
{
    "apiVersion": 1,
    "method": "error",
    "entity": "activityList",
    "errors": [
        {
            "type": "TYPE_ENTITY_PROPERTY",
            "code": "CODE_ACTIVITY_STATUS_INVALID",
            "entity": "activity",
            "entityId": "3956532",
            "propertyLabel": "astatus"
        },
        {
            "type": "TYPE_ENTITY_PROPERTY",
            "code": "CODE_MANDATORY_PROPERTY_EMPTY",
            "entity": "inventory",
            "entityId": "20998086",
            "propertyLabel": "inv_aid"
        }
    ]
}
```

Each element of the error list is an object always containing the following fields:

- **type**: Describes the type of error which occurred during the message processing, for example, invalid property value, internal error, and so on. Type determines the additional fields available in the error object, for example, property label.
- **code**: Contains a more detailed description of the error, for example, validation rule violated by the data sent by the plug-in.

Each element may optionally contain additional fields, such as entity, entityId, propertyLabel depending on the type of the error.

The following table describes the types of errors:
<table>
<thead>
<tr>
<th>Type</th>
<th>Occurs When…</th>
<th>Available Message Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error types related to entities</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE_ENTITY_ACTION</td>
<td>The requested action is not applicable for the specified entity.</td>
<td>entity entityId</td>
</tr>
<tr>
<td>TYPE_ENTITY_PROPERTY</td>
<td>The value of one of the properties submitted by the plug-in to be updated is invalid. or It violates some business rule for the given entity as well as the conditions.</td>
<td>entity entityId propertyLabel</td>
</tr>
<tr>
<td>TYPE_INTERNAL</td>
<td>Oracle Field Service Cloud is unable to process the message due to: Invalid format or contents of the message, or Unexpected internal error</td>
<td></td>
</tr>
<tr>
<td><strong>Error types related to action</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE_ACTION_ERROR</td>
<td>Action have an invalid format or in applicable</td>
<td>actionId entity entityId</td>
</tr>
<tr>
<td>TYPE_ACTION_PARAM</td>
<td>Action param has an invalid value or mandatory param is missing</td>
<td>actionId entity entityId paramName</td>
</tr>
<tr>
<td>TYPE_ACTION_PROPERTY</td>
<td>Value of one of the properties, submitted by the plug-in to be updated in the &quot;properties&quot; param, has invalid value</td>
<td>actionId entity entityId propertyLabel</td>
</tr>
<tr>
<td>TYPE_ACTION_FAILED</td>
<td>Action is rejected due to incorrect value of action params, which can’t be checked at the validation stage</td>
<td>actionId entity entityId</td>
</tr>
</tbody>
</table>

Available message fields:

- **entity**: The entity whose data is invalid (activity or inventory).
- **entityId**: The ID of the entity whose data is invalid (equals aid for activity and invid for inventory, for example, 10028719).
- **actionId**: The zero-based order number of erroneous action in the actions list, sent by plugin. E.g. 0, 1, 17 etc.
- **propertyLabel**: The label of the property, the value of which is invalid, for example, customer_number, WO_TYPE.
# Error Codes for Entities

The following table describes the error codes for entities:

<table>
<thead>
<tr>
<th>Code</th>
<th>Occurs When</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_ENTITY_ACTION</td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_ON_PAST_DATE_NOT_ALLOWED</td>
<td>The requested action is forbidden for the entity if it is assigned for an archived (past) route:</td>
</tr>
<tr>
<td></td>
<td>Updating properties of activity which is in the past and overnight/overtime limit has elapsed</td>
</tr>
<tr>
<td></td>
<td>Updating properties of inventory in the customer, installed, or deinstalled pool of the activity</td>
</tr>
<tr>
<td></td>
<td>which is in the past and overnight or overtime limit has elapsed</td>
</tr>
<tr>
<td>TYPE_ENTITY_PROPERTY</td>
<td></td>
</tr>
<tr>
<td>CODE_PROPERTY_VALUE_TOO_LARGE</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>Property type is field and the length of its value exceeds 119 UTF-16 code points</td>
</tr>
<tr>
<td></td>
<td>Property type is file, its GUI type is signature, and the length of its value exceeds 102400</td>
</tr>
<tr>
<td></td>
<td>UTF-16 code points</td>
</tr>
<tr>
<td></td>
<td>Property is neither field nor signature and the length of its value exceeds 32767 UTF-16 code</td>
</tr>
<tr>
<td></td>
<td>points</td>
</tr>
<tr>
<td>CODE_MANDATORY_PROPERTY_EMPTY</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>For activity: astatus value is empty</td>
</tr>
<tr>
<td></td>
<td>For inventory:</td>
</tr>
<tr>
<td></td>
<td>invpool is install, deinstall or customer and inv_aid value is empty</td>
</tr>
<tr>
<td></td>
<td>invpool is install, deinstall or provider and inv_pid value is empty</td>
</tr>
<tr>
<td></td>
<td>invpool value is empty</td>
</tr>
<tr>
<td>CODE_ACTIVITY_STATUS_INVALID</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>astatus of activity is not equal to any of the following: pending, started, complete, suspended,</td>
</tr>
<tr>
<td></td>
<td>notdone, cancelled</td>
</tr>
<tr>
<td></td>
<td>Transition from the current activity status to the new one specified in astatus is not allowed.</td>
</tr>
<tr>
<td>CODE_INVENTORY_POOL_INVALID</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>invpool of inventory is not equal to any of the following: customer, install, deinstall, provider</td>
</tr>
<tr>
<td></td>
<td>Transition from the current inventory pool to the new one specified in invpool is not allowed.</td>
</tr>
<tr>
<td>CODE_INVENTORY_AID_INVALID</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>invpool of inventory is provider and inv_aid value is not empty</td>
</tr>
<tr>
<td></td>
<td>invpool of inventory is customer or deinstall and the submitted inv_aid value is not equal to</td>
</tr>
<tr>
<td></td>
<td>the current value of inv_aid</td>
</tr>
<tr>
<td></td>
<td>inv_aid is not equal to aid of the started activity in the same route and the submitted inv_aid</td>
</tr>
<tr>
<td></td>
<td>value is not equal to the current value of inv_aid</td>
</tr>
<tr>
<td>CODE_INVENTORY_PID_INVALID</td>
<td>Any of the following:</td>
</tr>
<tr>
<td></td>
<td>invpool of inventory is provider, install or deinstall and the submitted inv_pid value is not equal</td>
</tr>
<tr>
<td></td>
<td>to the current value of inv_pid</td>
</tr>
<tr>
<td>CODE_ACTIVITY_STATUS_INVALID_FOR_FUTURE</td>
<td>astatus is not pending or cancelled and activity is assigned for the day in future relative to the</td>
</tr>
<tr>
<td></td>
<td>current date in the resources time zone</td>
</tr>
</tbody>
</table>
### Error Codes for Actions

The following table describes the error codes for actions:

<table>
<thead>
<tr>
<th>Code</th>
<th>Caused by Action</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_ACTION_ERROR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_ON_PAST_DATE_NOT_ALLOWED</td>
<td>• install&lt;br&gt;• deinstall&lt;br&gt;• undo_install&lt;br&gt;• undo_deinstall&lt;br&gt;• create&lt;br&gt;• delete</td>
<td>Any of these:&lt;br&gt;• &quot;inv_aid&quot; param of &quot;install&quot;, &quot;deinstall&quot;, &quot;undo_install&quot; or &quot;undo_deinstall&quot; action is equal to id of activity that is assigned for past date&lt;br&gt;• &quot;inv_aid&quot; param of &quot;create&quot; or &quot;delete&quot; action is equal to id of activity that is assigned for past date, and &quot;invpool&quot; is &quot;customer&quot;, &quot;install&quot; or &quot;deinstall&quot;</td>
</tr>
<tr>
<td>CODE_ENTITY_ID_INVALID</td>
<td>• install&lt;br&gt;• deinstall&lt;br&gt;• undo_install&lt;br&gt;• undo_deinstall&lt;br&gt;• create&lt;br&gt;• delete</td>
<td>&quot;invid&quot; param is not equal to id of any inventory in available pools</td>
</tr>
<tr>
<td>CODE_ACTION_UNKNOWN</td>
<td></td>
<td>&quot;action&quot; param is not equal to one of the supported inventory actions (e.g. &quot;install&quot;, &quot;create&quot;)</td>
</tr>
<tr>
<td>CODE_ACTION_ENTITY_UNKNOWN</td>
<td></td>
<td>&quot;entity&quot; param is not equal to &quot;inventory&quot;</td>
</tr>
<tr>
<td>TYPE_ACTION_PARAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_AID_INVALID</td>
<td>create</td>
<td>&quot;inv_aid&quot; param is sent for &quot;create&quot; action, and &quot;invpool&quot; is &quot;provider&quot;</td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_PID_INVALID</td>
<td>• deinstall&lt;br&gt;• create</td>
<td>Any of these:&lt;br&gt;• &quot;inv_pid&quot; param value is not equal to id of current resource or his teammates</td>
</tr>
<tr>
<td>Code</td>
<td>Caused by Action</td>
<td>Cause</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_POOL_INVALID</td>
<td>create</td>
<td>&quot;invpool&quot; param value is not equal to one of: &quot;customer&quot;, &quot;install&quot;, &quot;deinstall&quot;, &quot;provider&quot;</td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_TYPE_INVALID</td>
<td>create</td>
<td>&quot;invtype&quot; param value is not equal to label of one of Inventory Types, configured for Oracle Field Service Cloud</td>
</tr>
<tr>
<td>CODE_ACTION_MANDATORY_PARAM_EMPTY</td>
<td>install, deinstall, undo_install, undo_deinstall, create, delete</td>
<td>Any of these:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;invid&quot; param is not sent or its value is empty for &quot;install&quot;, &quot;deinstall&quot;, &quot;undo_install&quot;, &quot;undo_deinstall&quot; or &quot;delete&quot; action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;invpool&quot; param of &quot;create&quot; action is not sent or is empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;inv_aid&quot; param of &quot;install&quot; action is not sent or is empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;inv_pid&quot; param of &quot;deinstall&quot; action is not sent or is empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;inv_aid&quot; param of &quot;create&quot; action is not sent or is empty, and &quot;invpool&quot; is &quot;customer&quot;, &quot;install&quot; or &quot;deinstall&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;inv_pid&quot; param of &quot;create&quot; action is not sent or is empty, and &quot;invpool&quot; is &quot;provider&quot;, &quot;install&quot; or &quot;deinstall&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;quantity&quot; is not sent or is empty for inventory of non-serialized type</td>
</tr>
<tr>
<td>CODE_ACTION_PARAM_VALUE_INVALID</td>
<td>install, deinstall, undo_install, undo_deinstall, create</td>
<td>Any of these:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;properties&quot; param value is sent but is not a plain object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;quantity&quot; is sent for inventory of serialized type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* &quot;quantity&quot; is not a positive integer number</td>
</tr>
<tr>
<td>TYPE_ACTION_PROPERTY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_MANDATORY_PROPERTY_EMPTY</td>
<td>install, deinstall, undo_install, undo_deinstall, create</td>
<td>[Reserved]</td>
</tr>
<tr>
<td>CODE_ACTION_PROPERTY_VALUE_INVALID</td>
<td>install, deinstall, undo_install, undo_deinstall, create</td>
<td>Any of these:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Property type is 'file', its GUI type is 'signature' and its value is not a valid Data URI or it has the invalid MIME-type</td>
</tr>
<tr>
<td>Code Code</td>
<td>Caused by Action</td>
<td>Cause</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</td>
<td>-------</td>
</tr>
<tr>
<td>CODE_ACTIONPROPERTY_VALUE_TOO_LARGE</td>
<td>create</td>
<td>Property type is 'enumeration', and its value is not a valid enumeration item's index</td>
</tr>
<tr>
<td></td>
<td>install</td>
<td>Any of these: Property type is 'field' and length of its value exceeds 119 UTF-16 codepoints</td>
</tr>
<tr>
<td></td>
<td>deinstall</td>
<td>Property type is 'file', its GUI type is 'signature' and length of its value exceeds 102400 UTF-16 codepoints</td>
</tr>
<tr>
<td></td>
<td>undo_install</td>
<td>Property is neither field nor signature and length of its value exceeds 32767 UTF-16 codepoints</td>
</tr>
<tr>
<td></td>
<td>undo_deinstall</td>
<td></td>
</tr>
<tr>
<td></td>
<td>create</td>
<td></td>
</tr>
<tr>
<td>TYPE_ACTION_FAILED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_ACTIVITY_STATUS_INVALID</td>
<td>install</td>
<td>Any of these:</td>
</tr>
<tr>
<td></td>
<td>deinstall</td>
<td><em>inv_aid</em> param of &quot;install&quot;, &quot;deinstall&quot;, &quot;undo_install&quot; or &quot;undo_deinstall&quot; action is not equal to id of started activity</td>
</tr>
<tr>
<td></td>
<td>undo_install</td>
<td><em>inv_aid</em> param of &quot;create&quot; or &quot;delete&quot; action is not equal to id of started activity, and &quot;invpool&quot; is &quot;install&quot; or &quot;deinstall&quot;</td>
</tr>
<tr>
<td></td>
<td>undo_deinstall</td>
<td></td>
</tr>
<tr>
<td></td>
<td>create</td>
<td></td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_ACTIVITY_TYPE_INVALID</td>
<td>install</td>
<td></td>
</tr>
<tr>
<td></td>
<td>create</td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_ACTIVITY_UNKNOWN</td>
<td>install</td>
<td></td>
</tr>
<tr>
<td></td>
<td>create</td>
<td></td>
</tr>
<tr>
<td>CODE_ACTION_INVENTORY_POOL_TRANSITION_INVALID</td>
<td>install</td>
<td></td>
</tr>
<tr>
<td></td>
<td>deinstall</td>
<td></td>
</tr>
</tbody>
</table>
Oracle Field Service Cloud is unable to process the message due to unexpected change of the system’s state.

Oracle Field Service Cloud

The following figure describes the flow of steps when Oracle Field Service Mobility Cloud Service switches online and the application retries to connect to the Internet:
The steps that are executed when the application switches online are:

- Oracle Field Service Mobility Cloud Service switches to online mode.
- The iframe is created, if required.
- The Ready method is executed.
- The Wakeup method is executed; here, the event is ‘online’.
- The Sleep method is executed; here, the wakeupNeeded parameter is set to ‘true’.
- The iframe is destroyed, if it is created.

The steps that are executed when the application retries to connect to the Internet are:

- Oracle Field Service Mobility Cloud Service waits for five minutes.
- The iframe is created, if required.
- The Ready method is executed.
- The Wakeup method is executed; here, the event is ‘online’.
- The Sleep method is executed; here, the wakeupNeeded parameter is set to ‘false’.
- The iframe is destroyed, if it is created.

The following figure shows the flow of steps when the plug-in is unresponsive to the application’s requests:

The flow of steps when the plug-in doesn’t respond to the application’s requests at the time of switching online is:

- Oracle Field Service Mobility Cloud Service switches to online mode.
- The iframe is created, if required.
- The Ready method is executed.
- The Wakeup method is executed; here, the event is ‘online’.
- Oracle Field Service Mobility Cloud Service waits for two minutes.
- The iframe is destroyed, if it is created.

The following figure shows the flow of steps when the plug-in is opened by a user after the application switches to online mode:
The flow of steps when the plug-in is opened by a user after the application switches to online mode is:

- Oracle Field Service Mobility Cloud Service detects the active connection to the Internet.
- The iframe is created, if required.
- The Ready method is executed.
- The Wakeup method is executed.
- The user clicks the action link.
- The iframe is destroyed, if it is created.
- The iframe is created, if required.
- The Ready method is executed.
- The Open method is executed.
- The Close method is executed.
- The iframe is destroyed, if it is created.

The plug-in's iframe window is killed after the Close message is processed, regardless of whether the device is online or offline. So, no code runs after the plug-in is closed. It may have data that must be synchronized with its server, OFSC REST API, or with third-party services. So, the plug-in sends a message to Mobility when the data is synchronized and Mobility sends a message to the plug-in when it is online. If the plug-in supports offline mode, request Mobility to invoke it when the network connectivity is established.

A new parameter "wakeupNeeded" is added to the "close" message to perform the synchronization. If it's set to true, the hidden plug-in's iframe is opened in the background, as soon as Mobility goes online, but not earlier than 5 minutes (300 s)
after the plug-in is closed. See updated JSON Schema of the "close" message. After the plug-in's iframe is opened, Mobility sends the "wakeup" message to the plug-in in response to the "ready" message.

The plug-in must send the "sleep" message back to Mobility when it finishes synchronization, to allow the application to destroy the iframe. If the plug-in tried to synchronize, but couldn’t sent all the data, it must send the "sleep" message with the "wakeupNeeded" param set to true. In this case, Mobility opens the plug-in’s iframe in the background again, as soon as Mobility goes online, but no earlier than 5 minutes after the plug-in is closed. If the plug-in doesn’t send the "sleep" message in two minutes (120 s) after the "wakeup" message is sent, Mobility destroys its iframe and reopens it, as if the plug-in sent the "sleep" message with the "wakeupNeeded" param set to true.

To allow the plug-in to synchronize even after refreshing Mobility’s page or closing the browser, the new parameter "wakeupNeeded" is added to the "initEnd" message. If the plug-in didn’t synchronize during the two minutes that is allowed for initialization, it must send the "initEnd" message with the "wakeupNeeded" param set to true. In this case, Mobility opens the plug-in’s iframe in the background, as soon as Mobility goes online, but no earlier than 5 minutes after "initEnd" receiving the message.

The plug-in is opened in 5 minutes after it’s closed, if the "wakeupNeeded" param of "close", "sleep" or "initEnd" messages are set to true, even if Mobility didn’t detect offline when the plug-in was opened or closed. If the user opens the plug-in by clicking its action link in Mobility, the background iframe is destroyed without sending any messages to it. If plug-in still has data to be synchronized, it must send the "close" message with the "wakeupNeeded" param set to true.

**Property Value Length Limits**

Limits are applied to the property values that are submitted by the plug-in through the plug-in API. If the length of a value exceeds the limit, an error is returned as part of the message, using the error method.

**Fields (property type is field):** Fields are encrypted by AES and stored in the TINYBLOB columns. AES output block is 16 bytes, so the ciphertext length is always divisible by 16 (L % 16 = 0). Additionally, AES requires one extra block for the data whose length is divisible by 16. So, the maximum plain data to store is ceil(255/16)*16 - 1 = 239. JavaScript uses UTF-16 for strings, so one Unicode character may take up 2 to 4 bytes. But the String.length property uses UTF-16 code points for counting, which is 2 bytes. Therefore, the length of the string containing one 4-byte UTF-16 char will be 2. So, only ceil(239/2) = 119 code points can be stored without truncating.

**Signatures (property type is file and GUI element is Signature):** We assume that the value contains only MIME-type and correct base64 string, so each character takes up 2 bytes as JavaScript uses UTF-16. To avoid overflow of the LocalStorage we limit each signature to 200 KB (1024*200/2 = 102400 characters).

**Properties (any other property type):** Properties are stored in TEXT columns. The maximum amount of data to store is 65 535 bytes (2^16 - 1). JavaScript uses UTF-16 for strings, so one character may take up 2 to 4 bytes. But the String.length property uses UTF-16 code points for counting, which is 2 bytes. Therefore, length of the string containing one 4-byte UTF-16 char will be 2. So, in the worst case scenario, only ceil(65535/2) = 32767 code points can be stored without truncating.

**JSON Schema for Message Data**

```json
{  "type": "object",
    "properties": {
      "type": "string",
      "minLength": 0,
      "maxLength": 65535
    },
    "required": ["type"]
}
```
"properties": {
  "apiVersion": {
    "type": "number",
    "enum": [1]
  },
  "method": {
    "type": "string",
    "enum": ["ready"]
  },
  "sendInitData": {
    "type": "boolean"
  },
  "showHeader": {
    "type": "boolean"
  },
  "enableBackButton": {
    "type": "boolean"
  },
  "dataItems": {
    "type": "array",
    "items": {
      "type": "string",
      "enum": [
        "resource",
        "scheduledActivities",
        "nonScheduledActivities",
        "resourceInventories",
        "installedInventories",
        "deinstalledInventories",
        "customerInventories"
      ]
    }
  }
},
"required": ["apiVersion", "method"]
}

**init Method**

{
  "type": "object",
  "properties": {
    "apiVersion": {
      "type": "number",
      "enum": [1]
    },
    "method": {
      "type": "string",
      "enum": ["init"]
    }
  },
  "required": ["apiVersion", "method"]
}

**initEnd Method**

{
  "type": "object",
  "properties": {
    "apiVersion": {
      "type": "number",
      "enum": [1]
    }
  }
}
open Method

```
{
    "apiVersion": {"type": "number", "enum": ["1"]},
    "method": {"type": "string", "enum": ["open"]},
    "entity": {"type": "string", "enum": ["activity", "activityList", "activityInventory", "activityInventoryList", "inventory", "inventoryList"]},
    "resource": {
        "type": "object",
        "properties": {
            "pid": {"type": "number"},
            " pname": {"type": "string"},
            "external_id": {"type": "string"},
            "ptype": {"type": "string"},
            "email": {"type": "string"},
            " pphone": {"type": "string"},
            "pcapacity_bucket": {"type": "number"}
        },
        "patternProperties": {
            "^\+$": {"anyOf": [
                {"type": "string"},
                {"type": "number"}
            ],
            "description": "Key: Any property of an OFSC Activity; Value: value of this property"}
        }
    }
}
```

"length": {"type": "number"},
"ETA": {"type": "string"},
"status": {"type": "string"},
"aid": {"type": "number"},
"workzone": {"type": "number"},
"end_time": {"type": "string"},
"delivery_window": {"type": "string"},
"acoord_status": {"type": "string"},
"acoord_x": {"type": "number"},
"acoord_y": {"type": "number"},
"travel": {"type": "number"},
"sla_window_start": {"type": "string"},
"sla_window_end": {"type": "string"},
"apoints": {"type": "number"},
"activity_capacity_categories": {"type": "string"},
"atime_of_booking": {"type": "string"},
"atime_of_assignment": {"type": "string"},
"auto_routed_to_provider_id": {"type": "number"},
"auto_routed_to_date": {"type": "string"},
"auto_routed_to_provider_name": {"type": "string"},
"first_manual_operation": {"type": "number"},
"first_manual_operation_interface": {"type": "number"},
"first_manual_operation_user_id": {"type": "number"},
"first_manual_operation_user_login": {"type": "string"},
"first_manual_operation_user_name": {"type": "string"}
},
"patternProperties": {
"^\d+$": {
"anyOf": [
{"type": "string"},
{"type": "number"}
],
"description": "Key: Any property of an OFSC Activity; Value: value of this property"
}
},
"inventory": {
"type": "object",
"properties": {
"invid": {"type": "number"},
"invaid": {"type": "number"},
"invpid": {"type": "number"},
"invpool": {"type": "string"},
"invsn": {"type": "string"},
"invtype": {"type": "string"},
"quantity": {"type": "number"},
"inv_change_invid": {"type": "number"}
},
"patternProperties": {
"^\d+$": {
"anyOf": [
{"type": "string"},
{"type": "number"}
],
"description": "Key: Any property of an OFSC Activity; Value: value of this property"
}
},
"activityList": {
"type": "object",
"patternProperties": {
"^\d+$": {
"type": "object",
"properties": {
"cname": {"type": "string"},
"caddress": {"type": "string"},
"caddess": {"type": "string"},
"caddess": {"type": "string"}
"ccity": {
  "type": "string",
},
"czip": {
  "type": "number",
},
"cstate": {
  "type": "string",
},
"customer_number": {
  "type": "string",
},
"c_xid": {
  "type": "number",
},
"cphone": {
  "type": "string",
},
"cemail": {
  "type": "string",
},
"ccell": {
  "type": "string",
},
"atype": {
  "type": "string",
},
"aworktype": {
  "type": "string",
},
"time_slot": {
  "type": "number",
},
"service_window": {
  "type": "string",
},
"appt_number": {
  "type": "string",
},
"c_zid": {
  "type": "number",
},
"language": {
  "type": "number",
},
"cmessageftime": {
  "type": "number",
},
"activity_workskills": {
  "type": "string",
},
"length": {
  "type": "number",
},
"ETA": {
  "type": "string",
},
"astatus": {
  "type": "string",
},
"aid": {
  "type": "number",
},
"aworkzone": {
  "type": "number",
},
"end_time": {
  "type": "string",
},
"delivery_window": {
  "type": "string",
},
"acoord_status": {
  "type": "string",
},
"acoord_x": {
  "type": "number",
},
"acoord_y": {
  "type": "number",
},
"travel": {
  "type": "number",
},
"sla_window_start": {
  "type": "string",
},
"sladate": {
  "type": "string",
},
"apoints": {
  "type": "number",
},
"activity_capacity_categories": {
  "type": "string",
},
"atime_of_booking": {
  "type": "string",
},
"atime_of_assignment": {
  "type": "string",
},
"auto_routed_to_provider_id": {
  "type": "number",
},
"auto_routed_to_date": {
  "type": "string",
},
"auto_routed_to_provider_name": {
  "type": "string",
},
"first_manual_operation": {
  "type": "number",
},
"first_manual_operation_interface": {
  "type": "number",
},
"first_manual_operation_user_id": {
  "type": "number",
},
"first_manual_operation_user_login": {
  "type": "string",
},
"first_manual_operation_user_name": {
  "type": "string"
}.

"patternProperties": {
  "^\d+$": {
    "anyOf": [
      {
        "type": "string"
      },
      {
        "type": "number"
      }
    ],
    "description": "Key: Any property of an OFSC Activity; Value: value of this property"
  }
}

"inventoryList": {
  "type": "object",
  "patternProperties": {
    "//d+$": {
      "type": "object",
      "properties": {
        "invid": {
          "type": "number"
        },
        "inv_aid": {
          "type": "number"
        },
        "inv_pid": {
          "type": "number"
        },
        "invpool": {
          "type": "string"
        },
        "invsn": {
          "type": "string"
        },
        "invtype": {
          "type": "string"
        }
      }
    }
  }
}

"inventory": {
  "type": "string"
},
"inventory_id": {
  "type": "number"
},
"inventory_name": {
  "type": "string"
}.

"inventoryList": {
  "type": "object"
}.
"quantity": {"type": "number"},
"inv_change_invid": {"type": "number"}
},
"patternProperties": {
"^.+$": {
"anyOf": [
{"type": "string"},
{"type": "number"}
],
"description": "Key: Any property of an OFSC Activity; Value: value of this property"
}
}
,"required": ["apiVersion", "method", "entity", "resource"]
}

>Note: position_in_route, time_delivered, date, pid, atravelarea, activity_alerts, activity_compliance, eta_end_time fields are not supported and may not be added to the Activity properties for Mobility context layout.

close Method

{
"type": "object",
"properties": {
"apiVersion": {
"type": "number",
"enum": [1]
},
"method": {
"type": "string",
"enum": ["close"]
},
"backScreen": {
"type": "string",
},
"backActivityId": {"type": "string"},
"wakupNeeded": {
"type": "boolean"
},
"activity": {
"type": "object",
"required": ["aid"],
"properties": {
"cname": {
"type": "string"
},
"caddress": {
"type": "string"
},
"ccity": {
"type": "string"
},
"czip": {
"type": "number"
},
"cstate": {
"type": "string"
}
}
"type": "string",
"customer_number": {
  "type": "string"
},
"c_ssid": {
  "type": "number"
},
"cphone": {
  "type": "string"
},
"cemail": {
  "type": "string"
},
"ccell": {
  "type": "string"
},
"time_slot": {
  "type": "number"
},
"service_window": {
  "type": "string"
},
"appt_number": {
  "type": "string"
},
"clanguage": {
  "type": "number"
},
"cmessagetime": {
  "type": "number"
},
"length": {
  "type": "number"
},
"status": {
  "type": "string"
},
"aid": {
  "type": "string"
},
"sla_window_start": {
  "type": "string"
},
"sla_window_end": {
  "type": "string"
},
"apoints": {
  "type": "number"
},
"patternProperties": {
  "^\+\+$": {
    "anyOf": [
      {
        "type": "string"
      },
      {
        "type": "number"
      }
    ],
    "type": "object",
    "properties": {
      "fileName": {
        "type": "string"
      },
      "fileContents": {
        "type": "object"
      }
    }
  }
}
"required": ["fileName", "fileContents"]
  }]
},
"description": "Key: Any property of an OFSC Activity; Value: value of this property"
}
},
"inventory": {
  "type": "object",
  "required": ["invid"],
  "properties": {
    "invid": {
      "type": "string"
    },
    "inv_id": {
      "type": "string"
    },
    "inv_pid": {
      "type": "string"
    },
    "invpool": {
      "type": "string"
    },
    "invsn": {
      "type": "string"
    },
    "invtype": {
      "type": "string"
    },
    "quantity": {
      "type": "number"
    },
    "patternProperties": {
      "^\d+$": {
        "anyOf": [
          { "type": "string" },
          { "type": "number" },
          { "type": "object", "properties": { "fileName": { "type": "string" } } },
          { "type": "object", "properties": { "fileContents": { "type": "string" } } }
        ],
        "required": ["fileName", "fileContents"]
      }
    },
    "description": "Key: Any property of an OFSC Activity; Value: value of this property"
  }
},
"activityList": {
  "type": "object",
  "patternProperties": {
    "\d+": {
      "type": "object",
      "properties": {
        "cname": {
          "type": "string"
        },
        "caddress": {
          "type": "string"
        }
      }
    }
  },


```
}
"ccity": {"type": "string"},
"czip": {"type": "number"},
"cstate": {"type": "string"},
"customer_number": {"type": "string"},
"c_zid": {"type": "number"},
cphone": {"type": "string"},
cemail": {"type": "string"},
ccell": {"type": "string"},
time_slot": {"type": "number"},
service_window": {"type": "string"},
appt_number": {"type": "string"},
clanguage": {"type": "number"},
cmessage_time": {"type": "number"},
length": {"type": "number"},
astatus": {"type": "string"},
"aid": {"type": "string"},
"sla_window_start": {"type": "string"},
"sla_window_end": {"type": "string"},
apoints": {"type": "number"},
patternProperties": {
"^.*$": {
"anyOf": [{"type": "string"}, {"type": "number"}, {"type": "object"}],
"type": "object"}
```


"properties": {
  "fileName": {
    "type": "string"
  },
  "fileContents": {
    "type": "object"
  },
  "required": ["fileName", "fileContents"]
},
"description": "Key: Any property of an OFSC Activity; Value: value of this property"
},
"required": ["aid"]
},
"inventoryList": {
  "type": "object",
  "patternProperties": {
    "\d+$": {
      "type": "object",
      "required": false,
      "properties": {
        "invid": {
          "type": "string"
        },
        "inv_aid": {
          "type": "string"
        },
        "inv_pid": {
          "type": "string"
        },
        "invpool": {
          "type": "string"
        },
        "invsn": {
          "type": "string"
        },
        "invtype": {
          "type": "string"
        },
        "quantity": {
          "type": "number"
        }
      },
      "patternProperties": {
        "^\+$": {
          "anyOf": [
            { "type": "string" },
            { "type": "number" },
            { "type": "object",
              "properties": {
                "fileName": {
                  "type": "string"
                },
                "fileContents": {
                  "type": "object"
                }
              },
              "required": ["fileName", "fileContents"]
            }
          ],
          "description": "Key: Any property of an OFSC Activity; Value: value of this property"
        }
      }
    }
  }
},
"description": "Key: Any property of an OFSC Activity; Value: value of this property"


```json
{
  "actions": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "entity": {
          "type": "string"
        },
        "action": {
          "type": "string"
        },
        "invtype": {
          "type": "string"
        },
        "invpool": {
          "type": "string"
        },
        "invid": {
          "type": "string"
        },
        "inv_aid": {
          "type": "string"
        },
        "inv_pid": {
          "type": "string"
        },
        "quantity": {
          "type": "number"
        },
        "properties": {
          "type": "object",
          "properties": {
            "fileName": {
              "type": "string"
            },
            "fileContents": {
              "type": "object"
            }
          },
          "required": ["fileName", "fileContents"]
        }
      },
      "required": ["entity", "action"]
    },
    "required": ["apiVersion", "method"]
  }
}
```
wakeup Method
{
    "type": "object",
    "properties": {
        "apiVersion": {
            "type": "number",
            "enum": [1]
        },
        "method": {
            "type": "string",
            "enum": ["wakup"]
        },
        "event": {
            "type": "string",
            "enum": ["online"]
        }
    },
    "required": ["apiVersion", "method"]
}

sleep Method
{
    "type": "object",
    "properties": {
        "apiVersion": {
            "type": "number",
            "enum": [1]
        },
        "method": {
            "type": "string",
            "enum": ["sleep"]
        },
        "wakeupNeeded": {
            "type": "boolean"
        }
    },
    "required": ["apiVersion", "method"]
}

error Method
{
    "apiVersion": {"type": "number", "enum": ["1"]},
    "method": {"type": "string", "enum": ["error"]},
    "entity": {
        "type": "string",
        "enum": ["activity", "activityList", "activityInventory", "activityInventoryList", "inventory", "inventoryList"]
    },
    "errors": {
        "type": "object",
        "fields": {
            "type": {"type": "string", "enum": ["TYPE_ENTITY_ACTION", "TYPE_ENTITY_PROPERTY", "TYPE_INTERNAL"]},
            "code": {
                "type": "string",
                "enum": [
                    "CODE_ACTION_ON_PAST_DATE_NOT_ALLOWED",
                    "CODE_PROPERTY_ON_PAST_DATE_NOT_ALLOWED"
                ]
            }
        }
    }
}
"CODE_PROPERTY_VALUE_TOO_LARGE",
"CODE_MANDATORY_PROPERTY_EMPTY",
"CODE_ACTIVITY_STATUS_INVALID",
"CODE_INVENTORY_POOL_INVALID",
"CODE_INVENTORY_AID_INVALID",
"CODE_INVENTORY_PID_INVALID",
"CODE_ACTIVITY_STATUS_INVALID_FOR_FUTURE",
"CODE_ACTIVITY_STATUS_STARTED_ALREADY_IN_QUEUE",
"CODE_ACTIVITY_STATUS_INVALID_FOR_INACTIVE_QUEUE",
"CODE_UNKNOWN"
]
},
"entity": {"type": "string", "enum": ["activity", "inventory"]},
"entityId": {"type": "string"},
"propertyLabel": {"type": "string"}
},
"required": ["type", "code"]
},
"required": ["apiVersion", "method"]

JSON Example

open Method

```json
{
  "apiVersion": 1,
  "method": "open",
  "entity": "activity",
  "resource": {
    "pid": 5000038,
    "pname": "RAYNER, Faye",
    "external_id": "55038",
    "gender": "1"
  },
  "activity": {
    "WO_COMMENTS": "AUTOMATIC TRANSFER WORK ORDER/n/n",
    "astatus": "started",
    "aid": 3956534
  },
  "inventories": {
    "20997919": {
      "invid": 20997919,
      "inv_aid": 3956534,
      "inv_pid": 5000038,
      "invpool": "install",
      "invsn": "SABDFWKNZ"
    },
    "20998078": {
      "invid": 20998078,
      "inv_aid": 3956532,
      "invpool": "customer",
      "invsn": "5CTBME4AW090379"
    },
    "20998080": {
      "invid": 20998080,
      "inv_aid": 3956533,
      "invpool": "customer",
      "invsn": "SABGZTWGM"
    }
  }
}
```
close Method

```json
{
  "apiVersion": 1,
  "method": "close",
  "backScreen": "default",
  "actions": [
    // INSTALL
    {
      "entity": "inventory",
      "action": "install",
      "invid": 21258426,
      "inv_aid": 4224031,
      "properties": // Properties can be updated too
      {
        "PORT_INFO": "A0|1|1|0|7.9|QF9-0719537",
        "EQUIPMENT_ETHERNET": "08:00:27:ea:d1:bd",
      }
    },
    {
      "entity": "inventory",
      "action": "install",
      "invid": 21229417,
      "inv_aid": 4224031,
      "quantity": 12, // Install only 12 pieces of NSI
      "properties": // Model should be set if needed
      {
        "inventory_model": "RG6 - BLK",
      }
    },
    // DEINSTALL
    {
      "entity": "inventory",
      "action": "deinstall",
      "invid": 21064418,
      "inv_pid": 3000001
    },
    // CREATE
    {
      "entity": "inventory",
      "action": "create",
      "invtype": "NT",
      "invpool": "installed",
      "inv_aid": 4224031,
      "inv_pid": 3000001,
      "quantity": 100,
      "properties":
      {
        "inventory_model": "RG6 - BLK",
      }
    },
    // DELETE
    {
      "entity": "inventory",
      "action": "delete",
      "invid": "1484311067004-9891"
    }
  ]
}
```
Barcode Scanner Method

The plug-in framework supports remote procedure calls from Oracle Field Service Mobility Cloud Service Barcode Scanner Service. You can use this method and create a plug-in to scan barcodes. The Barcode Scanner button is available in the Inventory search screen, if you open Mobility through the native application. If you open the inventory list through a browser, the button is not available. The plug-in framework in release 18A is backward compatible with the version introduced in release 17.8.

If you use this method to create a relevant plug-in, your Android device turns into a scanner, so you need not use external tools such as custom Android keyboards. You can use the barcode Plug-in API to:

- Find inventory in the pools and show where the inventory is currently.
- Search an activity related to a specific inventory, in other words, scan inventory to search activities related to this inventory.
- Show details of the inventory searched (Name, Label, etc.).
- Depending on the pool where the inventory is available, suggest actions such as install, undo deinstall, undo exchange to the technician.
- Allow entering and updating a barcode number through a keyboard.

Syntax

`scanBarcode`

Parts and equipment usually have barcodes printed on their package or on the device. This procedure provides barcode and 2D (for example, QR, DATAMATRIX) code scanner functionality to make the search easier for the required items in the inventory pools. When this procedure is called, the scanner window is opened, which shows the live camera picture. When the barcode is recognized, the scanner windows are closed, and the resulting values are sent to the plug-in through the callProcedureResult method. If the barcode scanner is unavailable or Oracle Field Service Mobility Cloud Service is not run inside the Mobile app, the corresponding error code is returned to the plug-in through an error message. Having Oracle Field Service Cloud Mobile (for Android or iOS) is a prerequisite to use the Barcode Scanner Mobile Plug-in Framework to search by barcode.

Supported barcode and 2D code types

<table>
<thead>
<tr>
<th>Barcode Type</th>
<th>Android</th>
<th>iOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR_CODE</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DATA_MATRIX</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UPC_A</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UPC_E</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EAN_8</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EAN_13</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CODE_39</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
openLink

This procedure provides a common way to open external URLs either with Oracle Field Service Mobility Cloud Service run in the web browser or in Mobile app. When Oracle Field Service Cloud Mobile is used, the URL opens externally in a web browser. If not, it opens as a new browser tab.

callProcedure

```json
{
    "apiVersion": 1,
    "method": "callProcedure",
    "procedure": "scanBarcode",
    "callId": "123abc"
}
{
    "apiVersion": 1,
    "method": "callProcedure",
    "procedure": "openLink",
    "callId": "123abc",
    "params": {
    }
}
```

callProcedure Method Parameters:

<table>
<thead>
<tr>
<th>Param Name</th>
<th>Mandatory</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>Yes</td>
<td>Integer</td>
<td>API version</td>
</tr>
<tr>
<td>method</td>
<td>Yes</td>
<td>String</td>
<td>Must equal &quot;callProcedure&quot;</td>
</tr>
<tr>
<td>procedure</td>
<td>Yes</td>
<td>String</td>
<td>Procedure name</td>
</tr>
<tr>
<td>callId</td>
<td>Yes</td>
<td>String</td>
<td>Unique string identifier which is used to apply procedure response within plug-in</td>
</tr>
</tbody>
</table>

The current version of the method supports the scanBarcode and openLink values for the procedure parameter.

Example of callID generation

```javascript
function generateCallId() {
    return btoa(String.fromCharCode.apply(null, window.crypto.getRandomValues(new Uint8Array(16))));
}
```

callProcedureResult method

Oracle Field Service Mobility Cloud Service sends the message with this method to the plug-in after the execution of a procedure initiated by the callProcedure method. If the procedure fails, the error message is sent instead. callProcedure method parameters:
scanBarcode procedure: For this procedure, the resultData param of callProcedureResult message is an object, which contains the following fields:

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>String</td>
<td>API version</td>
</tr>
<tr>
<td>Format</td>
<td>String</td>
<td>Type of recognized barcode. See Supported barcode and 2D code types.</td>
</tr>
<tr>
<td>Cancelled</td>
<td>String</td>
<td>Equals true, if user closes the scanner window before the code is recognized.</td>
</tr>
</tbody>
</table>

When the method is successfully executed:

```
{
  "apiVersion": 1,
  "method": "callProcedureResult",
  "callid": "123abc"
  "resultData": {
    "text": "PT9012308",
    "format": "QR_CODE",
    "cancelled": false
  }
}
```

When the method is canceled:

```
{
  "apiVersion": 1,
  "method": "callProcedureResult",
  "callid": "123abc"
  "resultData": {
    "text": "PT9012308",
    "format": "QR_CODE",
    "cancelled": true
  }
}
```

Errors

Example

```
{
  code: "CODE_PROCEDURE_UNKNOWN",
  procedure: "scanBarcode"
}
```

Type of errors

<table>
<thead>
<tr>
<th>Type</th>
<th>Occurs When...</th>
<th>Available Message Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPEPROCEDURE_ERROR</td>
<td>Procedure call not valid due to missed params, and procedure executed with error.</td>
<td>Procedure [Name of the procedure where the error occurred.]</td>
</tr>
</tbody>
</table>
### Error codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Caused by Procedure</th>
<th>Error Type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_PROCEDURE_ERROR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE_CALL_ID_EMPTY</td>
<td>scanBarcode</td>
<td>Validation</td>
<td>Empty callId param.</td>
</tr>
<tr>
<td>CODE_CALL_ID_INVALID</td>
<td>scanBarcode</td>
<td>Validation</td>
<td>Invalid callId param.</td>
</tr>
<tr>
<td>CODE_CALL_ID_DUPLICATE</td>
<td>scanBarcode</td>
<td>Validation</td>
<td>Duplicate callId param.</td>
</tr>
<tr>
<td>CODE_PROCEDURE_FAILED</td>
<td>scanBarcode</td>
<td>Execution</td>
<td>Procedure execution failed to various reasons.</td>
</tr>
<tr>
<td>CODE_PROCEDURE_UNKNOWN</td>
<td>scanBarcode</td>
<td>Execution</td>
<td>Procedure was called with unknown procedure name.</td>
</tr>
<tr>
<td>CODE_PROCEDURE_UNAVAILABLE</td>
<td>scanBarcode</td>
<td>Internal</td>
<td>Oracle Field Service Mobility Cloud Service related to procedure is not available.</td>
</tr>
<tr>
<td>CODE_PROCEDURE_ACCEPTS_NO_P</td>
<td>scanBarcode</td>
<td>Validation</td>
<td>Procedure was called with params.</td>
</tr>
<tr>
<td>TYPE_PROCEDURE_PARAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CODE_PROCEDURE_MANDATORY_P</td>
<td>openLink</td>
<td>Validation</td>
<td>Mandatory param is missed.</td>
</tr>
<tr>
<td>CODE_PROCEDURE_PARAM_VALUE_</td>
<td>openLink</td>
<td>Validation</td>
<td>Param value is not valid.</td>
</tr>
</tbody>
</table>

### A Sample Plug-In

The sample plug-in includes the following files:

- index.html
- styles.css
- plugin.js
- signature.js
- cache.manifest
- logo.svg

```html
<!DOCTYPE html>
<html manifest="cache.manifest">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Sample Plugin</title>
```
The Plug-In Framework

```html
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>
<br/>```
Entity

Entity data items

Ask OFSC to send only selected items on next open

Resource information

Scheduled activity list

Non-scheduled activity list

Inventories of resource

Installed inventories

Inventories of customer
<div class="section section--visible">
<h2>Background synchronization</h2>
</div>

```html
<style.css
/*
Copyright (c) 2013, Yahoo! Inc. All rights reserved.
Code licensed under the BSD License:
http://yuilibrary.com/license/
version: 3.18.1
*/
html{
    color:#000;
    background:#FFF;
}
body,
div,
dl,
dt,
dd,
ul,
o1,
l1,
h1,
h2,
h3,
h4,
h5,
h6,
pre,
code,
form,
fieldset,
legend,
input,
textarea,
```
Chapter 2
The Plug-In Framework
Chapter 2

The Plug-In Framework
font-size: 100%;
white-space: pre;
vertical-align: top;
margin: 4px 0;
}

select.value {
font-size: 16px;
height: 19px;
-webkit-appearance: none;
-moz-appearance: none;
appearance: none;
color: #444;
padding: 0;
border-radius: 0;
vertical-align: baseline;
}

.writable {
border-bottom: 1px dashed black;
min-width: 60px;
}

.button {
display: inline-block;
padding: 7px 11px;
border: 1px solid #B9B9B9;
border-radius: 5px;
cursor: pointer;
outline: none;
font-weight: bold;
font-size: 16px;
background: #F0F0F0;
margin-bottom: 10px;
margin-right: 7px;
}

.button__generate_sign {
margin: 0;
font-weight: 400;
padding: 2px 7px;
}

.submit {
background: #ee3526;
color: #fff;
border-color: #B50000;
}

.json {
padding: 10px;
white-space: pre;
font-family: monospace;
border-radius: 5px;
margin: 0 0 10px;
background: #353535;
color: white;
}

.json__request {
display: none;
}

.json__response {
background: none;
}

ORACLE
plugin.js

"use strict";

(function($) {
  window.Plugin = function(debugMode) {
    this.debugMode = debugMode || false;
    this._messageListener = null;
  };

  $.extend(window.Plugin.prototype, {
    /**
     * Dictionary of enums
     */
    dictionary: {
      astatus: {
        pending: {
          label: 'pending',
          translation: 'Pending',
          outs: ['started', 'cancelled'],
          color: '#FFDE00'
        },
        started: {
          label: 'started',
          translation: 'Started',
          outs: ['complete', 'suspended', 'notdone', 'cancelled'],
          color: '#A2DE61'
        },
        complete: {
          label: 'complete',
          translation: 'Completed',
          outs: [],
          color: '#79B6EB'
        },
        suspended: {
          label: 'suspended',
          translation: 'Suspended',
          outs: [],
          color: '#9FF'
        },
        notdone: {
          label: 'notdone',
          translation: 'Not done',
          outs: [],
          color: '#60CECE'
        },
        cancelled: {
          label: 'cancelled',
          translation: 'Cancelled',
          outs: [],
          color: '#80FF80'
        }
      }
    }
  });
});
invpool: {
  customer: {
    label: 'customer',
    translation: 'Customer',
    outs: ['deinstall'],
    color: '#04D330'
  },
  install: {
    label: 'install',
    translation: 'Installed',
    outs: ['provider'],
    color: '#00A6F0'
  },
  deinstall: {
    label: 'deinstall',
    translation: 'Deinstalled',
    outs: ['customer'],
    color: '#00F8E8'
  },
  provider: {
    label: 'provider',
    translation: 'Resource',
    outs: ['install'],
    color: '#FFE43B'
  }
},

/**
 * Which field shouldn't be editable
 * format:
 * parent: {
 *  key: true|false
 * }
 *
 */
renderReadOnlyFieldsByParent: {
  data: {
    apiVersion: true,
    method: true,
    entity: true
  },
  resource: {
    pid: true,
    pname: true,
    gender: true
  }
},

/**
 * Check for string is valid JSON
 *
 * @param {*} str - String that should be validated
 *
 * @returns {boolean}
 *
 * @private
 */
_isJson: function(str)
{
  try
  {
```javascript
JSON.parse(str);
} catch (e) {
 return false;
} return true;
},

/**
 * Return origin of URL (protocol + domain)
 * @param {String} url
 * @returns {String}
 */
_getOrigin: function(url)
{
 if (url != '') {
 if (url.indexOf('://') > -1) {
 return 'https://' + url.split('/')[2];
 } else {
 return 'https://' + url.split('/')[0];
 }
 return ''; }
},

/**
 * Return domain of URL
 * @param {String} url
 * @returns {String}
 */
_getDomain: function(url)
{
 if (url != '') {
 if (url.indexOf('://') > -1) {
 return url.split('/')[2];
 } else {
 return url.split('/')[0];
 }
 return ''; }
},

/**
 * Sends postMessage to document.referrer
 * @param {Object} data - Data that will be sent
 */
```
* @private
*/
_sendPostMessageData: function(data)
{
    if (document.referrer !== '')
    {
        this._log(window.location.host + ' -> ' + data.method + ' ' + this._getDomain(document.referrer),
            JSON.stringify(data, null, 4));

        parent.postMessage(JSON.stringify(data), this._getOrigin(document.referrer));
    }
},

/**
* Handles during receiving postMessage
*
* @param {MessageEvent} event - Javascript event
* @private
*/
_getPostMessageData: function(event)
{
    if (typeof event.data !== 'undefined')
    {
        if (this._isJson(event.data))
        {
            var data = JSON.parse(event.data);

            if (data.method)
            {
                this._log(window.location.host + ' <- ' + data.method + ' ' + this._getDomain(event.origin),
                    JSON.stringify(data, null, 4));

                switch (data.method)
                {
                    case 'open':
                        this.pluginOpen(data);
                        break;
                    case 'error':
                        data.errors = data.errors || {error: 'Unknown error'};
                        this._showError(data.errors);
                        break;
                    default:
                        alert('Unknown method');
                }
            }
            else
            {
                this._log(window.location.host + ' <- NO METHOD ' + this._getDomain(event.origin), null, null, true);
            }
        }
        else
        {
            this._log(window.location.host + ' <- NO JSON ' + this._getDomain(event.origin), null, null, true);
        }
    }
    else
    {
        this._log(window.location.host + ' <- NO DATA ' + this._getDomain(event.origin), null, null, true);
    }
},


/**
 * Show alert with error
 *
 * @param {Object} errorData - Object with errors
 *
 * @private
 */
 showError: function(errorData)
 {
 alert(JSON.stringify(errorData, null, 4));
 },

/**
 * Logs to console
 *
 * @param {String} title - Message that will be log
 * @param {String} [data] - Formatted data that will be collapsed
 * @param {String} [color] - Color in Hex format
 * @param {Boolean} [warning] - Is it warning message?
 *
 * @private
 */
 log: function(title, data, color, warning)
 {
 if (!this.debugMode)
 {
 return;
 }
 if (!color)
 {
 color = '#0066FF';
 }
 if (!!data)
 {
 console.log('[Plugin API] ' + data);
 console.groupEnd();
 }
 else
 {
 }
 },

/**
 * Business login on plugin init
 */
 pluginInit: function()
 {
 var data = {
 initTime: new Date().getTime() 
 }; 
 this._log(window.location.host + ' INIT. SET DATA TO LOCAL STORAGE', JSON.stringify(data, null, 4));
 localStorage.setItem('pluginInitData', JSON.stringify(data));
 },

/**
 * Business login on plugin open
 *
 * @param {Object} receivedData - JSON object that contain data from OFSC
 */
 pluginOpen: function(receivedData)
 {

this._log(window.location.host + ' OPEN. GET DATA FROM LOCAL STORAGE',
JSON.stringify(JSON.parse(localStorage.getItem('pluginInitData')), null, 4));
$('.json__local-storage').text(localStorage.getItem('pluginInitData'));
$('.section__local-storage').show();

$('.form').html(this.renderForm(receivedData));

$('.key').each(function(index, item)
{
  if ($(item).siblings('.value').has('.items').size() !== 0)
  {
    $(item).addClass('clickable');
  }
}).on('click', function()
{
  if ($(this).siblings('.value').has('.items').size() !== 0)
  {
    $(this).siblings('.value').toggle();
    $(this).toggleClass('collapsed');
  }
});

$('.button__generate_sign').on('click', function(e)
{
  var canvasElement = $('<canvas>').addClass('value').attr({height: 240, width: 320}).get(0);
  $(e.target).after(canvasElement);
drawSampleSignature(canvasElement);

  $(e.target).parents('.item').addClass('edited');
  this._updateResponseJSON();
  $(e.target).remove();
}).bind(this);

$('.value__item').on('input, change', function(e)
{
  //IE10+
  $(e.target).parents('.item').addClass('edited');
  this._updateResponseJSON();
}).bind(this);

$('.back_method_select, .back_activity_id').on('change', function(e)
{
  //IE10+
  this._updateResponseJSON();
}).bind(this);

$('.json__request').text(JSON.stringify(receivedData, null, 4));

$('.submit').click(function()
{
  var json_response = $('.json__response');
  if (json_response.is(':hidden') === true)
  {
    var form = this.parseForm($('.form'));
    this._sendPostMessageData(form.data);
  }
  else
  {
    if (this._isJson(json_response.text()))
    {
      var data = JSON.parse(json_response.text());
      this._sendPostMessageData(data);
    }
  }
});
alert('JSON parse error!');
}
$.bind(this));

$('.section__ofsc-data').show();
,
/**
 * Render JSON object to DOM
 * @param {Object} data - JSON object
 * @returns {jQuery}
 */
renderForm: function(data)
{
    return this.renderCollection('data', data, true);
},
/**
 * Render JSON object to follow HTML:
 * <div class="item">
 * <div class="key">{key}</div>
 * <div class="value">{value}</div>
 * </div>
 * <div class="item">
 * <div class="key">{key}</div>
 * <div class="value">
 * <div class="items">
 * <div class="item">
 * <div class="key">{key}</div>
 * <div class="value">{value}</div>
 * </div>
 * <div class="item">
 * <div class="key">{key}</div>
 * <div class="value">{value}</div>
 * </div>
 * ...
 * </div>
 * </div>
 * ...
 * ...
 * ...

 * @param {String} key - Collection name
 * @param {Object} items - Child items of collection
 * @param {Boolean} [isWritable] - Will render as writable?
 * @param {number} [level] - Level of recursion
 * @param {string} [parentKey] - parent Key
 * @returns {jQuery}
 */
renderCollection: function(key, items, isWritable, level, parentKey)
{
    var render_item = $('<div>').addClass('item');
    var render_key = $('<div>').addClass('key').text(key);
    var render_value = $('<div>').addClass('value value__collection');
    var render_items = $('<div>').addClass('items');
    isWritable = isWritable || false;
    level = level || 1;
    parentKey = parentKey || '';

    var newParentKey = key;
if (items)
{
  $.each(items, function(key, value)
  {
    if (value && typeof value === 'object')
    {
      render_items.append(this.renderCollection(key, value, isWritable, level + 1, newParentKey));
    }
    else
    {
      render_items.append(this.renderItem(key, value, isWritable, level + 1, newParentKey).get(0));
    }
  }.bind(this));
}

render_item.append(render_key)/*.append('<span>: </span>')*/;
render_value.append(render_items);
render_item.append($('<br>'));
render_item.append(render_value);
return render_item;
}

/**
 * Render key and value to follow HTML:
 *
 * <div class="item">
 *   <div class="key">{key}</div>
 *   <div class="value">{value}</div>
 * </div>
 *
 * @param {String} key - Key
 * @param {String} value - Value
 * @param {Boolean} [isWritable] - Will render as writable?
 * @param {number} [level] - Level of recursion
 * @param {string} [parentKey] - parent Key
 * @returns {jQuery}
 */
renderItem: function(key, value, isWritable, level, parentKey)
{
  var render_item = $('<div>').addClass('item');
  var render_value;
  var render_key;
  isWritable = isWritable || false;
  level = level || 1;
  parentKey = parentKey || '';
  render_key = $('<div>').addClass('key').text(key);
  render_item.append(render_key).append('<span class="delimiter">: </span>');
  if (value === null)
  {
    value = '';
  }
  if (typeof this.renderReadOnlyFieldsByParent[parentKey] !== 'undefined' &&
  typeof this.renderReadOnlyFieldsByParent[parentKey][key] !== 'undefined' &&
  this.renderReadOnlyFieldsByParent[parentKey][key] === true)
  {
    isWritable = false;
  }

  if (value === null)
  {
    value = '';
  }

  if (typeof this.renderReadOnlyFieldsByParent[parentKey] !== 'undefined' &&
  typeof this.renderReadOnlyFieldsByParent[parentKey][key] !== 'undefined' &&
  this.renderReadOnlyFieldsByParent[parentKey][key] === true)
  {
    isWritable = false;
  }
switch (key)
{
    case "csign":
    if (isWritable)
    {
        render_value = $('button').addClass('button button__generate_sign').text('Generate');
    }
    break;
    default:
    if (this.dictionary[key])
    {
        render_value = this.renderSelect(this.dictionary, key, value, isWritable).addClass('value value__item');
    }
    else
    {
        render_value = $('div').addClass('value value__item').text(value);
        if (isWritable)
        {
            render_value.addClass('writable').attr('contenteditable', true);
        }
    }
    break;
}
render_item.append(render_value);
return render_item;
},

/**
 * Render enums
 *
 * <select class="value [writable]" [disabled]>
 * <option value="{value}" [selected]>{dictionary}</option>
 * ...
 * </select>
 *
 * @param {Object} dictionary - Dictionary that will be used for Enum rendering
 * @param {String} key - Just field name
 * @param {String} value - Selected value
 * @param {Boolean} isWritable - Will render as writable?
 * @returns {HTMLElement}
 */
renderSelect: function(dictionary, key, value, isWritable)
{
    var render_value;
    var outs = dictionary[key][value].outs;
    var allowedValues = [value].concat(outs);
    var disabled = '';
    render_value = $('select').css({background: dictionary[key][value].color});
    if (isWritable)
    {
        render_value.addClass('writable');
    }
    if (!outs.length || !isWritable)
    {
        render_value.attr('disabled', true);
        render_value.removeClass('writable');
    }
    $.each(allowedValues, function(index, label)
}
{ render_value.append('<option' + (label === value ? ' selected' : '') + ' value="' + dictionary[key][label].label + '">' + dictionary[key][label].translation + '</option>'); }

return render_value;
,
/**
 * Parse form (root HTML element that was rendered)
 * @param {HTMLElement} element - root HTML element
 * @returns {Object}
 */
parseForm: function(element)
{
    var form = {
        apiVersion: 1,
        method: 'close',
        backScreen: $('.back_method_select').val()
    };

    if (form.data.backScreen === 'activity_by_id')
    {
        $.extend(form.data, {
            backActivityId: $('.back_activity_id').val()
        });
    }

    $.extend(form.data, this.parseCollection(element).data);

delete form.data.entity;
delete form.data.resource;

return form;
},
/**
 * Convert HTML elements to JSON
 * @param {HTMLElement} rootElement - Root element that should be parsed recursively
 * @returns {Object}
 */

<div class="key">activity</div>
<div class="value value__collection">
    <div class="items">
        <div class="item edited">
            <div class="key">WO_COMMENTS</div>
            <div class="value">text_comments</div>
        </div>
        <div class="item">
            <div class="key">aid</div>
            <div class="value">4225274</div>
        </div>
        <div class="item">
            <div class="key">caddress</div>
            <div class="value">text_address</div>
        </div>
    </div>
</div>

converts to:
```
* 
* { 
*  "aid": "4225274",
*  "WO_COMMENTS": "text_comments"
* } 
* */

parseCollection: function(rootElement) {

  var returnObject = {};

  $(rootElement).children('.item').each(function(itemIndex, item) {
    var parentKey;
    var valueKey;
    var value;
    var mandatoryField = false;

    parentKey = $(rootElement).parent().siblings('.key').get(0);
    valueKey = $(item).children('.key').get(0);

    // Logic of mandatory fields
    if ((parentKey !== undefined) && ( $(parentKey).text() == 'activity' && $(valueKey).text() == 'aid') || $(parentKey).text() == 'inventory' && $(valueKey).text() == 'invid')) {
      mandatoryField = true;
    }

    if ($(item).hasClass('item') === true && ($(item).hasClass('edited') === true || mandatoryField)) {
      value = $(item).children('.value').get(0);
      if ($(value).children('.items').size() > 0) {
        returnObject[$(valueKey).text()] = this.parseCollection($(value).children('.items').get(0));
      } else {
        switch ($(value).prop("tagName")) {
          case 'SELECT':
            returnObject[$(valueKey).text()] = $(value).val();
            break;
          case 'CANVAS':
            returnObject[$(valueKey).text()] = value.toDataURL();
            break;
          default:
            returnObject[$(valueKey).text()] = $(value).text();
            break;
        }
      }
    }
  }).bind(this);

  return returnObject;
},

/** 
* Update JSON 
* 
* @private 
*/
```
_updateResponseJSON: function()
{
    var form = this.parseForm($('.form'));
    $('.json__response').text(JSON.stringify(form.data, null, 4));
},

/**
 * Initialization function 
 */
init: function()
{
    $('.back_method_select').change(function()
    {
        if ($('.back_method_select').val() == 'activity_by_id')
        {
            $('.back_activity_id').show();
        }
        else
        {
            $('.back_activity_id').val('').hide();
        }
    });

    $('.json_request_toggle').click(function()
    {
        $('.json__response').hide();
        $('.json__request').toggle();
    });

    $('.json_response_toggle').click(function()
    {
        $('.json__request').hide();
        this._updateResponseJSON();
        $('.json__response').toggle();
        }.bind(this));

    this._messageListener = this._getPostMessageData.bind(this);

    window.addEventListener("message", this._messageListener, false); //Only IE9+

    this.pluginInit();

    this._sendPostMessageData({
        apiVersion: 1,
        method: 'ready'
    });

})(jQuery);

signature.js

function drawSampleSignature(canvas)
{
    if (!canvas.getContext)
    {
        return;
    }
    var c = canvas.getContext('2d');

    c.fillStyle="ffffff";
    c.strokeStyle = "#000000";
    c.lineWidth = 1.5;
    c.lineCap = "round";
Oracle Field Service Cloud
Mobile Plug-in Framework
Chapter 2
The Plug-In Framework

```javascript
// t
```

```javascript
cache.manifest
CACHE:
./index.html
./style.css
./plugin.js
./signature.js
./logo.svg
//code.jquery.com/jquery-2.1.4.min.js

FALLBACK:

NETWORK:
*

logo.svg
```
<svg xmlns:svg="http://www.w3.org/2000/svg"
xmlns="http://www.w3.org/2000/svg" version="1.0" width="662.84644"
height="94.145668" id="svg115845">
<defs id="defs115847">
<clipPath id="clp82">
<path d="M 1001.6,870.49 L 1036.3,870.49 L 1036.3,857.41 L 1001.6,857.41 L 1001.6,870.49 z " id="path1826"/>
</clipPath>
<clipPath id="clp83">
<path d="M 1001.6,870.49 L 1036.3,870.49 L 1036.3,857.41 L 1001.6,857.41 L 1001.6,870.49 z " id="path1835"/>
</clipPath>
<clipPath id="clp84">
<path d="M 1001.6,870.49 L 1036.3,870.49 L 1036.3,857.41 L 1001.6,857.41 L 1001.6,870.49 z " id="path1844"/>
</clipPath>
<clipPath id="clp81">
<path d="M 1001.6,870.49 L 1036.3,870.49 L 1036.3,857.41 L 1001.6,857.41 L 1001.6,870.49 z " id="path1844"/>
</clipPath>
```
Upgrading from Previous Versions

Suppose you have created a plug-in API for a specific version of Oracle Field Service Cloud and Oracle Field Service Cloud is upgraded with the next version. You must upgrade your plug-in API to make it compatible with the latest release of Oracle Field Service Cloud.

You can upgrade your plug-ins to version 16.2 according to the following rules:

- Existing external plug-ins will continue functioning according to their settings; their type will change from External to HTML5 Application.
- You can edit the settings of the existing external plug-ins in version 16.2.

⚠️ CAUTION: Selecting the Use Plugin API check box for a plug-in created in a previous version makes the plug-in unusable.

- Existing internal plug-ins will continue functioning according to their settings.
- You cannot edit the settings of the existing internal plug-ins; they will be displayed as Read-Only.
- No new internal plug-ins can be created.

The plug-in API now accepts raw JS objects as values for PostMessage data. Original JSON strings are still supported, so backward compatibility with the existing plug-ins framework that were implemented in release 17.2 are retained. However, file properties cannot be updated in this case.

Accessing Oracle Field Service Cloud Public API

To avoid cross-domain communication blocking when an Oracle Field Service Cloud API is called from a plugin:

- For Oracle Field Service Cloud hosted plug-ins: When calling an Oracle Field Service Cloud API, the plug-in must use a plug-in hosting domain (available in Java script as a value of window.location.hostname property) instead of api.etadirect.com.
- For externally hosted plug-ins: The plug-in must not call an Oracle Field Service Cloud API directly from the browser. Instead, the plug-in must call its server side. All the API calls must be performed by the server side and the call results transmitted to the plug-in.
3 Hosting Plug-Ins

How to Host a Plug-In

If your plug-in consists only of HTML, CSS, and JavaScript files and doesn’t contain server-side files, then you can upload it in Oracle Field Service Cloud. No additional hosting is required. The plug-in framework handles the communication between the hosted plug-in and Oracle Field Service Cloud. You can host a maximum of 10 plug-ins per instance.

The steps to host a plug-in are:

• Prepare the plug-in for upload.
• Upload the plug-in.

After hosting a plug-in, you can:

• Use it in an action link
• Move between instances
• Modify
• Rollback to a previous version
• Delete

Note: A hosted plug-in works only with Oracle Field Service Mobility Cloud Service.

Preparing a Plug-In for Upload

The plug-in must be in a specific format to be uploaded. If not, you cannot upload it, you must host it elsewhere.

The plug-in files must meet the following requirements:

• You must upload a ZIP archive of the plug-in files.
• You can upload only the files of following types:
  o .html
  o .css
  o .js
  o .jpg
  o .jpeg
  o .png
  o .gif
  o .svg
  o appcache
You can organize files in sub-directories, but you must have the "index.html" file in the root folder.

- Each file can be a maximum of 1 MB and the total size of the compressed archive must be less than 500 KB.
- You can have a maximum of 10 files or directories in the archive.

Note: The plug-in files uploaded in Oracle Field Service Cloud are available by unique URLs on the Internet. The URLs are generated automatically and contain a long string. There is no authentication to access these files, so anyone who has the direct link to the file can download the file. Therefore, do not store any sensitive information such as passwords or login names in the plug-in archive. If you do not want your code to be available without authentication, we recommend that you do not use the hosted plug-in functionality. Be aware that the communication between the plug-in and Oracle Field Service Cloud starts only when a user successfully logs in to Oracle Field Service Cloud.

Working Offline

You can create the plug-in to work offline using the Application Cache functionality, which is specified in HTML5 specification (section "5.7 Offline Web Applications").

You can include the manifest file in the archive and associate your index.html file with it. For example, if the name of the manifest file is "manifest.appcache" then include the following code in the html file:

```html
<html manifest="manifest.appcache">
```

Do not add this line if your plug-in is intended to work only online. Each version of the uploaded plug-in is hosted in a separate directory. If you use the application cache functionality, the browser considers each new version of the plug-in as a separate offline application. Further, by default the browser doesn't clean up the application cache of the previous versions of the plug-in. So, do not create too many versions of the plug-ins on the production instance.

Tip: If you experience an overflow of the application cache in the browser, clean up the offline cache of the browser.

Upload a Plug-In

You must upload the plug-in archive to use it as a hosted plug-in.

Uploading a plug-in includes:

1. Creating the name of the plug-in.
2. Uploading the plug-in archive.

1. To create the name of the plug-in:
   a. Click Configuration > Action Management > Hosted Plugins.
      The Hosted Plugins screen appears, with the list of existing hosted plug-ins.
   b. Click the plus icon.
      The Add Hosted Plug-in dialog appears.
   c. Enter the name of the plug-in and click Add.
This name is displayed when you link the plug-in on the Action management screen. End-users do not see the plug-in name.

2. To upload the archive, click Browse on the Hosted Plugins screen and select the ZIP archive that is ready for upload.

The Version history section is populated with:
   ◦ The user name of the user that uploaded the files.
   ◦ The date on which the archive is uploaded.
   ◦ A link to download the archive.

To be able to use the plug-in, you must add it to an action link. See the Use a hosted plug-in in action link topic.

Modify, Download, or Delete an Archive

After uploading a plug-in archive, you might want to modify it, download it, or delete it.

1. To modify a hosted plug-in, you upload a newer version. To upload a newer version of the archive, click Browse on the Hosted Plugins screen and upload it again.

   Whenever you upload a newer version of a plug-in, the following happens:
      ◦ The current version becomes a historical one.
      ◦ The previous version is removed from Oracle Field Service Cloud.
      ◦ The newly uploaded version becomes the current one.

2. To delete a plug-in first unassign it from all the action links it is added to. Then, click the minus icon on the Hosted Plugins page.

   The plug-in is deleted with all its historical versions.

3. To download a plug-in, determine the version that you want to download in the Version history section. Click Download.

4. To rollback to a previous version, download the version that you want to rollback to. Click Browse and upload it again.

5. To move all the uploaded plug-ins between instances, export from the required instance using the Export function on the Action Management screen. Import the exported files using the Import function in the target instance.

6. To move a single plug-in between instances, download it from the required instance. Upload it in the target instance.

Use in an Action Link

To be able to use a hosted plug-in, you must first add it to an action link and then add the action link to a screen using Visual form editor. In other words, the name of the plug-in created in the Hosted plug-ins screen is added to an action link in the Add action link screen.

1. Click Configuration.

2. Click Action Management in the Displays section.

3. Click Add action link.

   The Add action link window opens.

4. Complete the following fields on the General tab:
### Oracle Field Service Cloud

**Chapter 3**

**Mobile Plug-in Framework**

**Hosting Plug-Ins**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action name (English)</td>
<td>A mandatory field defining the action or plug-in name in the English language. The action or plug-in appears under this name in the actual context.</td>
</tr>
<tr>
<td>Action name (other languages)</td>
<td>Action or plug-in name translations to other languages, if used.</td>
</tr>
<tr>
<td>Action label</td>
<td>A mandatory field defining a unique action or plug-in label.</td>
</tr>
<tr>
<td>Entity</td>
<td>Entity (activity, inventory, required inventory, resource, service request, user) to which the action or plug-in is to be related. For example, if you select Inventory, the action will appear only in the contexts related to inventory. Leave the field blank for the action to be available in all contexts of all the entities.</td>
</tr>
<tr>
<td>Interface</td>
<td>The application interface (Manage or Mobility) in which the action is to be used. Hosted plug-ins work only in the Mobility interface.</td>
</tr>
<tr>
<td>Base action</td>
<td>Base actions to be performed as a result of the custom action. The list of available base actions is filtered according to the Entity you have selected. For example, if you select required inventory for Entity, then Base action lists only the actions related to required inventory. The base action from which the plug-in is to be derived, if needed. When a base action is selected, the resulting plug-in functions per the same rules as the base action. It appears only in the contexts in which the base action appears and is shown or hidden according to the same visibility conditions. For example, if start_activity is selected as the base action for a plug-in, the plug-in is only shown in the context of a pending activity when there is no started activity in the same route, similar to the Start action.</td>
</tr>
<tr>
<td>Is Plugin</td>
<td>Whether the custom action is a plug-in. Select this check box. The Plug-in details tab is added to the Add/Edit action link window and the Custom params tab is no longer shown.</td>
</tr>
</tbody>
</table>

5. Complete the following fields in the **Plug-in detail** tab:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The plug-in type. Select HTML5 application. This means, the plug-in uses an external application to extend the functionality. An HTML5 application plug-in can be used in one of the following Mobility contexts - Activity List, Edit., View Activity, Inventory Grid or Add/Details Inventory.</td>
</tr>
<tr>
<td>Use Plug-in API</td>
<td>Determines if the plug-in works through a third-party Plug-in API or as a regular external plug-in. Select this check box.</td>
</tr>
<tr>
<td>Use hosted plugin</td>
<td>Defines whether you want to use a hosted plug-in. Select this check box.</td>
</tr>
<tr>
<td>Hosted plugin</td>
<td>The hosted plug-in that you want to use. Select the name from the drop-down menu. This field displays the names that you have created in the first step of the Upload a plug-in procedure.</td>
</tr>
<tr>
<td>Disable Action Link in Offline</td>
<td>Defines whether the plug-in is disabled in offline mode. Clear this check box if you want the plug-in to work in offline mode with Oracle Field Service Mobility Cloud Service.</td>
</tr>
<tr>
<td>Secure parameters</td>
<td>The secure information such as user name and password used to access a third-party service through HTTP. For example, a third-party map service. This data is encrypted and stored. Use the plus icon to add a new key-value pair. You can add a maximum of 20 key-value pairs, after which the icon is hidden. The maximum size of the parameters allowed is 5 KB. This size includes the data structure overhead and doesn’t correspond to the length of keys and values of strings. Changes to the secure data are sent to Oracle Field Service Mobility Cloud Service during the next synchronization. The data is sent to the plug-in when the next message is sent.</td>
</tr>
</tbody>
</table>

6. Click **OK** on the Add action link window.
7. To add the action link to a screen using Visual Form Editor:
   a. Click Configuration > User Types.
   b. Select the user type for which you want to add the action link and then click Screen configuration.
   c. Expand Mobility and click the screen to which you want to add the action link.
      The Visual Form Editor for that screen appears.
   d. Click Actions. Drag and drop the newly created action link to the required area of the screen.
   e. Click Save.
      The action link is displayed on the selected screen, the next time you log in to Oracle Field Service Mobility Cloud Service.
4 Using plug-ins

Configure the Application to Use a Plug-In

You create a plug-in as an HTML5 application and add it to the Edit/View activity context in the Visual Form Editor.

1. Configure the activity, inventory, and resource properties that you want to update through the plug-in in the Activity Properties, Resource Properties, and Resource Properties plug-in context layouts in Mobility Cloud Service. The following figure shows the Edit/View activity Visual Form Editor:

The plug-in action link appears as a button on the Activity details screen in Mobility Cloud Service. The following figure shows the plugin button on the Activity details screen:

2. Tap the plug-in button in Oracle Field Service Mobility Cloud Service. The plug-in window opens showing the activity, inventory, and resource properties configured in the plug-in API contexts. You can change the values of these fields from the plug-in window to update the corresponding properties in Oracle Field Service Cloud. The following figure shows a sample plug-in window and includes the activity, inventory, and resource properties:
The following figure shows the property values changed in plug-in window (left) and updated in Oracle Field Service Cloud (right):
Using the Plug-In

When you create an HTML5 application plug-in, you can use it in the Activity list, Edit/View activity, Inventory grid, Add/Details inventory Mobility Cloud Service contexts. This topic describes how to use a plug-in in Oracle Field Service Cloud.

Additionally, you can define the properties of the Oracle Field Service Cloud entities to be sent in the plug-in message. These properties are added to the special context layouts related to plug-ins. A separate context exists for activity, inventory and resource properties. The properties defined in the corresponding contexts will be sent in the plug-in API messages. The property visibilities are also checked and observed. The following figure shows the Screen configuration screen with the context layout options:
Initialization

At the time of initializing Oracle Field Service Cloud, the related URL requests are sent for all plug-ins through the plug-in API. Each URL opens in an invisible iframe, and waits for a ready message. Plug-ins can have one of the following statuses when loaded:

- The plug-in is successfully loaded and ready to work in the offline mode.
- The plug-in has not been loaded properly because no ready message has been sent within two minutes.

On initializing each Mobility Cloud Service instance, the application attempts to initialize the plug-ins through the plug-in API. This step is necessary, because Oracle Field Service Cloud must provide the possibility for the plug-in to download files into the application cache while the application is online. Initialization must prepare the plug-in to work offline. If the plug-in is not initialized, Oracle Field Service Cloud attempts to initialize it again when the plug-in starts. But if the plug-in is not initialized upon start of the application and the application goes offline, you cannot open the plug-in. An error message is displayed.

Initialization Errors

While initialization is in progress, the screen displays the message: The plugin has not loaded. After the plug-in is initialized successfully, the plug-in screen appears. If initialization fails, the message: The plugin has not loaded is displayed on the plug-in screen. The message The following plugin has not been loaded: [Plugin Name] is displayed on other Mobility Cloud Service screens. If the plug-in is not initialized when the application starts, tap the plug-in action link to force plug-in initialization. You will see the same message to let you know that the plug-in has not loaded yet and, if errors occur, the failed initialization message appears.

Possible Transitions

This topic describes the possible transitions of activity statuses and inventory pools in a plug-in API. Any status change that does not correspond to the flow results in an error.

The following chart demonstrates the flow of activity status changes:
The status always starts from pending; from pending it can transition to canceled, or pending to started, started to complete, or started to not done, and started to suspended.

>Note: The current version of the plug-in API does not support setting final statuses to multi-day activities or multi-day activity segments.

Status Changes of Inventory Pools
The following chart demonstrates the flow of inventory pool status changes. Any status change that does not correspond to the flow results in an error:

The possible transitions are resource to install, install to resource, customer to de-install, and de-install to customer.
# Revision History

This document will continue to evolve as existing sections change and new information is added.

<table>
<thead>
<tr>
<th>Date</th>
<th>What’s Changed</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2018</td>
<td>New topic added: Accessing Oracle Field Service Cloud Public API</td>
<td></td>
</tr>
<tr>
<td>February 2018</td>
<td>The following topics are added:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Storing and passing sensitive data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Barcode scanner method</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• How to host a plug-in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Preparing a plug-in for upload</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Working offline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Upload a plug-in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Manage uploaded archives</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Use in an action link</td>
<td></td>
</tr>
<tr>
<td>December 2017</td>
<td>Minor changes for clarity and consistency</td>
<td></td>
</tr>
<tr>
<td>August 2017</td>
<td>The following topics are updated:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Upgrading from previous versions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The following topics are added:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hiding the header when the plug-in is open</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Preventing the user from using Back</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Example for ready method</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Returning from the plug-in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Example of “close” method with different “backScreen” cases</td>
<td></td>
</tr>
<tr>
<td>June 2017</td>
<td>All the topics are updated to make the content more accurate. The order of</td>
<td>All the topics are updated to make the content more accurate. The</td>
</tr>
<tr>
<td></td>
<td>topics is changed to provide a logical flow to the guide.</td>
<td>order of topics is changed to provide a logical flow to the guide.</td>
</tr>
<tr>
<td>May 2017</td>
<td>The following topics are updated:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Basics of the plug-in framework</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Basic HTTP authentication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Code examples</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Available methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• JSON schema for message data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• JSON example</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Error handling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Error codes for actions topic is added</td>
<td></td>
</tr>
<tr>
<td>April 2017</td>
<td>Initial release</td>
<td></td>
</tr>
</tbody>
</table>