Oracle Field Service Cloud
Integrating with History API
# Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>i</th>
</tr>
</thead>
</table>

## 1 Introduction

| Document Purpose | 1 |
| Scope of the Document | 1 |
| Target Audience | 1 |
| Accessing the APIs | 1 |
| Glossary | 1 |

## 2 History API Introduction

| History API Introduction | 5 |

## 3 Benefits of Using History API

| Benefits of Using History API | 7 |

## 4 History API Workflow

| History API Workflow | 9 |

## 5 Authentication

| History API Authentication Methods | 13 |

## 6 History API Access

| History API Access | 15 |

## 7 History API Operation Description

| Objects Monitored by History | 17 |
| History API Request | 17 |
| History API Response | 18 |
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 Introduction

Document Purpose

The document is intended to provide description of the History API used to retrieve the history of operations performed in Oracle Field Service Cloud (referred to further as OFSC). The data retrieved by the History API can be further used by external applications.

Scope of the Document

This document describes the History API request used to retrieve the history records and responses received for such request.

Target Audience

The document is intended for developers and programmers working with the OFSC history in order to integrate OFSC with external systems.

Accessing the APIs

To access the Oracle Field Service Cloud APIs, you must use the https://api.etadirect.com URL scheme. All old URL schemes such as, companyname.etadirect.com, na.etadirect.com, eu.etadirect.com, and so on are deprecated for Oracle Field Service Cloud versions 15.8 and later.

For example, if you are using https://companyname.etadirect.com/soap/inbound/?wsdl to access the Inbound WSDL API, the URL per the new scheme is https://api.etadirect.com/soap/inbound/?wsdl.

Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>Any time-consuming work performed by a resource</td>
</tr>
<tr>
<td>Term</td>
<td>Explanation</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Bucket</td>
<td>Entity appearing on the resource tree which can contain resources of a defined type and be assigned activities</td>
</tr>
<tr>
<td>Capacity</td>
<td>Workforce possessing the necessary work skills available at a certain moment of time</td>
</tr>
<tr>
<td>Capacity bucket</td>
<td>Bucket used for Quota management</td>
</tr>
<tr>
<td>Context</td>
<td>OFSC or API screen showing all available properties and action links</td>
</tr>
<tr>
<td>Delivery window</td>
<td>Statistically calculated time period in which a resource is expected to start an activity</td>
</tr>
<tr>
<td>ETA</td>
<td>Estimated time of arrival. Predicted time at which a resource will arrive at an appointment and start an activity, calculated dynamically for current and historical data</td>
</tr>
<tr>
<td>Field</td>
<td>Property present in the system by default</td>
</tr>
<tr>
<td>Group</td>
<td>Feature on the resource tree identifying a particular type of resource</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface, allows to use software by manipulating images rather than by issuing text commands</td>
</tr>
<tr>
<td>Inventory</td>
<td>Equipment that can be installed or deinstalled during an activity</td>
</tr>
<tr>
<td>Linked activities</td>
<td>Two separate activities related so that the completion or start of one is dependent on the completion or start of the other</td>
</tr>
<tr>
<td>Manage</td>
<td>Oracle Field Service Core Manage Cloud Service (former Manage Application). Serves as the command center for field operations and the central hub for viewing real-time information about resources and their activities across the entire field organization</td>
</tr>
<tr>
<td>Mass activity</td>
<td>Activity involving 2 or more resources</td>
</tr>
<tr>
<td>Mobility</td>
<td>Oracle Field Service Mobility Cloud Service (former Mobility Application). A web-based application for mobile resources to execute and manage work-related activities and ongoing communications</td>
</tr>
<tr>
<td>Non-scheduled</td>
<td>Activity not assigned to a specific date</td>
</tr>
<tr>
<td>Not-ordered</td>
<td>Activity with an unspecified order of execution in a route, so that it can be executed at any time during the working day. Not-ordered activities do not have defined ETAs or delivery windows</td>
</tr>
<tr>
<td>Ordered</td>
<td>Activity with a defined place in a route, which must be performed at a specified time of day. The order of activities can be changed; ordered activities can be changed to not-ordered activities and vice-versa</td>
</tr>
<tr>
<td>Preassigned activity</td>
<td>Activity which was assigned to a specific resource before the routing run</td>
</tr>
<tr>
<td>Term</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Property</td>
<td>Field and field value, assigned to an entity in OFSC (to user, resource, activity or inventory). There are fields and company-defined properties.</td>
</tr>
<tr>
<td>Repeating activity</td>
<td>Activity recurring with a predefined frequency in a predefined period</td>
</tr>
<tr>
<td>Resource</td>
<td>Element in the resource tree representing a defined company asset</td>
</tr>
<tr>
<td>Resource Tree</td>
<td>Hierarchy of company resources, showing “parent-child” relationships</td>
</tr>
<tr>
<td>Route</td>
<td>List of activities assigned to a resource for a specific date, or a list of non-scheduled activities assigned to a resource</td>
</tr>
<tr>
<td>Routing</td>
<td>Process of assigning activities to resources (usually automated)</td>
</tr>
<tr>
<td>Service Window</td>
<td>Time frame expected by the customer for an activity as scheduled by the company</td>
</tr>
<tr>
<td>SLA window</td>
<td>Interval of time (that may involve a range of dates) within which certain work has to be performed according to the Service Level Agreement</td>
</tr>
<tr>
<td>Teamwork</td>
<td>Feature that allows resources to assist each other in an activity or on an on-going basis</td>
</tr>
</tbody>
</table>
| Time Slot       | 1) Fixed service window, defined with a name and label, specifying when certain types of activities can be performed  
|                 | 2) Service Window (if the activity type does not support time slots)                                   |
| User            | 1) Person using OFSC  
|                 | 2) Entity used for authentication and authorization, allowing people or external software to access OFSC |
| Work Skill      | 1) Activity that a resource is qualified to perform (resource property)  
|                 | 2) Qualification required to perform an activity (activity property)                                   |
| Work Zone       | Defined geographical area in which a resource can perform an activity                                 |
| Forecasting     | Oracle Field Service Forecasting Cloud Service. New feature of OFSC allowing to forecast the company workload on the basis of historical data |
Oracle Field Service Cloud logs changes made to activities, inventory, routes, etc. The history records the performed operation, the time of such operation, the user which performed the operation and the actual changes which were made. The History API serves as an advanced means of retrieving such actions and their details for further use by external applications.

Events are logged in history and become available for retrieval as soon as they occur. This allows real-time data collection and processing which creates a dynamic picture of OFSC performance.
3 Benefits of Using History API

The functions of the History API are somewhat similar to those of the Outbound API. However, there are certain aspects in which using the History API can be more beneficial:

<table>
<thead>
<tr>
<th>Outbound API</th>
<th>History API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires a Middleware SOAP server for integration between OFSC and the client application.</td>
<td>No Middleware SOAP server is required on the customer’s side.</td>
</tr>
<tr>
<td>More complex configuration involving message step settings.</td>
<td>No message step configuration.</td>
</tr>
<tr>
<td>Only actions changing activity statuses (‘start_activity’, ‘complete_activity’, etc.) are returned.</td>
<td>In addition to the actions changing activity statuses, changes to the activity properties (‘update_activity’) are returned.</td>
</tr>
<tr>
<td>Supports complex transactions, including changing data in OFSC based on Middleware response.</td>
<td>Focused on retrieving history data, does not change OFSC state.</td>
</tr>
<tr>
<td>Does not specify which properties of the activity, route, etc. were changed.</td>
<td>Specifies the properties changed by the operation.</td>
</tr>
<tr>
<td>Wider and more complex functionality.</td>
<td>Simple and easy to use.</td>
</tr>
</tbody>
</table>

In addition, History API serves as a valuable supplement to the Daily Extract functionality. Daily Extract, as its name suggests, extracts OFSC data on the daily basis for the entire day and even, when the Overnight functionality is enabled, for the day before. At the same time, the History API is intended for real-time continuous data retrieval which enables immediate action tracking.
4 History API Workflow

History API Workflow

In order to start using the History API, the user needs to send the initial request containing only the authentication parameters. For example, the request with HTTP Basic authentication will have the following format:

https://api.etadirect.com/rest/history/v1/route/?company=test.instance&request_auth_basic

If authentication is successful, a response with no history data is returned:

```
{
    "found":true,
    "next_token":"140108-571,0",
    "history":[]
}
```

The response contains the "next_token" field the value of which is to be used in the subsequent request. The "next_token" value defines the moment in time starting from which historical data will be returned. The GET request with the "token" value is to be sent in the following format:

GET /rest/history/v1/route/?company=test.instance&count=150&token=140115-808,27

The previous request will return the first 150 (the "count" parameter value) historical records after the moment referenced by the token. The response will also contain a new "next_token" value (referencing the moment in time when the request was sent) to be used in subsequent requests:

```
{
    "found":true,
    "next_token":"140108-571,0",
    "history":[
        HISTORY RECORDS
    ]
}
```

This procedure allows retrieving a continuous flow of history records starting from the time when the initial request was sent.

History Data Retrieval

To poll for events, the client needs to do the following:

1. Get events by specifying the subscriptionId, page, and limit (optional).
2. Do one of the following:
   - If there are events in the response, then process the events.
   - If there are no events in the response and the returned "nextPage" is equal to the "page" parameter specified in request, then:
     - Wait for an interval of time (such as one minute) for new events to be generated.
     - Go to Step 1.
3. Request a next set of events after the events are processed. Extract either the "nextPage" field from the response and specify it in the next request or use a link with "rel"="next". The link contains "page" and "subscriptionId" parameters. With either the link, or with next page extracted, the client should go to 1.
4. Go to Step 1, either with the link rel=next or with the value of the next Page response field.

The following example shows how to get events:

```
curl -u 'clientId@instanceName:clientSecret' 'https://api.etadirect.com/rest/history/v1/route/?count=2&token=160902-440,0'
```

Response Header Example

```
Connection: close
Accept-Encoding: gzip, deflate
Authorization: Basic *****
Host: api.etadirect.com
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

Response

```
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 25 Apr 2016 12:36:21 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: close

{
"found": true,
"nextTokenPage": "160425-457,1",
"items": [
{
"eventType": "requiredInventoryCreated",
"time": "2016-04-25 12:36:11",
"user": "myroot",
"activityDetails": {
"activityId": 19828,
"resourceId": "Ira1",
"date": "2016-04-25"
},
"requiredInventoryDetails": {
"inventoryType": "CABLE_MODEM"
},
"requiredInventoryChanges": {
"quantity": "1"
}
},
"links": [
{
"rel": "describedby",
"href": "https://api.etadirect.com/rest/ofscCore/v1/metadata-catalog/events"
},
{
"rel": "canonical",
"href": "https://api.etadirect.com/rest/ofscCore/v1/events/?subscriptionId=a0fd97e62abca26a79173c974d1e9c19f46a254a&page=160425-457,0"
},
{
"rel": "next",
"href": "https://api.etadirect.com/rest/ofscCore/v1/events/?subscriptionId=a0fd97e62abca26a79173c974d1e9c19f46a254a&page=160425-457,1"
}
]
}```
How do I ensure that I don’t receive the same set of objects in the subsequent requests?

In every polling request, the client application must send the page parameter, which works in the same way as an event ID, and the limit parameter. Irrespective of the number of events returned, the response contains the nextPage field. Passing the value of the nextPage field to subsequent requests returns only events following the previous call. If the value of the nextPage field is equal to the value of the request parameter "page", then it indicates that all events until current time have been retrieved and the client application can pause before sending the next request.

The sequence of operations is as follows:

1. Perform the call to request two events from the stored page:

   - cURL command:
     ```
     curl -u 'clientId@instanceName:clientSecret' 'https://api.etadirect.com/rest/history/v1/route/?count=2&token=160902-440,0'
     ```

   - Response
     ```
     {
     "nextPage": "160902-478,0",
     "items": [
     {...
     event 1...
     },
     {...
     event 2...
     }
     ]
     }
     ```

2. Perform the next call without any delay since the nextPage value (160902-478,0) is different from received page parameter value (events is equal to limit (two)160902-440,0):

   - cURL command
     ```
     curl -u 'clientId@instanceName:clientSecret' 'https://api.etadirect.com/rest/history/v1/route/?count=2&token=160902-440,0'
     ```

   - Response
     ```
     {
     "nextPage": "160902-478,1",
     "items": [
     {...
     event 3...
     }
     ]
     }
     ```

3. Perform the next call without any delay since the nextPage value (160902-478,1) is different from page parameter value (160902-478,0):

   - cURL command
     ```
     curl -u 'clientId@instanceName:clientSecret' 'https://api.etadirect.com/rest/history/v1/route/?count=2&token=160902-440,1'
     ```

   - Response
     ```
     { "found": true,
     "nextPage": "160902-478,1",
     ```
"items": [
]

You should make a delay before the next call since the nextPage field is the same as the page parameter value (160902-478,1).

Note: The History API uses the database storing history records for 3 months. Any records accumulated in the database over the period of 3 months can be retrieved. The maximum number of records retrieved by a single request is 1,000 (the "count" parameter of the request).
5 Authentication

History API Authentication Methods

You can find information on authentication at the following location: Authentication Methods.
6 History API Access

Access to the History API is set using **Configuration, Applications**.

Click **Add New** and select the **History API** check box to grant access to the History API.
Chapter 6
History API Access
7 History API Operation Description

Objects Monitored by History

The History API returns history records of changes to the following types of objects of OFSC:

- route (one calendar day of one resource with a list of scheduled or non-scheduled activities assigned to the resource)
- activity (time-consuming action performed by a resource)
- activity link (correlation between start and end of two activities)
- resource preference (rules defining required, preferred or forbidden resources for a certain activity. Resource preferences determine activity assignment)
- required inventory (inventory necessary to complete a certain activity)
- inventory (equipment installed or deinstalled during an activity performance)
- request (message generated as a result of ‘send Service Request’ operation and assigned to a specific entity in OFSC)

History API Request

The History API uses a single method, namely, the GET method, to retrieve all history for all objects in OFSC which were updated within the time elapsed since the previous token issuance. The GET request has the following format:

```
GET /rest/history/v1/route/?company=test.instance&count=150&token=140115-808,27
```

where:

- company – company name as used in authentication
- count – the maximum number of records to be returned in the response (the maximum value: 1000, the default value: 100)
- token – string defining the point from which history is to be returned. The token is created at time of the initial valid request and remains valid for the next 3 months. Each subsequent response contains a new token. If a request is sent with the same token as the previous request, the response contains the history returned for the previous request plus all changes logged after that. However, if a request is sent with a token for which the 3-months validity period has expired, the request is not processed. If a request is sent with the new token, the response contains changes logged after such token creation.

Optionally, the request can contain the ‘debug’ parameter allowing data to be returned in human-friendly format. Otherwise, the data is returned as a single string.

The ‘debug’ parameter use is recommended for testing the functionality. In this case the optimum scenario includes authentication using the HTTP Basic method which requires no complex configuration and a request sent with the ‘debug’ parameter which will allow the user to view the response in the easily readable format. The user is then able to check whether the returned data is complete and correct.

> Note: a response in such format will not be processed by the external application.
History API Response

A valid GET request returns a response containing the records of actions performed on OFSC objects, up to the "count" number defined in the request. A single response may contain records on different objects update.

Each History API response always contains a package header.

The package header contains the request result ('true' for a valid request and 'false' for an invalid one) and the next token which can be used in subsequent history requests. The package header is followed by the history records.

For the sake of clarity, all examples are shown in the human-friendly format.

```json
{
    "found":true,
    "next_token":"140108-571,0",
    "history":[
    HISTORY RECORDS
    ]
}
```

The history data returned in a response is organized in records, each containing one change to an OFSC object. Each record has a record header consisting of the following fields:

- "operation" – name of the operation logged in the history
- "action_time" – time of the action
- "user" – OFSC user performing the action. If the action was performed by the system, the "user" field is omitted

```json
{
    "operation": "create_route",
    "action_time": "2014-01-15 13:36:54",
    "user": "admin",
    "route": {
        "date": "2014-01-15",
        "resource_id": "33036",
        "changes": {
            "activated": "2014-01-15 13:36:00",
            "time_zone": "Eastern",
            "calendar_time_from": "2014-01-15 12:00:00",
            "calendar_time_to": "2014-01-15 21:00:00",
            "calendar_points": "100"
        }
    }
}
```

The header is followed by the name of the object, the fields identifying the particular object and the description of changes performed to the object.

History API Response Description

The description of history records logged for each OFSC object monitored by history is as follows:

- **Note:** the History API logs the same operations which are used in respect of the corresponding objects by the Activity Management API.
Route Update

The following route operations are logged in the history and returned by the History API:

- create_route (route formation in OFSC occurring when activities are assigned to a resource or when the resource having no assigned activities activates their route to start their working day)
- update_route (change of the resource’s working calendar)
- start_route (start of the resource’s working day by activating the route)
- end_route (end of the resource’s working day by deactivating the route)
- restart_route (reopening of the resource’s route by reactivating the previously deactivated route)

Each route-related record contains the route identifier fields defining the route to which the history record is related:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>route date in the YYYY-MM-DD format</td>
</tr>
<tr>
<td>resource_id</td>
<td>external ID of the resource to which the route is assigned</td>
</tr>
</tbody>
</table>

The records may optionally contain the "changes" field containing the fields and their values changed for the route. The "changes" field logs updates of the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activated</td>
<td>date and time of the route activation in the YYYY-MM-DD HH:MM:SS format</td>
</tr>
<tr>
<td>deactivated</td>
<td>date and time of the route deactivation in the YYYY-MM-DD HH:MM:SS format</td>
</tr>
<tr>
<td>calendar_time_from</td>
<td>date and time of start of the resource’s calendar assigned to the route in the YYYY-MM-DD HH:MM:SS format</td>
</tr>
<tr>
<td>calendar_time_to</td>
<td>date and time of end of the resource’s calendar assigned to the route in the YYYY-MM-DD HH:MM:SS format</td>
</tr>
<tr>
<td>calendar_points</td>
<td>number of points assigned to the resource for the route (if the company uses points)</td>
</tr>
<tr>
<td>time_zone</td>
<td>the resource’s time zone</td>
</tr>
<tr>
<td></td>
<td>Note: all times are returned in GMT.</td>
</tr>
<tr>
<td>traveling_time</td>
<td>time from the end of the last activity in the route till the arrival to the final location (contains the actual traveling time value if the 'Working time includes travel from last activity' option is enabled for the resource, otherwise contains '0')</td>
</tr>
</tbody>
</table>

Route Update Response Example

The following example shows the history record of an activated route creation:

```json
{
   "operation": "create_route",
   ...
}```
"action_time": "2014-01-15 13:36:54",
"user": "admin",
"route": {
"date": "2014-01-15",
"resource_id": "33036",
"changes": {
"activated": "2014-01-15 13:36:00",
"time_zone": "Eastern",
"calendar_time_from": "2014-01-15 12:00:00",
"calendar_time_to": "2014-01-15 21:00:00",
"calendar_points": "100"
}
}
}

Activity Update

The following activity operations are logged in the history and returned by the History API:

- create_activity (formation of a new activity)
- update_activity (change of the activity properties)
- start_activity (change of the activity status to 'started' meaning that the resource has begun performing the activity)
- suspend_activity (change of the activity status to 'pending' and a simultaneous creation of a new activity with the 'suspended' status is created duplicating the original activity)
- complete_activity (change of the activity status to 'complete' meaning a successful performance of the activity)
- notdone_activity (change of the activity status to 'notdone' meaning that a started activity cannot be successfully completed for some reason)
- cancel_activity (change of the activity status to 'cancelled' meaning that the activity has not been started and will not be performed)
- delete_activity (removal of a canceled activity from the system. An activity can be deleted only from an inactive route)
- delay_activity (change of the end time of a started activity to extend the activity duration)
- reopen_activity (creation of a 'pending' activity duplicating a previously completed, cancelled or not-done activity)
- prework_activity (creation of an additional activity necessary for performance of a pending activity. Preworks are always created in the 'started' status)

For the following operations the identifier fields contain old values and the "changes" field contains new values:

- move_activity (activity reassignment to a different resource and/or date)
- reschedule_activity (activity move to a different date and/or time)

Each activity-related record contains the activity identifier fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>date of the route to which the activity is assigned in the YYYY-MM-DD format</td>
</tr>
<tr>
<td>resource_id</td>
<td>external ID of the resource to which the activity is assigned</td>
</tr>
<tr>
<td>activity_id</td>
<td>internal activity ID</td>
</tr>
<tr>
<td>appt_number</td>
<td>activity number</td>
</tr>
</tbody>
</table>
The records may optionally contain the "changes" field containing the fields and their values changed for the route. The "changes" field logs updates of the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>activity type</td>
</tr>
<tr>
<td>status</td>
<td>activity status</td>
</tr>
<tr>
<td>worktype</td>
<td>activity work type</td>
</tr>
<tr>
<td>workzone</td>
<td>activity work zone</td>
</tr>
<tr>
<td>duration</td>
<td>activity duration in minutes</td>
</tr>
<tr>
<td>time_slot</td>
<td>label of activity time slot</td>
</tr>
<tr>
<td>service_window_start</td>
<td>customer service window start date and time in the YYYY-MM-DD HH:MM:SS format</td>
</tr>
<tr>
<td>service_window_end</td>
<td>customer service window end date and time in the YYYY-MM-DD HH:MM:SS format</td>
</tr>
<tr>
<td>delivery_window_start</td>
<td>activity delivery window start time in the HH:MM:SS format</td>
</tr>
<tr>
<td>delivery_window_end</td>
<td>activity delivery window end time in the HH:MM:SS format</td>
</tr>
<tr>
<td>sla_window_start</td>
<td>activity SLA window start in the YYYY-MM-DD</td>
</tr>
<tr>
<td>sla_window_end</td>
<td>activity SLA window end in the YYYY-MM-DD</td>
</tr>
<tr>
<td>name</td>
<td>customer’s name</td>
</tr>
<tr>
<td>phone</td>
<td>customer’s regular (land) phone number</td>
</tr>
<tr>
<td>email</td>
<td>customer’s email address</td>
</tr>
<tr>
<td>address</td>
<td>customer’s address</td>
</tr>
<tr>
<td>city</td>
<td>customer’s city of residence</td>
</tr>
<tr>
<td>zip</td>
<td>customer’s zip/post code</td>
</tr>
<tr>
<td>state</td>
<td>customer’s state of residence</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>language</td>
<td>notification language</td>
</tr>
<tr>
<td></td>
<td>company specific language label (en, es, etc.)</td>
</tr>
<tr>
<td>reminder_time</td>
<td>reminder notification time: how many minutes before the activity start time</td>
</tr>
<tr>
<td></td>
<td>the customer should be notified</td>
</tr>
<tr>
<td>time_zone</td>
<td>customer’s time zone</td>
</tr>
<tr>
<td></td>
<td>Note: all times are returned in GMT</td>
</tr>
<tr>
<td>coord_status</td>
<td>whether or not activity coordinates were found</td>
</tr>
<tr>
<td>coordx</td>
<td>latitude of the activity (of the customer’s location)</td>
</tr>
<tr>
<td>coordy</td>
<td>longitude of the activity (of the customer’s location)</td>
</tr>
<tr>
<td>start_time</td>
<td>ETA time (for and time when the activity was started)</td>
</tr>
<tr>
<td>end_time</td>
<td>predicted or actual end time of activity</td>
</tr>
<tr>
<td>team_id</td>
<td>external ID of the team-holder – the head resource within a team</td>
</tr>
<tr>
<td>unordered</td>
<td>parameter defining that there is no specific time within the resource’s</td>
</tr>
<tr>
<td></td>
<td>route when the activity has to be performed</td>
</tr>
<tr>
<td>position_in_route</td>
<td>number of the activity in the route</td>
</tr>
<tr>
<td>first_manual_operation</td>
<td>name of the first manual operation performed to the activity after routing</td>
</tr>
<tr>
<td>first_manual_operation_user</td>
<td>login of the user performing the first manual operation to the activity</td>
</tr>
</tbody>
</table>

Activity Update Response Example

The following example shows the history record of a started route creation:

```json
{
  "operation": "create_activity",
  "action_time": "2014-01-15 16:34:29",
  "user": "admin",
  "activity": {
    "date": "2014-01-15",
    "resource_id": "33011",
    "activity_id": 3998009,
    "changes": {
      "ACTIVITY_NOTES": "just lunch",
      "type": "regular",
      "status": "started",
      "worktype": "LU",
      "duration": "60",
```
"start_time": "2014-01-15 16:34:00",
"time_slot": "lunch",
"service_window_start": "2014-01-15 20:00:00",
"service_window_end": "2014-01-15 20:30:00",
"language": "en",
"time_zone": "Pacific",
"position_in_route": "1",
"time_of_booking": "2014-01-15 16:34:29",
"time_of_assignment": "2014-01-15 16:34:29"
}

The following example shows the history record of an activity move between resources. In this case, the identifier field "resource_id" contains the old value, while the "resource_id" field in "changes" contains the new one:

{  
  "operation": "move_activity",
  "action_time": "2014-01-15 17:34:28",
  "user_id": "admin",
  "activity": {  
    "date": "2014-01-15",
    "resource_id": "33015", // old resource id
    "activity_id": 3956550,
    "appt_number": "#137163458",
    "customer_number": "019942164",
    "changes": {  
      "resource_id": "33011" // new resource id
    }
  }
},

{  
  "operation": "update_activity",
  "action_time": "2014-01-15 17:34:28",
  "user": "admin",
  "activity": {  
    "date": "2014-01-15",
    "resource_id": "33011", // new resource id
    "activity_id": 3956550,
    "appt_number": "#137163458",
    "customer_number": "019942164",
    "changes": {  
      "duration": "30",
      "start_time": "2014-01-16 00:13:52",
      "traveling_time": "11",
      "delivery_window_start": "2014-01-15 23:30:00",
      "delivery_window_end": "2014-01-16 00:45:00",
      "position_in_route": "12",
      "time_of_assignment": "2014-01-15 17:34:28"
    }
  }
}

The following example shows the history record of an activity reordering:

{  
  "operation": "reorder_activity",
  "action_time": "2015-01-25 16:25:24",
  "user": "admin",
  "activity": {  
    "date": "2015-01-25",
    "resource_id": "33015",
    "activity_id": 3954830,
    "appt_number": "137165178",
    "customer_number": "019942099"
  }
}
Activity Link Update

The following activity link operations are logged in the history and returned by the History API:

- link_activities (creation of a link between two activities)
- unlink_activities (removal of a link between two activities)

Each activity link-related record contains the link identifier fields:

For the first activity:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link_type</td>
<td>label of the link type</td>
</tr>
<tr>
<td>from_activity</td>
<td>structure containing identifier fields of the activity from which the link is established</td>
</tr>
<tr>
<td>to_activity_id</td>
<td>internal ID of the activity to which the link is established</td>
</tr>
<tr>
<td>to_appt_number</td>
<td>number of the activity to which the link is established</td>
</tr>
</tbody>
</table>

For the second activity:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link_type</td>
<td>label of the link type</td>
</tr>
<tr>
<td>to_activity</td>
<td>structure containing identifier fields of the activity to which the link is established</td>
</tr>
<tr>
<td>from_activity_id</td>
<td>internal ID of the activity from which the link is established</td>
</tr>
<tr>
<td>from_appt_number</td>
<td>number of the activity from which the link is established</td>
</tr>
</tbody>
</table>

⚠️ Note: for each activity link created or removed, the History API response contains two records – for the first activity in the link and for the second activity.

Activity Link Update Response Example

The following example shows the history record of a link creation:

```json
{
  "operation": "link_activities",
  "action_time": "2014-01-15 16:54:48",
  "user": "admin",
  "activity_link": {
    "link_type": "start-before",
    "from_activity": {
      "date": "2014-01-15",
      "..." // other fields
    },
    "to_activity": {
      "..." // other fields
    }
  }
}
```
Oracle Field Service Cloud
Integrating with History API

Chapter 7
History API Operation Description

Resource Preference Updates

The following activity link operations are logged in the history and returned by the History API:

- set_resource_preferences (definition of resources preferred, required or forbidden for an activity)
- del_resource_preferences (deletion of any previously set resource preferences)

Each preference-related record contains the resource preference identifier fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activity</td>
<td>structure containing identifier fields of the activity for which resource preferences are set or deleted</td>
</tr>
<tr>
<td>type</td>
<td>resource preference type (preferred, forbidden, or required)</td>
</tr>
<tr>
<td>resource_id</td>
<td>external ID of the resource for which preferences are set or deleted</td>
</tr>
</tbody>
</table>

Activity Link Update Response Example

The following example shows the history record of a preferred resource adding:

```json
{
  "operation": "set_resource_preferences",
  "action_time": "2014-01-15 17:05:16",
  "user": "admin",
  "preference": {
    "activity": {
      "date": "2014-01-15",
      "resource_id": "33003",
      "activity_id": 3956464,
      "appt_number": "#137163544",
      "customer_number": "019922286"
    },
    "resource_id": "11129",
    "to_activity_id": 3954821,
    "to_appt_number": "#137165187"
  }
}
```
Required Inventory Updates

The following required inventory operations are logged in the history and returned by the History API:

- `create_required_inventory` (adding required inventory to an activity)
- `update_required_inventory` (changing properties of required inventory)
- `delete_required_inventory` (removing required inventory from an activity)

Each required inventory-related record contains the required inventory identifier fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activity</td>
<td>structure containing identifier fields of the activity for which required inventory is set, updated or removed</td>
</tr>
<tr>
<td>type</td>
<td>required inventory type</td>
</tr>
<tr>
<td>resource_id</td>
<td>required inventory model</td>
</tr>
</tbody>
</table>

The records may optionally contain the "changes" field containing the "quantity" field, if the inventory quantity was changed in the operation (for non-serialized inventory).

Required Inventory Update Response Example

The following example shows the history record of required inventory adding to an activity:

```json
{
  "operation": "create_required_inventory",
  "action_time": "2014-01-28 12:14:02",
  "user": "admin",
  "required_inventory": {
    "type": "Wire",
    "model": "RG-45",
    "activity": {
      "date": "2014-01-28",
      "resource_id": "33035",
      "activity_id": 3954885,
      "appt_number": "137165123",
      "customer_number": "019921925"
    },
    "changes": {
      "model": "RG-45",
      "quantity": "5"
    }
  }
}
```

Inventory Updates

The following inventory operations are logged in the history and returned by the History API with the activity identifier:

- `install_inventory` (moving the inventory from the 'resource' pool to the 'install' pool)
- `deinstall_inventory` (moving the inventory from the 'customer' pool to the 'deinstall' pool)
• create_customer_inventory (adding inventory to the 'customer' pool)
• delete_inventory (removing inventory from the 'customer' pool)
• undo_install_inventory (moving the inventory from the 'install' pool to the 'resource' pool)
• undo_deinstall_inventory (moving the inventory from the 'deinstall' pool to the 'customer' pool)
• exchange_inventory_install (moving the inventory from the 'resource' pool to the 'install' pool in an 'exchange_inventory' operation)
• exchange_inventory_deinstall (moving the inventory from the 'customer' pool to the 'deinstall' pool in an 'exchange_inventory' operation)
• update_customer_inventory (change of customer inventory properties)

Note: ‘exchange_inventory_install' and ‘exchange_inventory_deinstall' are two records of the same 'exchange_inventory' operation but logged for each of the two inventories involved in the exchange. The following inventory operations are logged in the history and returned by the History API without the activity identifier:

• create_resource_inventory (adding inventory to the 'resource' pool)
• update_resource_inventory (change of resource inventory properties)
• delete_resource_inventory (removing inventory from the 'resource' pool)

Each inventory-related record contains the inventory identifier fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activity</td>
<td>structure containing identifier fields of the activity to which the inventory is assigned. As mentioned in the previous section, the 'activity' structure is returned only for certain inventory operations.</td>
</tr>
<tr>
<td>inventory_id</td>
<td>inventory ID</td>
</tr>
<tr>
<td>type</td>
<td>inventory type</td>
</tr>
<tr>
<td>serial_number</td>
<td>inventory serial number (for serialized inventory)</td>
</tr>
</tbody>
</table>

The records may optionally contain the "changes" field containing the fields and their values changed for the inventory. The "changes" field logs updates of inventory fields and company-defined properties. In addition to the fields listed in the following table, changes to other inventory fields existing in the system and company-defined inventory properties defined in the specific company can also be returned.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>inventory pool</td>
</tr>
<tr>
<td>quantity</td>
<td>number of inventories (for non-serialized inventory only)</td>
</tr>
</tbody>
</table>

Inventory Update Response Example
The following example shows the history record of inventory installation:
The following example shows the history record of customer inventory creation:

```
{
  "operation": "create_customer_inventory",
  "action_time": "2014-01-28 11:24:32",
  "user": "admin",
  "inventory": {
    "serial_number": "HRSC636029",
    "type": "NT",
    "status": "customer",
    "inventory_id": 21034416,
    "activity": {
      "date": "2014-01-28",
      "resource_id": "33003",
      "activity_id": 3954809,
      "appt_number": "#137165199",
      "customer_number": "019901104"
    },
    "changes": {
      "status": "customer",
      "type": "NT",
      "serial_number": "HRSC636029",
      "quantity": "1",
      "EQUIPMENT_ROOM_CODE": "test"
    }
  }
}
```

**Request Creation**

The following request operations are logged in the history and returned by the History API:

- create_customer_request (creation of a service request assigned to an activity)
- create_inventory_request (creation of a service request assigned to an inventory)
- create_resource_request (creation of a service request assigned to a resource)

Each request-related record contains the request identifier fields:
Field | Description
--- | ---
type | request type
date | request date
resource_id | external ID of the resource associated with the request

⚠️ Note: Depending on the request type, the identifier fields may also contain activity (for customer requests) and inventory (for inventory requests) identifier structures.

The records may optionally contain the "changes" field containing the fields and their values changed for the service request. The "changes" field logs updates of any service request company-defined properties existing in the system.

Request Creation Response Example

The following example shows the history record of a customer request creation:

```json
{
    "operation": "create_customer_request",
    "action_time": "2014-02-05 12:04:24",
    "user": "admin",
    "request": {
        "type": "SR",
        "date": "2014-02-06",
        "resource_id": "routing",
        "activity": {
            "activity_id": 3952162,
            "appt_number": "#137167846",
            "customer_number": "019892755"
        },
        "changes": {
            "sr_body": "asfd"
        }
    }
}
```

The following example shows the history record of an inventory request creation:

```json
{
    "operation": "create_inventory_request",
    "action_time": "2014-02-05 13:42:57",
    "user": "admin",
    "request": {
        "type": "SR",
        "date": "2014-02-05",
        "resource_id": "33015",
        "activity": {
            "activity_id": 3954828,
            "appt_number": "137165180",
            "customer_number": "019946338"
        },
        "inventory": {
            "serial_number": "7213125210",
            "type": "TV",
            "inventory_id": 20994113
        },
        "changes": {
            "sr_subject": "asfd",
            "sr_body": "asfd"
        }
    }
}
```
} }
} 

8 Previous Versions

History API version 16.2 ensures backward compatibility with the previous versions of the product.

As compared to the previous versions, the current version of the History API includes 4 additional activity properties:

- sla_window_start
- sla_window_end
- first_manual_operation
- first_manual_operation_user