Working with Unified CCE APIs

Change Log

This section notes the new and changed APIs in this Packaged CCE release.

New APIs

- Contact Center AI Global Config API

Updated APIs

See	Notes
Status API	In Operation Rules, updated the topic to include Contact Center AI Global rule
Call Type API	Updated the topic to include Contact Center AI configuration details
Bulk Job API	Updated the topic to include Contact Center AI service information
Operation API	
Agent API	

See	Internal Change History
Call Type API	Contact Center AI service information removed and added configId
Agent API, Bulk Job API, and Operation API	Contact Center AI services information added
Status API	In Operation Rules, updated the topic to include Contact Center AI Global rule
Agent Service API	As this was descoped, removed the information

API Operations

There are five API operations, and they are invoked by HTTP methods.

Responses are provided using HTTP headers and HTTP body containing XML. For information on XML, see XML, on page 6.

create

The create operation uses the HTTP POST method to make one new item and return the URL of that item in the HTTP location header. That URL can then be used to perform the get, update, and delete operations. An XML body containing the parameters and values for the new item must be specified.

delete

The delete operation uses the HTTP DELETE method to delete one item. The item may be marked for deletion or permanently deleted depending on the item type.

To delete more than one item at a time, refer to the Operation API.

You cannot delete **BuiltIn** items (those automatically created by the system, such as the **BuiltIn** bucket interval), items referenced in scripts, or items referenced by other items.

get

The get operation uses the HTTP GET method to retrieve one item. For example, to return one bucket interval record, perform the get operation using the URL:
`https://<server>/unifiedconfig/config/bucketinterval/<id>` .

list

The list operation uses the HTTP GET method to retrieve a list of items. For example, to retrieve a list of bucket intervals, perform the list operation using the URL:
`https://<server>/unifiedconfig/config/bucketinterval`. See also Permissions, on page 9, Pagination, on page 7, Search, on page 10, and Sort, on page 11.

Query parameters:

- Summary list: Some APIs have parameters that include a large amount of data when returned, such as collections of references. Use this query parameter to reduce the number of parameters returned for each item in the list. For example, in the Skill Group API, if skill groups contain a large number of agents, a

large amount of data may be returned. Use this query option to return the basic skill group data along with the number of agents having the skill. Append the query parameter `summary=true` to the URL for the API; for example, `https://<server>/unifiedconfig/config/skillgroup?summary=true`.

update

The update operation uses the HTTP PUT method to modify one item. An XML body containing the parameters and values to update must be specified. For example, to update the name of a bucket interval, perform the update operation on the URL `https://<server>/unifiedconfig/config/bucketinterval/(id)` with the following body:

```
<bucketInterval>
   <name>newName</name>
   <changeStamp>0</changeStamp>
</bucketInterval>
```

Access

Administrator Access

Administrator access to Unified CCE Administration APIs and items is defined by the role for which the administrator is responsible.

The following API is not available for administrators:

- Agent Team API

The following APIs allow update with restrictions:

- Agent Call API
  - When updating an agent, administrators can only change the following parameters:
    - skillGroups
    - defaultSkillGroup
    - skillGroupsAdded
    - skillGroupsRemoved
    - agentAttributes
    - agentAttributesAdded
    - agentAttributesRemoved
    - password
- Skill Group API
  - When updating a skill group, administrators can only change the following parameters:
    - agents

- agentsAdded
- agentsRemoved

The following APIs are only available in the Contact Director deployment:

- Contact Share Group API
- Contact Share Rule API

Supervisor Access

The following APIs are read only:

- Agent Team API
  - Supervisors can only see teams that are on their peripheral.
- Precision Queue API

The following APIs allow update with restrictions:

- Attribute API
  - When updating an attribute, supervisors can only modify the collection of agentAttributes by adding, removing, or changing the value of agents who are on their teams.
- Agent Call API
  :
  - Supervisors can only see and update agents who are on their teams.
  - When updating an agent, supervisors can only change the following parameters:
    - skillGroups
    - defaultSkillGroup
    - skillGroupsAdded
    - skillGroupsRemoved
    - agentAttributes
    - agentAttributesAdded
    - agentAttributesRemoved
    - password
  - The Operation API can also be used to perform updates on agents.
- Skill Group API:
  - Supervisors can only see and update skill groups that are on their peripheral.
  - When updating a skill group, supervisors can only modify the collections of agents by adding or removing agents who are on their teams.

Agent Access

Agents have no access to the Unified CCE Administration APIs.

Authentication

To authenticate administrators and supervisors must provide a fully qualified user name (for example, user@domain.com) and password.

Deployment

In most deployments, all APIs are available except for the Contact Sharing APIs. You can only use the Contact Sharing APIs in a Contact Director deployment.

Usage and Behavior

Duplicate Parameters

If a parameter is duplicated, the final value that is specified will be used by the API.

Read-Only Fields

Read-only parameters are ignored on create and update operations.

References

References are a type of parameter that provide a way to connect one item to another item, defining the relationship between them.

For example, to define which team an agent belongs to, the agent contains a reference to a team. When performing list or get operations, the reference contains the refURL of the item and the name. For example:

```
<agent>
   <team>
     <refURL>/unifiedconfig/team/5000</refURL>
     <name>NameOfTeam</name>
   </team>
  …
</agent>
```

For items that do not have a name parameter, other parameters such as firstName and lastName are included.

```
<agent>
  <refURL>/unifiedconfig/config/agent/5000</refURL>
  <firstName>Jane</firstName>
  <lastName>Doe</lastName>
  <userName>username</userName>
  <agentId>8007</agentId>
  <canRemove>true</canRemove>
</agent>
```

When doing create or update, only the refURL parameter is required. Additional parameters are ignored. For example:

```
<agent>
   <team>
     <refURL>/unifiedconfig/team/5000</refURL>
   </team>
  …
</agent>
```

Items can also contain a collection of references. For example, if an agent belongs to multiple skill groups, the skillGroups parameter contains a reference to each associated skill group:

```
<agent>
  <skillGroups>
    <skillGroup>
      <refURL>/unifiedconfig/config/skillgroup/5001</refURL>
      <name>FirstSkill</name>
    </skillGroup>
    <skillGroup>
      <refURL>/unifiedconfig/config/skillgroup/5005</refURL>
      <name>AnotherSkill</name>
    </skillGroup>
  </skillGroups>
  …
</agent>
```

XML

XML is case sensitive. When XML data is sent to the server, the tag names must match. <Name> and <name> are two different XML elements.

Error Responses

Operations that fail return an HTTP status code (HTTP 1.1 Status Codes) indicating if there was a client error or server error. The body of the response contains a collection of API error items to provide additional information about the failure.

Parameters

- errorType: Indicates the type of error. This is the primary identifier for the problem and can be used to map the type to a user readable string. For example, if your application receives an error with the errorType of invalidInput.fieldRequired, then you could display "This field is a required field; it cannot be left blank" to the user.
- errorData: The name of the parameter that had the error.
- errorMessage: Extra information about the error that is intended for the developer. This information is typically a sentence or other string. It is not localized, so it should not be shown to the user.
- errorDetail: Some errors contain additional detail parameters that are included in the errorDetail parameter.
  - If the error type is invalidInput.outOfRange, then errorDetail includes the following parameters:
    - min: The minimum value allowed.
    - max: The maximum value allowed.
  - If you attempt to delete an item that is in use by other items, the errorType is referenceViolation.api and the errorDetail includes the following parameters:
    - referenceType: The type of item that references the item you tried to delete.
    - references: A collection of references, referencing the item you tried to delete, including the name and refURL of each referencing item.
    - totalCount: The total number of items referencing the item you attempted to delete.
    - totalShown: The total number of items included in the references collection.

Example Error Response

The following error is returned when attempting to create a call type with a negative value for the serviceLevelThreshold parameter:

```
<apiErrors>
    <apiError>
        <errorData>serviceLevelThreshold</errorData>
        <errorDetail>
            <min>1</min>
            <max>2147483647</max>
        </errorDetail>
        <errorMessage>This field must contain a value from 1 to 2147483647</errorMessage>
        <errorType>invalidInput.outOfRange</errorType>
    </apiError>
</apiErrors>
```

Pagination

Pagination allows you to limit the number of items returned by the list operation and provides information on how to get other pages.

Query Parameters

- startIndex: Specfies the index of the item at which to start. Zero-based: 0 is the first item.
- resultsPerPage: Specifies the number of items to retrieve. Minimum:1. Default: 25. Maximum: 100.

Returned Parameters

- totalResults: Total number of items.
- resultsPerPage: Number of items requested per page.
- startIndex: The index of the first item returned. If you request a startIndex that is greater than total items, a full last page is returned.
- nextPage: The URL to get the next page. This parameter is not returned if you are on the last page.
- prevPage: The URL to get the previous page. This parameter is not returned if you are on the first page.
- firstPage: The URL to get the first page.
- lastPage: The URL to get the last page.
- searchTerm: The value specified in the search query parameter. See Search, on page 10.
- sortTerm: The value specified in the sort query parameters. See Sort, on page 11.

Note Query parameters for search and sort are included in the URL.

Example Response

```
<pageInfo>
      <resultsPerPage>2</resultsPerPage>
      <startindex>0</startIndex>
      <totalResults>10</totalResults>
       <firstPage>  http://<server>/bucketIntervals/?resultsPerPage=2</firstPage>
      <lastPage> http://<server>/bucketIntervals/?startIndex=8&resultsPerPage=2</lastPage>

      <prevPage/>
```

```
      <nextPage> http://<server>/bucketIntervals/?startIndex=2&resultsPerPage=2</nextPage>
 </pageInfo>

<bucketIntervals>
   <bucketInterval/>
   <bucketInterval/>
</bucketIntervals>
```

Shared Parameters

changeStamp

- The version of the item. Initially set during a create (create, on page 2) operation.
- A changeStamp is a required parameter for the body of a PUT (update, on page 3) operation for items. If you do not provide a changeStamp, the update fails. This mechanism is in place so that two clients cannot edit the record at the same time.
- If the update is successful, the changeStamp is incremented.

description

- A description for this item.
- Optional parameter.
- No restriction of characters; OEM locale supported characters are allowed. For information on how to configure your system to support built-in character sets, see the latest version of the document *Cisco Unified Contact Center Enterprise Installation and Upgrade Guide* at https://www.cisco.com/c/en/us/support/customer-collaboration/unified-contact-center-enterprise/products-installation-guides-list.html.
- Maximum length of 255 characters.

name

- Required parameter.
- Maximum length of 32 characters allowed.
- Valid characters are period (.), underscore (_), and alphanumeric. The first character must be alphanumeric.
- Does not allow internationalized characters.

refURL

- The identifier for an item.
- Read-only parameter.

Permissions

Permissions information is included in list responses to indicate the write operations that the user is allowed to perform. If the API does not support any write operations, then permissions information is not returned.

Parameters

- canCreate: Indicates whether a create operation is allowed. Values are true/false. If the create operation is not supported by the API, then this parameter is not returned.
- canUpdate: Indicates whether an update operation is allowed. Values are true/false. If the update operation is not supported by the API, then this parameter is not returned.
- canDelete: Indicates whether a delete operation is allowed. Values are true/false. If the delete operation is not supported by the API, then this parameter is not returned.
- role: Type of role of the user performing the request. Values are administrator/supervisor.

Example Get Response

```
<permissionInfo>
    <canCreate>false</canCreate>
    <canUpdate>true</canUpdate>
    <canDelete>false</canDelete>
    <role>Administrator</role>
</permissionInfo>
```

Synchronous vs. Asynchronous Writes

Synchronous API calls are blocking calls that do not return until either the change has been completed or there has been an error. For asynchronous calls, the response to the API call is returned immediately with a polling URL while the request continues to be processed. In heavier load conditions, it can be more efficient to submit multiple async calls and periodically check the status than to wait for each call to complete before submitting the next one.

The following examples describe how to use the asynchronous feature to create a call type.

Performing Asynchronous Operations

The create, update, and delete operations can be performed asynchronously by including the query parameter `async=true`. The request is accepted if the operation is valid and the number of outstanding requests does not exceed the capacity. If the request is accepted, the response includes the following items:

- The response code is HTTP 202, indicating that the request has been accepted for processing.
- The location header specifies a URL that can be polled to receive updated information on the progress of the request.
- The response includes a body. See the next section **Asynchronous result parameters**.

Asynchronous Result Parameters

- progress: Indicates the current state of the request. Values include the following states:
    - IN_QUEUE: The request passed validation and capacity checks and was put in the queue.
    - IN_PROGRESS: The request is being processed.

Polling the Asynchronous Request Status

Use the URL from the location header of an asynchronous operation request to get updated status. Responses of this request are:

- If the request has not completed yet, the response contains the HTTP 202 response code, a location header with polling URL, and a response body.
- If the request has completed, the response is identical to the responses of synchronous operations, including the following:
  - For a successful create, the response code is HTTP 201 and the location header has the URL of the created item.
  - For a successful update or delete, the response code will be HTTP 200.
  - For an unsuccessful update, a body will provide information about the failure.
- If the request has been in queue for over 30 seconds, then it is removed and an error indicates that the request timed out.

Search

The list operation can be modified to return data you are looking for by applying the search query parameter.

Default Search Parameters

Typically, the name and description fields are searched when specifying a search string. Refer to each API section for the default search parameters permitted. For example, a query parameter of `q=abc` causes the list operation to return only entries with a name or description containing **abc**. The search value for default parameters has the following behaviors and restrictions:

Values:

- Are case-insensitive.
- Can be contained anywhere in the parameter value.
- Can match any of the default parameters.
- Cannot include SQL wildcards. They are not supported.
- Must be URL encoded. For example, **&** must be converted to **%26** so that it is not treated as a separator for additional query parameters.

Note The default search is a case-insensitive substring search on multiple columns and hence may take a longer time to respond, depending on the number of records and columns to search for.

Advanced Search

Advanced search parameters allow specific parameters to be searched. Refer to each API section for the advanced search parameters permitted. Advanced search parameters can be combined with a default search value. For example, applying the search query parameter of `q=abc routingType:1` to a dialed number

list operation returns results where the routingType is set to one, and one of the default search parameters contains **abc**. Advanced search also has the following restrictions:

- Search terms must be separated by a space.
- Search terms can be specified in any order.

Sort

A sort query parameter can be used to specify the order of the results in a list response.

The query parameter is **sort=<parameterName>** order, where:

- parameterName: The name of the parameter that you want to sort on. This is case sensitive, so it must match the parameter in the API exactly.
- order: Specifies the order of the sort. Values are as follows:
  - asc: Perform an ascending sort. This is the default if no order is specified.
  - desc: Perform a descending sort.

Example

For example, to find all the CallTypes whose name or description contains *supervisor*, sorted in ascending order by *name*:

```
https://<server>/unifiedconfig/config/calltype?q=supervisor&sort=name
```

Sort