The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: http://www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

© 2016 Cisco Systems, Inc. All rights reserved.
# CONTENTS

## CHAPTER 1
### Working with Unified CCE APIs
- Change Log
- API Operations
- Access
- Usage and Behavior
- Error Responses
- Pagination
- Shared Parameters
- Permissions
- Synchronous vs. Asynchronous Writes
- Search
- Sort

## CHAPTER 2
### Active Directory Domain API

## CHAPTER 3
### Administrator API

## CHAPTER 4
### Agent API

## CHAPTER 5
### Agent Team API

## CHAPTER 6
### Attribute API

## CHAPTER 7
### Bucket Interval API

## CHAPTER 8
### Bulk Job API
CHAPTER 9  Congestion Control API  37

CHAPTER 10  Contact Share Group API  39

CHAPTER 11  Contact Share Rule API  41

CHAPTER 12  Context Service Configuration API  43

CHAPTER 13  Context Service Registration API  45

CHAPTER 14  Deployment API  47

CHAPTER 15  Deployment Type Info API  49

CHAPTER 16  Internet Script Editor API  53

CHAPTER 17  Machine Inventory API  55

CHAPTER 18  Media Routing Domain API  63

CHAPTER 19  Operation API  67

CHAPTER 20  Outbound API  71
  Outbound Campaign API  72
  Campaign Status API  78
  Do Not Call API  80
    Create Do Not Call List  81
  Import API  82
  Personal Callback API  86
  Time Zone API  91

CHAPTER 21  Precision Queue API  93
Contents

CHAPTER 22  Scan API  97

CHAPTER 23  Single Sign-On Global State API  99

CHAPTER 24  Single Sign-On Registration API  101

CHAPTER 25  Single Sign-On Status API  103

CHAPTER 26  Skill Group API  107

CHAPTER 27  Status API  111
  Configuration Rules  112
  Operation Rules  113
Working with Unified CCE APIs

Cisco Unified Contact Center Enterprise (Unified CCE) uses REST-based API functions accessed over HTTP. Five API operations are supported, and each is mapped to an HTTP operation. For more information, see API Operations, on page 3.

This document explains the operations and parameters for each configurable item in Unified CCE.

To review examples on how to interact with the Unified Config and Outbound REST web services in Java, see the CCE Config Sample REST Toolkit at https://developer.cisco.com/site/package-contact-center/documentation/.

- Change Log, page 1
- API Operations, page 3
- Access, page 4
- Usage and Behavior, page 6
- Error Responses, page 7
- Pagination, page 8
- Shared Parameters, page 9
- Permissions, page 10
- Synchronous vs. Asynchronous Writes, page 10
- Search, page 11
- Sort, page 12

Change Log

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

New APIs

- Bulk Job API, on page 33
- Campaign Status API, on page 78
Updated APIs

<table>
<thead>
<tr>
<th>See</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Agent API, on page 17 | New parameters:  
  - ssoEnabled - This parameter is sortable.  
  The userName parameter must now be in the format name@domain.  
  The supervisorUserInfo parameter has been removed. The userName parameter for a supervisor must be set to that supervisor’s UPN account name. |
# 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 7.

## 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 10, Pagination, on page 8, Search, on page 11, and Sort, on page 12.

**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:

```xml
<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**, on page 23

The following APIs allow update with restrictions:

- **Agent API**, on page 17

  * When updating an agent, administrators can only change the following parameters:
• skillGroups
• defaultSkillGroup
• skillGroupsAdded
• skillGroupsRemoved
• agentAttributes
• agentAttributesAdded
• agentAttributesRemoved
• password

• **Skill Group API, on page 107**
  • 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, on page 39**
• **Contact Share Rule API, on page 41**

**Supervisor Access**

The following APIs are read only:

• **Agent Team API, on page 23**
  • Supervisors can only see teams that are on their peripheral.

• **Precision Queue API, on page 93**

The following APIs allow update with restrictions:

• **Attribute API, on page 27**
  • 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 API, on page 17:**
  • 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, on page 107:**
  - 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:

```xml
<agent>
  <team>
```
For items that do not have a name parameter, other parameters such as firstName and lastName are included.

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

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:

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: Specifies the index of the item at which to start. Zero-based: 0 is the first item.


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 11.

• sortTerm: The value specified in the sort query parameters. See Sort, on page 12.

---

**Example Response**

```xml
<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 3) operation.

- A changeStamp is a required parameter for the body of a PUT (update, on page 4) 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 native character sets, see the latest version of the document *Cisco
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

```xml
<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.

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
Active Directory Domain API

Use the Active Directory Domain API to list the active directory domains currently defined in your call center environment. It is read-only, and does not require authentication.

URL

https://<server>/unifiedconfig/config/activedirectorydomain

Operations

- **list**: Retrieves a list of active directory domains.

Parameters

- **name**: The name of the domain.
- **systemDomain**: Indicates if the system is a member of this domain. Values are true/false.

Example Get Response

```xml
<results>
<activedirectoryDomains>
<activedirectoryDomain>
  <name>boston.com</name>
  <systemDomain>true</systemDomain>
</activedirectoryDomain>
<activedirectoryDomain>
  <name>cisco.com</name>
  <systemDomain>false</systemDomain>
</activedirectoryDomain>
</activedirectoryDomains>
</results>
```
CHAPTER 3

Administrator API

An administrator is an Active Directory user who has been provided access to the system. Use the Administrator API to list the administrators currently defined in the database, define new administrators, and view, edit, and delete existing administrators.

URL
https://<server>/unifiedconfig/config/administrator

Operations

- **create**: Creates one administrator.
- **delete**: Permanently deletes one administrator.
- **get**: Returns one administrator, using the URL https://<server>/unifiedconfig/config/administrator/<id>.
- **list**: Retrieves a list of administrators.
- **update**: Updates one administrator.

Parameters

- **refURL**: The refURL of the administrator. See Shared Parameters, on page 9.
- **changeStamp**: See Shared Parameters, on page 9.
- **description**: See Shared Parameters, on page 9.
- **userName**: Required. The unique username of an existing Active Directory account. Maximum length of 64 characters.
- **domainName**: The domain for this administrator. If blank, system uses the default domain name. Maximum length of 64 characters.
- **customer**: A reference to a customer, including the refURL.
- **readOnly**: Required. Specifies whether the administrator has read-only access to the APIs and tools. Values are true/false.
Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• userName</td>
<td>• userName</td>
</tr>
<tr>
<td>• domainName</td>
<td>• domainName</td>
</tr>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

Example Get Response

```xml
<administrator>
  <changeStamp>0</changeStamp>
  <domainName>domain</domainName>
  <userName>user1</userName>
  <readOnly>false</readOnly>
  <customer>
    <refURL>/unifiedconfig/config/customer/(id)</refURL>
  </customer>
</administrator>
```
Chapter 4

Agent API

Agents respond to contacts from customers. Use the Agent API to list the agents currently defined in the database, and view and edit existing agents.

Note

Access to this API is different for supervisors and administrators. For more information, see Access, on page 4.

URL

https://<server>/unifiedconfig/config/agent/

Operations

• get: Returns one agent, using the URL
  https://<server>/unifiedconfig/config/agent/<id>.

• list: Retrieves a list of agents.

  Query parameters:

  * selectedAttribute: Use this query parameter to augment the returned agent parameters with an extra parameter called selectedAttribute. This parameter indicates if the agent belongs to the attribute with the ID specified in this query parameter. For example, to find out which agents belong to the specified attribute, add selectedAttribute=5000.

  Note

  Using selectedAttribute automatically sets the summary list query parameter to true.

  * selectedSkillGroup: Use this query parameter to augment the returned agent parameters with an extra parameter called selectedSkillGroup. This parameter indicates if the agent belongs to the skillgroup with the ID specified in this query parameter. For example, to find out which agents belong to the specified skill group, add selectedSkillGroup=5001.

  Note

  Using selectedSkillGroup automatically sets the summary list query parameter to true.
• ignoreRole: Use this query parameter to allow a supervisor to see a list of all agents in the system, including agents that the supervisor does not supervise. For example, to see all agents, add ignoreRole=true.

• Summary list: See list, on page 4.

• update: Updates one agent.

Parameters

• refURL: The refURL for the agent. See Shared Parameters, on page 9.

• agentId: The unique peripheral number. Maximum length of 11 characters allowed. Default is an autogenerated 7-digit number.

• changeStamp: See Shared Parameters, on page 9.

• description: See Shared Parameters, on page 9.

• agentStateTrace: Indicates if agent state tracing is turned on for the agent. True or false.

• agentDeskSettings: A reference to the agent's agentDeskSettings, including the refURL and name. See References, on page 6.

• peripheral: Indicates the peripheral, including the ID and name, on which the agent resides.

• person: Required. Includes the following parameters:
  • firstName: Agent's first name. Maximum of 32 characters. International characters are allowed.
  • lastName: Agent's last name. Maximum of 32 characters. International characters are allowed.
  • password: Agent's password. Maximum of 256 ASCII characters. Password is case-sensitive. The password can be used when creating or updating, but is not returned.
    If the ssoEnabled parameter is set to true, the password is not saved.
  • userName: Agent's login name. Maximum of 255 ASCII characters. Must be unique. For supervisors and for agents with single sign-on (SSO) enabled, the username is the user's Active Directory or SSO account username.
  • loginEnabled: Whether the agent can log in. True or false. Default is true.
  • ssoEnabled: Whether single sign-on is supported at the agent level. True or false. Default is false. This parameter takes effect only when the global level SSOEnabled is mixed.

• supervisor: Required. Indicates whether the agent is marked as supervisor. True or false.
  If set to true, the person userName is used for the supervisor username and domain.

• agentAttributes: A collection of agent attribute (Attribute API, on page 27) references for this agent, including the description, refURL, name, and dataType for each associated attribute. Also includes the attributeValue parameter which indicates the value (true or false or 1-10) of the attribute for this agent. See References, on page 6.

• agentAttributesAdded: A collection of agent attribute references (Attribute API, on page 27) to be added to the agent, including the agent refURL and the attributeValue of each attribute. If the attributeValue is not specified, it is assigned the default value. Agents that already have this attribute are updated with the specified attributeValue. This parameter is update only, and cannot be used with
the agentAttributes parameter. This parameter can be used with the agentAttributesRemoved parameter. See References, on page 6.

• agentAttributesRemoved: A collection of agent attribute references (Attribute API, on page 27) to be removed from the agent, including the refURL of each attribute. This parameter is update only, and cannot be used with the agentAttributes parameter. This parameter can be used with the agentAttributesAdded parameter. See References, on page 6.

• selectedAttribute: Indicates if the agent has the specified attribute. Returned only when using the selectedAttribute query parameter. True or false.

• skillGroups: A collection of skill group references for this agent, including the refURL and name of each associated skill group. See References, on page 6.

• skillGroupsAdded: A collection of skill group references to be added to the agent, including the refURL of each skill group to be added. This parameter is update only, and cannot be used with the skillGroups parameter on an update as it does not affect existing skill groups. This parameter can be used with the skillGroupsRemoved parameter. See References, on page 6.

• skillGroupsRemoved: A collection of skill group references to be removed from the agent, including the refURL of each skill group to be removed. This parameter is update only, and cannot be used with the skillGroups parameter on an update as it does not affect existing skill groups. This parameter can be used with the skillGroupsAdded parameter. See References, on page 6.

• defaultSkillGroup: A reference to a skill group, including the refURL and name. Identifies the default skill group associated with this agent. See References, on page 6.

• selectedSkillGroup: Indicates if the agent has the specified skill group. Returned only when using the selectedSkillGroup query parameter. True or false.

• agentTeam: A reference to the agent's team, including the refURL and name. See References, on page 6.

• supervisorTeams: If this agent has supervisor access, this collection of references is for this supervisor's teams, including the refURL and name of each supervised team. See References, on page 6.

**Search and Sort Values**

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• agentId</td>
<td>• agentId</td>
</tr>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
<tr>
<td>• person.firstName</td>
<td>• supervisor</td>
</tr>
<tr>
<td>• person.lastName</td>
<td>• agentStateTrace</td>
</tr>
<tr>
<td>• person.userName</td>
<td>• person.firstName</td>
</tr>
<tr>
<td></td>
<td>• person.lastName</td>
</tr>
<tr>
<td></td>
<td>• person.userName</td>
</tr>
<tr>
<td></td>
<td>• person.loginEnabled</td>
</tr>
<tr>
<td></td>
<td>• person.ssoEnabled</td>
</tr>
</tbody>
</table>

Cisco Unified Contact Center Enterprise Developer Reference, Release 11.5(1)
See Search, on page 11 and Sort, on page 12.

Advanced search parameters

There are several advanced searches you can perform on the Agent API, including supervisor, attributes, skillgroups, team, and include and exclude (agentId).

• **supervisor:** *(true/false)* Find agents that are (or are *not*) supervisors.
  * q=supervisor:true Returns all agents who are supervisors.
  * q=supervisor:false Returns all agents who are *not* supervisors.

• **attributes:** *(attr1 & attr2 & attr3, ...)* find *all* agents that have *all* the specified attributes. Up to ten attributes can be specified. The attribute names are fully matched.

• **skillgroups:** *(skill1 & skill2 & skill3, ...)* find *all* agents that have *all* the specified skillgroups. Up to ten skillgroups can be specified. The skillgroup names are fully matched.

• **team:** *(team1|team2|team3, ...)* find *all* agents who belong to *any* of the specified teams. Up to ten team names can be specified. The team name is fully matched.

• **include:** *(ID1 & ID2 & ID3, ...)* find *all* specified agents even if they do not meet other search criteria. Each agentId is fully matched.

• **exclude:** *(ID1 & ID2 & ID3, ...)* exclude *all* specified agents from the results even if they meet all other search criteria. Each agentId is fully matched.

Example Get Response

```
<agent>
  <changeStamp>2877</changeStamp>
  <refURL>/unifiedconfig/config/agent/5017</refURL>
  <agentId>8006</agentId>
  <agentStateTrace>false</agentStateTrace>
  <peripheral>
    <id>5000</id>
    <name>CUCM_PG_1</name>
  </peripheral>
  <description>an agent</description>
  <person>
    <firstName>Agent2</firstName>
    <lastName>Agent2</lastName>
    <loginEnabled>true</loginEnabled>
    <userName>Agent2@cisco.com</userName>
    <password>mypassword</password>
    <ssoEnabled>false</ssoEnabled>
  </person>
  <agentDeskSettings>
    <name>test2</name>
    <refURL>/unifiedconfig/config/agentdesksetting/5434</refURL>
  </agentDeskSettings>
  <supervisor>true</supervisor>
  <agentAttributes>
    <agentAttribute>
      <attribute>
        <refURL>/unifiedconfig/config/attribute/5004</refURL>
        <name>Sales</name>
        <dataType>4</dataType>
        <description>Sales proficiency</description>
      </attribute>
      <attributeValue>8</attributeValue>
      <description>masters certification</description>
    </agentAttribute>
  </agentAttributes>
</agent>
```
<agentAttribute />
</agentAttributes>

<skillGroups>
  <skillGroup>
    <refURL>/unifiedconfig/config/skillgroup/5229</refURL>
    <name>Support</name>
  </skillGroup>
</skillGroups>

<defaultSkillGroup>
  <refURL>/unifiedconfig/config/skillgroup/5229</refURL>
  <name>Support</name>
</defaultSkillGroup>

<agentTeam>
  <refURL>/unifiedconfig/config/agentteam/5003</refURL>
  <name>theTeam</name>
</agentTeam>

<supervisorTeams>
  <supervisorTeam>
    <refURL>/unifiedconfig/config/agentteam/5003</refURL>
    <name>theTeam</name>
  </supervisorTeam>
  <supervisorTeam>
    <refURL>/unifiedconfig/config/agentteam/5006</refURL>
    <name>theBTeam</name>
  </supervisorTeam>
</supervisorTeams>

</agent>
Agent Team API

You can associate a set of agents to a team with a specific supervisor. The supervisor can run reports on that team and receive Supervisor Assist requests from its members.

You can use the Agent Team API to list and view the agent teams currently defined in the database.

Note

Access to this API is different for supervisors and administrators. For more information, see Access, on page 4.

URL

https://<server>/unifiedconfig/config/agentteam

Operations

- **get**: Returns one agent team, using the URL
  https://<server>/unifiedconfig/config/agentteam/<id>.

- **list**: Retrieves a list of agent teams.
  
  * Query parameters:
  
  * Summary list: See list, on page 4.

Parameters

- refURL: The refURL of the agent team. See Shared Parameters, on page 9.

- name: The name of the agent team. See Shared Parameters, on page 9.

- description: See Shared Parameters, on page 9.

- dialedNumber: A reference to an internal dialed number () for the agent team, including the refURL and dialed number string. See References, on page 6.

- agents: A collection of agent (Agent API, on page 17) references, including the refURL, first name, last name, username, and agent ID for each agent on the team. See References, on page 6.
• agentCount: Read-only field. Number of agents on the team.

• supervisors: A collection of supervisor (Agent API, on page 17) references, including the refURL, first name, last name, user name, and agent ID for each supervisor who supervises this team. See References, on page 6.

• supervisorCount: Read-only field. Number of supervisors who supervise this team.

Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name (default)</td>
</tr>
<tr>
<td>description</td>
<td>description</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

Example Get Response

```xml
<agentTeam>
  <refURL>/unifiedconfig/config/agentteam/(id)</refURL>
  <name>team1</name>
  <dialedNumber>
    <refURL>/unifiedconfig/config/dialednumber/(id)</refURL>
    <dialedNumberString>8885551212</dialedNumberString>
  </dialedNumber>
  <description>test agent team1</description>
  <agents>
    <agent>
      <refURL>/unifiedconfig/config/agent/(id_1)</refURL>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <userName>username</userName>
      <agentId>8006</agentId>
    </agent>
    <agent>
      <refURL>/unifiedconfig/config/agent/(id_2)</refURL>
      <firstName>Jane</firstName>
      <lastName>Doe</lastName>
      <userName>username</userName>
      <agentId>8007</agentId>
    </agent>
  </agents>
  <supervisors>
    <supervisor>
      <refURL>/unifiedconfig/config/agent/(id_3)</refURL>
      <firstName>Mary</firstName>
      <lastName>Hart</lastName>
      <userName>username</userName>
      <agentId>8008</agentId>
    </supervisor>
    <supervisor>
      <refURL>/unifiedconfig/config/agent/(id_4)</refURL>
      <firstName>Jack</firstName>
      <lastName>Jones</lastName>
      <userName>username</userName>
      <agentId>8009</agentId>
    </supervisor>
  </supervisors>
</agentTeam>
```
<changeStamp>0</changeStamp>
</agentTeam>
CHAPTER 6

Attribute API

Attributes identify a call routing requirement, such as language, location, or agent expertise. You can create two types of attributes: boolean or proficiency. For example, you can create a Boston attribute that specifies that the agent assigned to this attribute must be located in Boston. Then, if a precision queue requires an agent who lives in Boston, then an agent with the attributes Boston = True is a good match. When you create a proficiency attribute, you assign a proficiency level to the agent.

Use the Attribute API to list the attributes currently defined in the database, define new attributes, and view, edit, and delete existing attributes.

URL

https://<server>/unifiedconfig/config/attribute

Operations

- **create**: Creates an attribute.
- **delete**: Marks one attribute and associated Agent attribute values for deletion, but does not permanently delete them.
- **get**: Returns one attribute, using the URL
  https://<server>/unifiedconfig/config/attribute/<id>.
- **list**: Retrieves a list of attributes.

* Query parameters:

  - **selectedAgentCount**: Use this query parameter to augment attribute information about multiple agents. The selectedAgentCount parameter shows the number of specified agents associated with this attribute. For example, to find out how many of agents 5000, 5001, 5002, and 5003 in the list have this associated attribute, add
    selectedAgentCount=5000,5001,5002,5003.

  - **Summary list**: See list, on page 4.
**update**: Updates one attribute.

* **Query Parameters:**

  * **removeNonMatchingDepartmentalRefs**: Use this query parameter to remove all agent attributes from an attribute when they no longer belong to the department id specified in the query parameter or the global department. For example, to remove all agent attributes that do not belong to department 5000 or the global department, add removeNonMatchingDepartmentalRefs=5000. If this parameter is not specified, the agent attributes must belong to the attribute's department or the global department.

**Parameters**

  * **refURL**: The refURL of the attribute. See Shared Parameters, on page 9.
  * **name**: The name of the attribute. See Shared Parameters, on page 9.
  * **changeStamp**: See Shared Parameters, on page 9.
  * **description**: See Shared Parameters, on page 9.
  * **dataType**: The data type of the attribute. Values are:

    * 3: Boolean.
    * 4: Proficiency.

  * **defaultValue**: Used to specify the default value for the attribute when assigned to an agent, if no explicit value is provided. Values are:

    * Boolean: true/false.
    * Proficiency: 1-10.

  * **agentAttributes**: A collection of agent attribute references for this attribute, including the description, refURL, and read-only parameters agentId, userName, firstName, and lastName. Also includes the attributeValue parameter which indicates the value (true/false or 1-10) of the attribute for this agent. See References, on page 6.

  * **agentAttributesAdded**: A collection of agent attribute references (Attribute API, on page 27) to be added to the attribute, including the agent refURL and the attributeValue of each agent. If the attributeValue is not specified, it is assigned the default value. Agents that already have this attribute are updated with the specified attributeValue. This parameter is update only, and cannot be used in conjunction with the agentAttributes parameter. This parameter can be used with the agentAttributesRemoved parameter. See References, on page 6.

  * **agentAttributesRemoved**: A collection of agent attribute references (Attribute API, on page 27) to be removed from the attribute, including the refURL of each agent. This parameter is update only, and cannot be used in conjunction with the agentAttributes parameter. This parameter can be used with the agentAttributesAdded parameter. See References, on page 6.

  * **agentCount**: Read-only field. Number of agents associated with the attribute.

  * **selectedAgentCount**: Read-only field. Indicates the number of specified agents associated with this attribute. Returned only when using the selectedAgentCount query parameter.
Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name (default)</td>
</tr>
<tr>
<td>• description</td>
<td>• dataType</td>
</tr>
<tr>
<td></td>
<td>• defaultValue</td>
</tr>
<tr>
<td></td>
<td>• description</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

Example Get Response

```xml
<attribute>
  <changeStamp>0</changeStamp>
  <refURL>/unifiedconfig/config/attribute/5002</refURL>
  <agentCount>1</agentCount>
  <dataType>4</dataType>
  <defaultValue>1</defaultValue>
  <description>test</description>
  <name>AttributeA</name>
  <agentAttributes>
    <agentAttribute>
      <attributeValue>5</attributeValue>
      <description>test</description>
      <agent>
        <refURL>/unifiedconfig/config/agent/8467</refURL>
        <agentId>123</agentId>
        <firstName>Jane</firstName>
        <lastName>Smith</lastName>
        <userName>jsmith</userName>
      </agent>
    </agentAttribute>
  </agentAttributes>
</attribute>
```
Bucket Interval API

Configure bucket intervals to report how many calls are handled or abandoned during specific, incremental time slots. Each bucket interval has a maximum of nine configurable time slots, called Upper Bounds. Upper Bounds are ranges measured in seconds to segment and capture call-handling activity. You can run reports that show calls answered and calls abandoned for these intervals.

Use the Bucket Intervals API to add new bucket intervals, edit the name of an existing bucket interval, get a list of all of the configured bucket intervals, and delete existing bucket intervals.

**URL**

https://<server>/unifiedconfig/config/bucketinterval

**Operations**

- **create**: Creates one bucket interval.
- **delete**: Deletes one bucket interval from the database.
- **get**: Returns one bucket interval, using the URL
  https://<server>/unifiedconfig/config/bucketinterval/<id>.
- **list**: Retrieves a list of bucket intervals.
- **update**: Updates the name of one bucket interval.

**Parameters**

- **refURL**: The refURL of the bucket interval. See Shared Parameters, on page 9.
- **name**: The name of the bucket interval. See Shared Parameters, on page 9.
- **changeStamp**: See Shared Parameters, on page 9.
- **upperBound1**: Required. The first Bucket Interval value, in seconds. Must be greater than 0. This parameter cannot be updated.
- **upperBound2 to upperBound 9**: Optional. The next Bucket Interval values, in seconds. Each must be greater than the previous upperBound field or be left blank (if blank, all remaining upperBound fields must also be blank). These parameters cannot be updated.
Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name (default)</td>
</tr>
<tr>
<td></td>
<td>• upperBound 1-9</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

Example Get Response

```xml
<bucketInterval>
  <refURL>/unified/config/bucketInterval/(id)</refURL>
  <name>test</name>
    <upperBound1>10</upperBound1>
    <upperBound2>20</upperBound2>
    <upperBound3>30</upperBound3>
    <upperBound4>40</upperBound4>
    <upperBound5>50</upperBound5>
    <upperBound6>60</upperBound6>
    <upperBound7>70</upperBound7>
    <upperBound8>80</upperBound8>
    <upperBound9>90</upperBound9>
  <changeStamp>0</changeStamp>
</bucketInterval>
```
Bulk Job API

Bulk jobs are a fast and efficient way to update existing agent and supervisor usernames to SSO usernames. You can use the Bulk Job API to return a CSV template with records for agents and supervisors who are not enabled for SSO. You can also use the Bulk Job API to list the bulk jobs currently defined in the database, define new bulk jobs, and view or delete records of existing bulk jobs.

URL

https://<server>/unifiedconfig/config/bulkjob

Operations

- **create**: Creates one bulk job.
- **delete**: Permanently deletes one bulk job.
- **get**: Returns one bulk job using the URL
  https://<server>/unifiedconfig/config/bulkjob/<id>.
- **get** (template): Returns the ssomigration CSV template as a file attachment. The template includes the first 12,000 agent and supervisor accounts that are not enabled for SSO. The form of the request is
  https://<server>/unifiedconfig/config/bulkjob/templates/ssomigration.

  If the database contains more than 12,000 non-SSO users, get the first 12,000 and update their credentials for SSO. Then get the next 12,000 users and update their credentials for SSO. Repeat until there are no more agents to migrate.

  Non-SSO users can be filtered by peripheralId, agentTeamId, or both.

  To filter by peripheralId, use the following URL:
  https://<server>/unifiedconfig/config/bulkjob/templates/ssomigration?q=peripheralId:<peripheralId>.

  To filter by agentTeamId, use the following URL:
  https://<server>/unifiedconfig/config/bulkjob/templates/ssomigration?q=agentTeamId:<agentTeamId>.

  To filter by both peripheralId and agentTeamId, use the following URL:
  https://<server>/unifiedconfig/config/bulkjob/templates/ssomigration?q=peripheralId:<peripheralId> agentTeamId:<agentTeamId>.

- **list**: Retrieves a list of bulk jobs.
Parameters

- refURL: The refURL of the bulk job. See Shared Parameters, on page 9.
- description: See Shared Parameters, on page 9.
- fileContent: The content of the bulk CSV file. The size of the file must not exceed 3MB.
- createDateTime: The time the bulk job was submitted. It indicates the time in milliseconds elapsed from the zero epoch value of January 1, 1970, 00:00:00 GMT. Read-only.
- jobHostName: The Windows computer name of the AW that initiated the bulk job. Read-only.
- startTime: The time the bulk job began executing. Read-only.
- endTime: The time the bulk job completed or failed. Read-only.
- jobState: The current state of the job. Read-only.
  - 1: Queued
  - 2: Processing
  - 3: Succeeded
  - 4: Failed
  - 5: Cancelled
  - 6: Partially succeeded
- jobType: The job type. Optional. If this parameter is not provided, the value is determined automatically based on the header in the bulk job file. If the parameter is provided, the header must match the job type specified.
  The only valid value is 5, for SSO Migration.
- description: See Shared Parameters, on page 9.
- logFile: A URL to download the log file for the bulk job. Read-only.
- csvFile: A URL to download the CSV file that was originally uploaded in the fileContent parameter. Read-only.

Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.
### Bulk Job API

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
<tr>
<td></td>
<td>• jobType</td>
</tr>
<tr>
<td></td>
<td>• jobState</td>
</tr>
<tr>
<td></td>
<td>• jobHostName</td>
</tr>
<tr>
<td></td>
<td>• createDateTime</td>
</tr>
<tr>
<td></td>
<td>• startDateTime</td>
</tr>
<tr>
<td></td>
<td>• endDateTime</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

#### Example Get Response

Example response for a SSO migration create job that completed successfully:

```xml
<bulkJob>
  <changeStamp>2</changeStamp>
  <refURL>/unifiedconfig/config/bulkjob/(id)</refURL>
  <jobHostName>CCE_AW</jobHostName>
  <createDateTime>1461785074000</createDateTime>
  <startDateTime>1461785074000</startDateTime>
  <endDateTime>1461785074000</endDateTime>
  <jobState>3</jobState>
  <jobType>5</jobType>
  <description>sso migration bulk job</description>
  <logFile>
    <refURL>/unifiedconfig/config/bulkjob/(id)/log</refURL>
  </logFile>
  <csvFile>
    <refURL>/unifiedconfig/config/bulkjob/(id)/csv</refURL>
  </csvFile>
</bulkJob>
```

**Cisco Unified Contact Center Enterprise Developer Reference, Release 11.5(1)**
CHAPTER 9

Congestion Control API

Congestion control parameters determine how calls are treated by the system when too many calls are received at one time. Use the Congestion Control API to list or edit the current congestion control parameters in the database.

URL

https://<server>/unifiedconfig/config/congestioncontrol

Operations

• get: Returns the congestion control parameters, using the URL
  https://<server>/unifiedconfig/config/congestioncontrol.

• update: Updates the congestion control parameters.

Parameters

• deploymentType: The type of deployment. See Deployment Type Info API, on page 49.

• congestionEnabled: Indicates if congestion control is enabled. Value is true/false.

• congestionTreatmentMode: Mode to handle congestion. Values are:
  • 1: Dialed Number default label is used for call treatment.
  • 2: Treat call with Routing client default label.
  • 3: Treat call with System default label.
  • 4: Terminate with Dialog Fail/RouteEnd.
  • 5: Release message to the Routing client.

• systemDefaultLabel: Default label string to treat the calls subjected to congestion control. Only used if congestionTreatmentMode is set to 3 (Treat call with System default label).

• cpsCapacity: The maximum number of calls per second allowed.

• cpsCapacityDefault: The default value for the cpsCapacity parameter for the current deployment type. Read-only.
Example Get Response

```xml
<congestionControl>
  <deploymentType>0</deploymentType>
  <congestionTreatmentMode>1</congestionTreatmentMode>
  <congestionEnabled>true</congestionEnabled>
  <systemDefaultLabel></systemDefaultLabel>
  <cpsCapacity>100</cpsCapacity>
  <cpsCapacityDefault>150</cpsCapacityDefault>
</congestionControl>
```
CHAPTER 10

Contact Share Group API

The Contact Share group is a routing entity that operates on a single Contact Share rule with a set of Contact Share queues. It is used to determine to which target Unified CCE to send the call, given the definition of the rule and the realtime data on each of the queues. This API is only available on the Contact Director Deployment.

Note

To learn more about the other Contact Share APIs available in the Contact Director Deployment, see Access, on page 4.

URL

https://<server>/unifiedconfig/config/contactsharegroup

Parameters

- changeStamp: See Shared Parameters, on page 9.
- refURL: The refURL of the Contact Share group. See Shared Parameters, on page 9.
- contactShareRule: Required. The refURL of a Contact Share rule to apply to this group.
- name: Required. The name of the Contact Share group.
- acceptQueueIf: Optional. A logical expression that qualifies individual queues to be used in the group.
- description: Optional. The description of the group.
- contactShareQueues: Optional. Information for any Queues in the Contact Share group.

Operations

- create: Creates one Contact Share group.
- delete: Deletes one Contact Share group.
- get: Returns one Contact Share group, using the URL https://<server>/unifiedconfig/config/contactsharegroup<id>.
- list: Retrieves a list of Contact Share group.
• **update**: Updates one Contact Share group.

**Search and Sort Values**

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name (default)</td>
</tr>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
<tr>
<td>• groupExpression</td>
<td>• groupExpression</td>
</tr>
</tbody>
</table>

See **Search**, on page 11 and **Sort**, on page 12.

**Example Get Response**

```xml
<contactShareGroup>
  <changeStamp>0</changeStamp>
  <refURL>/unifiedconfig/config/contactsharegroup/5000</refURL>
  <contactShareRule>
    <name>Cisco_MED</name>
    <refURL>/unifiedconfig/config/contactsharerule/5000</refURL>
  </contactShareRule>
  <name>test</name>
  <acceptQueueIf>Call.PeripheralVariable1==1</acceptQueueIf>
  <description>test something</description>
  <contactShareQueues>
    <contactShareQueue>
      <refURL>/unifiedconfig/config/contactsharequeue/5000</refURL>
      <queueType>skillGroup</queueType>
      <targetInstance>
        <id>5000</id>
        <name>cs01</name>
      </targetInstance>
    </contactShareQueue>
  </contactShareQueues>
</contactShareGroup>
```
Contact Share Rule API

A Contact Share rule is required when configuring a Contact Share group. A group is sent in every routing request to the Contact Share process. Each group has a rule that defines the logic for selecting a skill group or precision queue in that group for a routing request. Use the Contact Share Rules API to create, update, list, and delete rules. This API is only available on the Contact Director deployment.

Note

To learn more about the other APIs available in the Contact Director deployment, see Access, on page 4.

URL

https://<server>:/unifiedconfig/config/contactsharerule

Parameters

- refURL: The refURL of the Contact Share rule. See Shared Parameters, on page 9.
- name: Required. The name of the Contact Share rule.
- ruleExpression: Required. The logical expression with which the rule operates.
- description: Optional. The description of the rule.
- changeStamp: See Shared Parameters, on page 9.

Operations

- create: Creates one Contact Share rule.
- delete: Permanently deletes one Contact Share rule.
- get: Returns one Contact Share rule, using the URL https://<server>:/unifiedconfig/config/contactsharerule<id>.
- list: Retrieves a list of Contact Share rules.
- update: Updates one Contact Share rule.
**Search and Sort Values**

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name (default)</td>
</tr>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
<tr>
<td>• ruleExpression</td>
<td>• ruleExpression</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

**Example Get Response**

```xml
<contactShareRule>
  <refURL>/unified/config/contactsharerule/(id)</refURL>
  <name>test</name>
  <ruleExpression>Call.PeripheralVariable==1</ruleExpression>
  <description>test something</description>
  <changeStamp>0</changeStamp>
</contactShareRule>
```
**Context Service Configuration API**

Use the Context Service Configuration API to configure Context Service settings for SocialMiner.

**URL**

https://<server>/unifiedconfig/config/contextservice

**Operations**

- **get**: Gets the Context Service configuration data, using the URL
  https://<server>/unifiedconfig/config/contextservice.

- **update**: Updates the Context Service configuration data.

**Parameters**

- **proxyUrl**: URL of the proxy server. The default is an empty string. Optional.
- **maxRetries**: The maximum number of times that the system attempts the operation. This parameter must be set to 1.
- **timeout**: The amount of time, in milliseconds, that the system waits before abandoning the attempt to perform the operation. The default is 1200 ms. The values are 200 - 15000 ms. Optional.
- **labMode**: Whether the Context Service is in lab mode. The default is false. Optional.

**Example Get Response**

```xml
<contextServiceConfig>
  <proxyUrl>http://example.com:80</proxyUrl>
  <maxRetries>1</maxRetries>
  <timeout>10000</timeout>
  <labMode>false</labMode>
</contextServiceConfig>
```
Context Service Registration API

Cisco Context Service is a cloud-based omnichannel solution for Cisco Contact Center Enterprise Solutions. It enables you to capture your customer’s interaction history by providing flexible storage of customer-interaction data across any channel.

Various components in the CCE Solution provide out of the box integration with Context Service. Context Service also provides an API for integration with your own applications or third-party applications to capture end-to-end customer-interaction data.

For more information about Context Service and to check service availability, see http://cisco.com/go/contextservice.

Use the Context Service Registration API to register SocialMiner with Context Service and get the current registration status.

**URL**

https://<server>/unifiedconfig/config/contextserviceregistration

**Operations**

- **get**: Returns the current Context Service registration state using the URL
  

- **update**: Updates Context Service connection data in SocialMiner and the CCE database.

**Parameters**

- **callbackUrl**: This parameter is optional in the Get URL. The browser is redirected to this URL to complete the registration or deregistration process.

- **state**: The Context Service registration state. Values are as follows:
  
  *notRegistered*
  
  *registered*
  
  *invalid*: The connection data used to register with Context Service is no longer valid. Reregister to recover from this state.
• managementUrl: Navigate to this URL to manage the Context Service. This parameter is returned if the state is registered.

• registrationUrl: Navigate to this URL to begin Context Service registration. This parameter is returned if you include the callbackUrl in the Get URL and the state is either notRegistered or invalid.

• deregistrationUrl: Navigate to this URL to begin Context Service deregistration. This parameter is returned if you include the callbackUrl in the Get URL and the state is registered.

• refURL: The refURL. See Shared Parameters, on page 9.

• connectionData: This parameter is required and the maximum size is 4 KB. connectionData is an opaque string used to connect to the Context Service. Use an empty string to clear the connectionData when deregistering.

Note You must deregister before reregistering to Context Service. If the current connectionData is not empty, you can set the value either to an empty string or to the current value.

Sample API Workflow

1 The user navigates to a custom Context Service web page.
2 The page calls the Get API to get the current registration status from Context Service. The page displays either a Register or Deregister button.
3 The user clicks Register or Deregister.
4 The page calls the Get API with a callback URL for a custom web page. This page can be the custom Context Service web page from step 1. The API returns either the Context Service registration or deregistration URL.
5 The browser redirects to the Context Service registration or deregistration URL.
6 The user completes the registration or deregistration wizard.
7 The browser redirects back to the callback URL with a connectionData query parameter appended to it.
8 The callback page calls the Update API with the connection data from step 7 to send the data to SocialMiner and the CCE database.
9 Go to step 2.

Example Get Response

```xml
<contextServiceRegistration>
  <state>registered</state>
  <managementUrl>(url)</managementUrl> <!-- Included if state is registered -->
  <registrationUrl>(url)</registrationUrl> <!-- Included if state is notRegistered and callbackUrl is provided -->
  <deregistrationUrl>(url)</deregistrationUrl> <!-- Included if state is registered and callbackUrl is provided -->
</contextServiceRegistration>
```

Example Update Request

```xml
<contextServiceRegistration>
  <connectionData>(data)</connectionData>
</contextServiceRegistration>
```
Deployment API

The Deployment API is used to view the deployment type of the installation. It is read-only, and does not require authentication. To change the deployment type, use the Deployment Type Info API.

**URL**

https://<server>/unifiedconfig/config/deployment

**Parameters**

- deploymentType: The type of deployment. See Deployment Type Info API, on page 49.
- supervisorLoginAllowed: Indicates whether the current deployment type allows supervisor login.

**Operations**

- **get**: Returns the deployment type of the installation using the URL
  https://<server>/unifiedconfig/config/deployment.

**Example Get Response**

```xml
<deployment>
  <deploymentType>7</deploymentType>
  <supervisorLoginAllowed>true</supervisorLoginAllowed>
</deployment>
```
Deployment Type Info API

Use the Deployment Type Info API to view or edit the current system deployment type.

URL
https://<server>/unifiedconfig/config/deploymenttypeinfo

Operations

• **get**: Returns the current deployment type and the results of the capacity and system validation tests, using the URL https://<server>/unifiedconfig/config/deploymenttypeinfo.

• **update**: Sets the specified deployment type if the system validation check, capacity check, and VM Validation for that deployment type pass and are required.

Parameters

• `changeStamp`: See Shared Parameters, on page 9.

• `vmHosts`: vmHost information, including name, address, username, and password parameters of Side A and Side B. Only required when switching to Packaged CCE, to allow access to the ESX servers for VM validation.

• `permissionInfo`: See Permissions, on page 10.

• `systemValidationStatus`: A collection of validationRules that show the potential errors regarding system configuration. Each rule contains the following parameters:

  • name: The name of the rule.
  • isValid: Indicates if the rule is passing. Values are true/false.
  • min: The minimum number of items required to match for this rule.
  • max: The maximum number of items required to match for this rule.
  • actual: The current number of items configured that match this rule.

• `capacityInfo`: A collection of capacityRules indicating if the capacity limits are valid. Each rule contains the following parameters:

  • name: The name of the capacity rule.
• max: The maximum number of items allowed for the rule.
• actual: The current number of items configured for the rule.

• vmValidationLogURL: The URL to download a file about VM layout validation.

• deploymentType: The type of deployment. The following types are supported:
  * 0: No deployment type specified. Initial type set at installation. Once set to another deployment type, you cannot switch back to 0.
  * 1: NAM (Deprecated)
  * 2: Contact Director
  * 3: NAM Rogger (Deprecated)
  * 4: ICM Router/Logger
  * 5: UCCE: 8000 Agents Router/Logger
  * 6: UCCE: 12000 Agents Router/Logger
  * 7: Packaged CCE: 2000 Agents
  * 8: ICM Rogger
  * 9: UCCE: 4000 Agents Rogger
  * 10: Packaged CCE: Lab Mode
  * 11: HCS-CC: 2000 Agents
  * 12: HCS-CC: 500 Agents (Deprecated)
  * 13: UCCE: Progger (Lab Only)
  * 14: HCS-CC: 4000 Agents
  * 15: HCS-CC: 12000 Agents

Note

Example Get Response
<deploymentTypeInfo>
<changeStamp>59</changeStamp>
<deploymentType>7</deploymentType>
<vmHosts>
<vmHost>
<name>sideA</name>
<address>10.86.141.10</address>
<userName>root</userName>
</vmHost>
<vmHost>
<name>sideB</name>
<address>10.86.141.29</address>
</vmHost>
</vmHosts>
<userName>root</userName>
  <password>pwexample</password>
</vmHost>
</vmHosts>
</deploymentTypeInfo>
Internet Script Editor API

The Internet Script Editor API indicates whether Internet Script Editor is enabled. If Internet Script Editor is enabled in Web Setup, the API displays the download link in the format https://<server>/install/iScriptEditor.exe.

URL
/internetscripteditor

Operations
• get: Returns whether Internet Script Editor is enabled and a download link.

Parameters
• enabled: Indicates whether Internet Script Editor is enabled in Web Setup. True or false.
• downloadLink: The download link for Internet Script Editor. This link appears only when the enabled parameter is true.

Example Get Response

<internetScriptEditor>
  <enabled>true</enabled>
  <downloadLink>https://10.10.10.207/install/iScriptEditor.exe</downloadLink>
</internetScriptEditor>
Machine Inventory API

This API returns the machines in the solution.

**URL**

https://<server>/unifiedconfig/config/machineinventory

**Operations**

- **create**: Creates a machine by updating the database. See the following table for restrictions per machine type.
- **delete**: Removes one machine.
- **get**: Returns one machine and all associated addresses and services based on the machine ID, using the URL https://<server>/unifiedconfig/config/machineinventory/<id>.
- **list**: Retrieves a list of all machines in the inventory. See the following table for more details.
- **update**: Updates one machine.

<table>
<thead>
<tr>
<th>Type</th>
<th>Create / Update / Delete operations allowed</th>
<th>Number allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCE_ROUTER</td>
<td>No</td>
<td>0/1 Side A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/1 Side B</td>
</tr>
<tr>
<td>CCE_PROGGER</td>
<td>No</td>
<td>0/1 Side A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/1 Side B</td>
</tr>
<tr>
<td>CCE_CALL_SERVER</td>
<td>No</td>
<td>0/1 Side A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/1 Side B</td>
</tr>
<tr>
<td>CCE_PG</td>
<td>No</td>
<td>0/150 Side A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/150 Side B</td>
</tr>
</tbody>
</table>

Cisco Unified Contact Center Enterprise Developer Reference, Release 11.5(1)
<table>
<thead>
<tr>
<th>Type</th>
<th>Create / Update / Delete operations allowed</th>
<th>Number allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIVE_DATA</td>
<td>All</td>
<td>0/1 Side A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/1 Side B</td>
</tr>
<tr>
<td>PRIMARY_AW</td>
<td>All</td>
<td>0/2 Primary</td>
</tr>
<tr>
<td>SECONDARY_AW</td>
<td>All</td>
<td>0/2 Secondary</td>
</tr>
<tr>
<td>CCE_AW</td>
<td>Update only</td>
<td>0/8 Primary</td>
</tr>
<tr>
<td>IDS_PUBLISHER</td>
<td>All</td>
<td>0 or more</td>
</tr>
<tr>
<td>IDS_SUBSCRIBER</td>
<td>No</td>
<td>0 or more</td>
</tr>
<tr>
<td>FINESSE_PRIMARY</td>
<td>All</td>
<td>0 or more</td>
</tr>
<tr>
<td>FINESSE_SECONDARY</td>
<td>Update and Delete only</td>
<td>0 or more</td>
</tr>
<tr>
<td>CUIC_PUBLISHER</td>
<td>All</td>
<td>0/1</td>
</tr>
<tr>
<td>CUIC_SUBSCRIBER</td>
<td>No</td>
<td>0/1</td>
</tr>
<tr>
<td>CUIC_PUBLISHER_STANDALONE</td>
<td>All</td>
<td>0 or more</td>
</tr>
<tr>
<td>CUIC_SUBSCRIBER_STANDALONE</td>
<td>Update and Delete only</td>
<td>0 or more</td>
</tr>
<tr>
<td>EXTERNAL_SOCIAL_MINER</td>
<td>All</td>
<td>0/1</td>
</tr>
</tbody>
</table>
When you create a CUIC_PUBLISHER, CUIC_PUBLISHER_STANDALONE, or FINESSE_PRIMARY machine, any other machine in the cluster (CUIC_SUBSCRIBER, CUIC_SUBSCRIBER_STANDALONE, or FINESSE_SECONDARY) is automatically added to the inventory. When you delete a CUIC_PUBLISHER, CUIC_PUBLISHER_STANDALONE, or FINESSE_PRIMARY machine, any other machines in the cluster are also removed from the inventory.

CUIC_PUBLISHER and CUIC_SUBSCRIBER have Unified Intelligence Center, Live Data, and Cisco Identity Service installed on the same VM. These machines can only be used in the following deployment types:

- UCCE: 4000 Agents Rigger (when used in the 2000 agent reference design)
- UCCE: Progger (Lab Only)
- HCS-CC: 500 Agents (Deprecated)
- HCS-CC: 2000 Agents

If you create a CUIC_PUBLISHER on a system that already contains a LIVE_DATA machine, the LIVE_DATA machine is deleted. If the system already contains a CUIC_PUBLISHER_STANDALONE or IDS_PUBLISHER machine, you receive an error message of error type INVENTORY_ONLY_ONE_CUIC_PUB_ALLOWED.

### Parameters

#### Machine parameters:

- refURL: The refURL of the machine. See Shared Parameters, on page 9.
- name: External name of the machine. For example, the VM hostname. Valid characters are period (.), hyphen (-), underscore (_), and alphanumeric. The first character must be alphanumeric. Maximum length is 128 characters.
- changeStamp: See Shared Parameters, on page 9.
- type: The type of machine.
- autogenerated: Indicates if the information was generated automatically.
- hostName: The hostname of the machine.
  - If you provide a hostname, a lookup is performed to find the FQDN for that host. If the FQDN is found, the hostName field is then set to that FQDN value and the Machine Address is set to the IP address corresponding to the hostname.
  - If you do not provide a hostname, a lookup is performed using the address field in the API request. If the FQDN is found, the hostName field is set to that FQDN value.
- networks: A collection of network parameters. See the network parameters below.

#### Network parameters:

- type: Must be public.
• versionInfo: Version info available depending on the product type. Includes the following parameters (by product type):
  ◦ UCCE: version, buildNumber, esNumber, patchVersion
  ◦ VOS: version and buildNumber

• address: The IP address. Must be valid hostname, IPv4, or IPv6 address.

• services: A collection of service parameters. See the services parameters below.

Services parameters:

• uri: The service URL, for example MR_PG_A URL.
  For MR_PG_A and MR_PG_B, the machines on which the MR PGs reside must be configured in the Machine Inventory:
  ◦ Most Unified CCE and HCS for Contact Center deployments use the CCE_PGs.
  ◦ Progger deployments use the CCE_PROGGERs.
  ◦ HCS500 and HCS1000 deployments use the CCE_CALL_SERVERs.

  The URLs must be different, and must follow the pattern
  /unifiedconfig/config/machineinventory/<id>.

• name: The hostname of service. This read-only parameter applies only to the MR_PG_A and MR_PG_B services. It is returned if the uri matches the unifiedconfig machine host (for example
  /unifiedconfig/config/machineinventory/1234 where 1234 is a valid id).

• port: The port for this service.
  For MR_PG_A and MR_PG_B, the default port is 38001, and the valid range is 10000 to 65535. If you do not specify a value or specify 0, the default port is used. MR_PG_A and MR_PG_B must use the same port.

• username: The username used to access the service. Username maximum is 128 characters.

• password: The password used to access the service. The password can be used when creating or updating, but is not returned.

• description: The description of the service. See Shared Parameters, on page 9.

• pairing: Indicates if services on different machines are related. Related services have a matching value.
  For Contact Share node, if the service type is ACTIVE_MQ, the pairing value contains the valid ApplicationGatewayID.
  For the service type PRINCIPAL_AW, the pairing value is "true" for the AW that manages credentials for the Context Service, and "false" for all other AWs.

• type: The service type. Values for type are as follows:
  ◦ LIVE_DATA_API: Connection information for the Live Data Rest API. Requires a username and password.
  ◦ LIVE_DATA_CASSANDRA: Connection information for the Live Data Cassandra database.
  ◦ LIVE_DATA_WEB_SOCKETS: Connection information for the Live Data Socket.IO service.
  ◦ TIP_PG: LiveData connection information for peripheral gateways.
- TIP_ROUTER: LiveData connection information for the router.
- TIP_PG_TOS: LiveData test-other-side connection information for peripheral gateways.
- TIP_ROUTER_TOS: LiveData test-other-side connection information for the router.
- ACTIVE_MQ: LiveData Active MQ connection information.
- STORM_DRPC: LiveData Storm DRPC connection information.
- AW_REST_API: Administration & Data Server (AW) Rest API
- SM_REST_API: SocialMiner REST API. Requires a username and password.
- MR_PG_A: Connection information for Side A MR PG. Required for SocialMiner machines.
- MR_PG_B: Connection information for the Side B MR PG. Required for SocialMiner machines.
- PRINCIPAL_AW: Indicates the AW that manages credentials for the Context Service.
- CONTEXT_SERVICE: Contains the last-written Context Service connection data.
- MR_PG_CONNECTION: Connection information for the MR PG. Required for SocialMiner machines.
- IDS: Indicates an Identity Server service. IDS contains the following parameters:
  - userName: The username of the IdS for the IDS_PUBLISHER machine.
  - components: Information about the component, including the component refURL and component name.
- IDSPRIMARY_REF: When present, the <uri> parameter for this service refers to the primary Identity Server (IdS) to which this component is associated.
  When you update the IDSPRIMARY_REF service type of a machine, the IDSSECONDARY_REF is automatically set to the other IdS in the cluster, if present.
- IDSSECONDARY_REF: Read only. The <uri> parameter for this service refers to the backup IdS to which this component is associated. For a create or update request, this field is automatically filled with the refURL of the secondary IdS (if present).
- PUBLISHER_REF: The <uri> parameter for this service refers to the publisher/primary machine's IP Address. This service is only applicable for subscriber/secondary machines.

### Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name (default)</td>
</tr>
<tr>
<td>description</td>
<td>description</td>
</tr>
<tr>
<td>hostName</td>
<td></td>
</tr>
</tbody>
</table>
See Search, on page 11 and Sort, on page 12.

**Advanced search parameters**

You can perform a machine type search or a service type search on the Machine Inventory API:

- **type:<machine type>**: Returns only machines of the specified type. For example, `type:EXTERNAL_SOCIAL_MINER` returns only external SocialMiner machines. The machine type is case-insensitive.

- **serviceType:<service type>**: Returns only machines associated with the specified service type. For example, `serviceType:IDS_PRIMARY_REF` returns only SSO-capable machines.

**Example Update Request**

```
https://<server>/unifiedconfig/config/machineinventory

<machine>
  <name>myLiveDataMachine</name>
  <hostName>liveDataHostName</hostName>
  <type>LIVE_DATA</type>
  <networks>
    <network>
      <type>PUBLIC</type>
      <address>1.2.3.4</address>
      <services>
        <service>
          <type>ACTIVE_MQ</type>
          <port>61616</port>
          <pairing>1</pairing>
        </service>
        <service>
          <type>STORM_DRPC</type>
          <port>3772</port>
          <pairing>1</pairing>
        </service>
        <service>
          <type>LIVE_DATA_WEB_SOCKETS</type>
          <port>12008</port>
          <pairing>1</pairing>
        </service>
        <service>
          <type>LIVE_DATA_API</type>
          <port>12005</port>
          <pairing>1</pairing>
          <userName>user</userName>
          <password>password</password>
        </service>
        <service>
          <type>LIVE_DATA_CASSANDRA</type>
          <port>12000</port>
          <pairing>1</pairing>
        </service>
      </services>
    </network>
  </networks>
</machine>
```

**Example Get Response**

```
https://<server>/unifiedconfig/config/machineinventory/<id>

<machine>
  <changeStamp>3</changeStamp>
  <refURL>/unifiedconfig/config/machineinventory/5000</refURL>
  <networks>
    <network>
      <address>1.2.3.4</address>
    </network>
  </networks>
</machine>
```
Example Get Response for IdS

Example Update Request for Primary IdS

This example request updates the primary IdS for Finesse:
Example Updating SSO-Capable Components Associated to an IdS

SSO-capable components can be associated to an IdS with the <components> element. For example, a PUT request like this updates the IDS_PRIMARY_REF and IDS_SECONDARY_REF of any SSO-capable components.

This operation also deassociates any existing component references to that IdS that aren’t in the list. For example, if there were three components associated to this IdS before the operation, only two components would be associated after the operation was complete.

```xml
<machine>
  <changeStamp>1</changeStamp>
  <refURL>/unifiedconfig/config/machineinventory/5000</refURL>
  <networks>
    <network>
      <type>PUBLIC</type>
      <address>10.10.10.22</address>
      <services>
        <service>
          <type>IDS</type>
          <components>
            <component>
              <refURL>/unifiedconfig/config/machineinventory/6000</refURL>
            </component>
            <component>
              <refURL>/unifiedconfig/config/machineinventory/6001</refURL>
            </component>
          </components>
        </service>
      </services>
    </network>
  </networks>
</machine>
```
Media Routing Domain API

A media routing domain is a collection of skill groups associated with a common media class. It is used to organize how requests for different media are routed.

Use the Media Routing Domain (MRD) API to list the MRDs currently defined in the database, define new MRDs, and view, edit, and delete existing MRDs.

The built-in Cisco_Voice MRD and legacyMultichannel MRDs are read-only; they cannot be created, updated, or deleted. You can perform all API operations on multichannel MRDs.

URL

https://<server>/unifiedconfig/config/mediaroutingdomain

Operations

- **create**: Creates an MRD.
- **delete**: Permanently deletes one MRD.
- **list**: Retrieves a list of MRDs.
- **get**: Returns one MRD using the URL
  https://<server>/unifiedconfig/config/mediaroutingdomain/<id>.
- **update**: Updates one MRD.

Parameters

- **refURL**: The refURL of the MRD. See Shared Parameters, on page 9.
- **name**: Name of the MRD. See Shared Parameters, on page 9.
- **description**: See Shared Parameters, on page 9.
- **id**: The database id of the MRD. Read-only field.
- **type**: The type of MRD. Values are as follows:
  - **voice**: Used only for the built-in Cisco_Voice MRD. These MRDs are read-only.
  - **legacyMultichannel**: Used for MRDS for the Enterprise Chat and Email application. These MRDs are read-only.
multichannel: (Default) Used for MRDs for Task Routing APIs.

- taskLife: If the connection goes down, the amount of time, in seconds, that the system waits before ending all tasks. Default is 1200.
- taskStartTimeout: The amount of time, in seconds, that the system waits between an agent being selected for a task and an agent being offered or beginning the task. When this time is reached, the system makes the agent Not Routable. Default is 30.
- maxTaskDuration: The maximum duration for a task, in seconds. Default is 28800.
- serviceLevelThreshold: Maximum time in seconds that a customer should wait before being connected with an agent. Default is 30.
- interruptible: Indicates if an agent can be interrupted by assigned tasks from another MRD. Values are true/false.
- maxTasksInQueue: The maximum number of tasks allowed to be queued at one time.
- maxTimeInQueue: The maximum amount of time, in seconds, a task can be queued.

**Search and Sort Values**

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name (default)</td>
</tr>
<tr>
<td>description</td>
<td>description</td>
</tr>
<tr>
<td>id</td>
<td>id</td>
</tr>
<tr>
<td></td>
<td>interruptible</td>
</tr>
<tr>
<td></td>
<td>maxTasksInQueue</td>
</tr>
<tr>
<td></td>
<td>maxTaskDuration</td>
</tr>
<tr>
<td></td>
<td>maxTimeInQueue</td>
</tr>
<tr>
<td></td>
<td>serviceLevelThreshold</td>
</tr>
<tr>
<td></td>
<td>taskLife</td>
</tr>
<tr>
<td></td>
<td>taskStartTimeout</td>
</tr>
</tbody>
</table>

See [Search, on page 11](#) and [Sort, on page 12](#).

**Advanced search parameters**

You can perform a nonVoiceOnly search on the Media Routing Domain API:

- **nonVoiceOnly**: Set this attribute to true in the search query parameter to make the API return only media routing domains other than the Cisco_Voice MRD. For example, q=nonVoiceOnly:true.
Example Get Response

<mediaRoutingDomains>
  <mediaRoutingDomain>
    <changeStamp>0</changeStamp>
    <refURL>/unifiedconfig/config/mediaroutingdomain/5001</refURL>
    <description>Media channel for routing Chat tasks</description>
    <id>5001</id>
    <type>multichannel</type>
    <interruptible>false</interruptible>
    <taskLife>1200</taskLife>
    <taskStartTimeout>30</taskStartTimeout>
    <maxTaskDuration>28800</maxTaskDuration>
    <maxTasksInQueue>1000</maxTasksInQueue>
    <maxTimeInQueue>1000</maxTimeInQueue>
    <name>Chat_Task_MR</name>
    <serviceLevelThreshold>30</serviceLevelThreshold>
  </mediaRoutingDomain>
  <mediaRoutingDomain>
    <changeStamp>0</changeStamp>
    <refURL>/unifiedconfig/config/mediaroutingdomain/1</refURL>
    <description>Default Media Routing Domain for Cisco_Voice</description>
    <id>1</id>
    <type>voice</type>
    <interruptible>false</interruptible>
    <taskLife>1200</taskLife>
    <taskStartTimeout>30</taskStartTimeout>
    <maxTaskDuration>28800</maxTaskDuration>
    <maxTimeInQueue>1000</maxTimeInQueue>
    <name>Cisco_Voice</name>
    <serviceLevelThreshold>30</serviceLevelThreshold>
  </mediaRoutingDomain>
</mediaRoutingDomains>
Operation API

Use the Operation API to save changes to several items of the same type in a single request. The following changes are allowed in an operation:

- delete: Multiple items of the same type. Any item that supports the delete operation can be deleted using the Operation API.
- Agent update: Update multiple agents (only available to supervisors).

URL

https://<server>/unifiedconfig/config/operation

HTTP Method

Use HTTP POST to submit a request to the Operation API.

Parameters

- operationType: Indicates if the items specified in the refURLs should be updated or deleted. Values are update/delete.
- refURLs: A collection of refURL parameters indicating which items are included in the request. See Shared Parameters, on page 9.
- changeset: Includes the parameters that are changed in an update operation. See Agent API, on page 17.
  - skillGroupsAdded
  - skillGroupsRemoved
  - agentAttributesAdded
  - agentAttributesRemoved

Example Delete Request

<operation>
  <operationType>delete</operationType>
  <refURLs>
    <refURL>/unifiedconfig/config/calltype/5000</refURL>
  </refURLs>
</operation>
Example Update Request

<operation>
  <operationType>update</operationType>
  <refURLs>
    <refURL>/unifiedconfig/config/agent/5000</refURL>
    <refURL>/unifiedconfig/config/agent/5001</refURL>
  </refURLs>
  <changeSet>
    <agent>
      <skillGroupsAdded>
        <skillGroup>
          <refURL>/unifiedconfig/config/skillgroup/6000</refURL>
        </skillGroup>
      </skillGroupsAdded>
      <skillGroupsRemoved>
        <skillGroup>
          <refURL>/unifiedconfig/config/skillgroup/6001</refURL>
        </skillGroup>
      </skillGroupsRemoved>
      <agentAttributesAdded>
        <agentAttribute>
          <attribute>
            <refURL>/unifiedconfig/config/attribute/5004</refURL>
          </attribute>
          <attributeValue>8</attributeValue>
        </agentAttribute>
      </agentAttributesAdded>
      <agentAttributesRemoved>
        <agentAttribute>
          <attribute>
            <refURL>/unifiedconfig/config/attribute/5005</refURL>
          </attribute>
        </agentAttribute>
      </agentAttributesRemoved>
    </agent>
  </changeSet>
</operation>

Response Parameters

- status: Indicates the state of the operation.
  * success: The operation succeeded for all items.
  * partialSuccess: The operation succeeded for some items, but other items had errors.
  * failure: The operation failed for all items.

- apiErrors: Errors indicate which items had errors and the cause of the error.

Example Success Response

The following example shows the response when the delete operation is successful:

<operationsResult>
  <status>success</status>
</operationsResult>
Example Partial Success Message

The following example shows a partial success response for a request to delete several agents:

```xml
<operationsResult>
  <apiErrors>
    <apiError>
      <errorDetail xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="resourceErrorDetail">
        <refURL>agent/1</refURL>
        <apiErrors>
          <apiError>
            <errorMessage>The specified ID does not exist in the database.</errorMessage>
            <errorType>notFound.dbData</errorType>
          </apiError>
        </apiErrors>
        <errorMessage>There were one or more errors processing the following request: delete agent/1</errorMessage>
        <errorType>operation.resourceErrors</errorType>
      </errorDetail>
      <errorMessage>There were one or more errors processing the following request: delete agent/2</errorMessage>
      <errorType>operation.resourceErrors</errorType>
    </apiError>
  </apiErrors>
  <status>partialSuccess</status>
</operationsResult>
```

Example Failure Response

The following example shows a failure response for a request to delete a call type that does not exist:

```xml
<operationsResult>
  <status>failure</status>
  <apiError>
    <errorDetail xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="resourceErrorDetail">
      <refURL>/unifiedconfig/config/calltype/9999</refURL>
      <apiErrors>
        <apiError>
          <errorMessage>The specified ID does not exist in the database.</errorMessage>
          <errorType>notFound.dbData</errorType>
        </apiError>
      </apiErrors>
      <errorMessage>There were one or more errors processing the following request: delete /unifiedconfig/config/calltype/9999</errorMessage>
      <errorType>operation.resourceErrors</errorType>
    </errorDetail>
  </apiError>
</operationsResult>
```
Outbound API

Outbound API allows you to use REST APIs to create, modify, and delete Outbound Option campaigns. Outbound API provides a streamlined mechanism for creating campaigns with a single associated query rule and import rule. As such, if you use the API to create a campaign, that campaign is not available in the Outbound Campaign tool. If a campaign was created with the API, you must use the API to view, edit, or delete it. If a campaign was created with the Outbound Campaign tool, you must use the Outbound Campaign tool to view, edit, or delete it.

Administrative scripts are not required for Outbound Option campaigns created with the Outbound API. If an administrative script is provided, the information in the script overrides the information defined in the API.

Outbound API consists of the following APIs:

- Outbound Campaign API: Use this API to define new Outbound Option campaigns, and to view, edit, or delete existing campaigns.
- Campaign Status API: Use this API to get the real-time status of running Outbound Option campaigns.
- Do Not Call API: Use this API to set the Do Not Call (DNC) import rule configuration for Outbound Option. This prevents the dialer from dialing numbers on the DNC list.
- Import API: Use this API to import customer contact information for an Outbound Option campaign.
- Personal Callback API: Use this API to configure your Outbound Option campaign to handle personal callbacks. You can create personal callback records individually or in bulk. You can also update or delete personal callback records.
- Time Zone API: Use this API to list all available time zones and to get information about a specified time zone.

- Outbound Campaign API, page 72
- Campaign Status API, page 78
- Do Not Call API, page 80
- Import API, page 82
- Personal Callback API, page 86
- Time Zone API, page 91
Outbound Campaign API

An Outbound Campaign makes outgoing calls to customers for a specific purpose or task and delivers these calls to agents.

Use the Outbound Campaign API to define new outbound campaigns, and to view, edit, or delete existing outbound campaigns.

URL

https://<server>/unifiedconfig/config/campaign

Operations

• create: Creates one campaign and stores it in the database.

• delete: Deletes one campaign from the database. The campaign is saved under Configuration Manager > Miscellaneous Tools > Deleted Objects.

  You can only delete campaigns that were created using the API. You cannot use the API to delete campaigns that were created in the Outbound Option Campaign tool.

• list: Retrieves a list of campaigns from the database. Only campaigns that were created using the Outbound Option API are retrieved. Campaigns created using the Outbound Option Campaign tool do not appear in the list.

• get: Returns one campaign from the database using the URL

  https://<server>/unifiedconfig/config/campaign/<id>.

  You must specify the ID for a campaign that was created using the API. If you specify the ID for a campaign created using the Outbound Option Campaign tool, the request returns CceDBDataNotFoundException.

• update: Updates one campaign in the database using the URL

  https://<server>/unifiedconfig/config/campaign/<id>.

  You can only update campaigns that were created using the API. You cannot use the API to update campaigns that were created in the Outbound Option Campaign tool.

Parameters

• name: Required. The name of the campaign. See Shared Parameters, on page 9.

• description: Optional description for the campaign. See Shared Parameters, on page 9.

• linesPerAgent: The number of lines dedicated to each agent in the campaign. Range is 1 to 100. Default is 1.5. This parameter performs as follows in the Outbound Option dialing modes:

  ◦ Preview mode: Ignored (always 1).
  ◦ Progressive mode: Used as defined.
  ◦ Predictive mode: Used as an initial value.

• maximumLinesPerAgent: The upper bound for the number of customers the dialer dials for a reserved agent when a campaign is running in predictive mode. Range is 1 to 100. Default is 2.
• abandonEnabled: True or false. Default is true.

• abandonPercent: When enabled (abandonEnabled is set to true), you can set the abandoned calls limit for the percentage of abandoned calls in the campaign. When disabled (abandonEnabled is set to false), the campaign dials without regard to the abandon limit. Range is 1 to 100. The granularity is to one 10th of a percent. Default is 3.

• predictiveCorrectionPace: A count of the number of live voice connections that must occur before the Dialer adjusts. Increasing this number results in less frequent adjustments based on larger sample size. Decreasing this number results in more frequent adjustments using a smaller sample size. Range is 10 to 5000. Default is 70.

• predictiveGain: The size of the adjustment to lines per agent each time an adjustment is made. Increasing this number results in larger lines per agent adjustments. Decreasing this number results in smaller lines per agent adjustments. Range is 0.1 to 3.0. Default is 1.

• noAnswerRingLimit: The number of times the software allows a dialed phone number to ring. Range is from 2 to 10. Default is 4.

• maxAttempts: The maximum number of attempts, including callbacks and retries. Range is from 1 to 100. Default is 3.

• minimumCallDuration: Minimum duration (in seconds) of an outbound call. If the outbound call is less than the specified value, Outbound Option considers the call to be customer abandoned and schedules the record for a retry. To disable this feature, set the parameter to 0. Range is 0 to 10. Default is 1.

• noAnswerDelay: The time (in minutes) that the software waits before calling back a no-answer call. Range is 1 to 99999. Default is 60.

• busySignalDelay: The time (in minutes) that the software waits before calling back a busy phone number. Range is 1 to 99999. Default is 60.

• customerAbandonedDelay: If a customer abandons a call, the time (in minutes) that the Dialer waits before calling the customer back. Range is 1 to 99999. Default is 30.

• dialerAbandonedDelay: If the Dialer abandons a call, the time (in minutes) that the Dialer waits before calling the customer back. Range is 1 to 99999. Default is 60.

• answeringMachineDelay: If an answering machine answers a call, the time (in minutes) that the Dialer waits before calling the customer back. Range is 1 to 99999. Default is 60.

• customerNotHomeDelay: If a customer is not home, the time (in minutes) that the Dialer waits before calling the customer back. Range is 1 to 99999. Default is 60.

• personalizedCallbackEnabled: If enabled, this parameter allows an agent to schedule a callback to a customer for a specific date and time. A personal callback connects the same agent who initiated the callback to the customer. True or false. Default is false.

• rescheduleCallbackMode: Determines how Outbound Option handles a personal callback if the agent is not available. Default is useCampaignDN. Options are as follows:
  * Use campaign DN
  * Same time next business day
  * Abandon

• campaignPurposeType: The type of campaign. Default is agentCampaign. The options are as follows:
• Agent Campaign: This type of campaign uses an outbound mode that causes the Dialer to transfer every customer call associated with a specific skill group to an agent.

• IVR Campaign: This type of campaign uses an outbound mode that causes the Dialer to transfer every customer call associated with a specific skill group to a service control-based IVR instead of an agent. This option allows a contact center to run unassisted outbound campaigns using prerecorded messages in the IVR.

• ipAmdEnabled: When enabled, directs the Dialer to perform a specific action if it detects an answering machine. True or false. Default is true.

• amdTreatmentMode: If enabled (ipAmdEnabled is set to true), when the Dialer detects an answering machine, it does one of the following:
  ◦ Abandon call (default)
  ◦ Transfer to agent
  ◦ Transfer to IVR route point
  ◦ Terminate tone detect

• ipTerminatingBeepDetect: When this parameter is set to true, the Dialer transfers the call after detecting the answering machine beep. True or false. Default is false.

• timeZone: Required. The refURL and display name for the selected time zone. The default time zone is UTC (Universal Coordinated Time). The display name is the text that may be displayed in a user interface and can be localized.

  ![Note](image)
  If time zone information changed due to periodic updates and the campaign's configured time zone is no longer valid, the following information is returned:

  ```xml
  <timeZone>
  <refURL>/unifiedconfig/config/timezone/INVALID</refURL>
  <displayName>INVALID Time Zone</displayName>
  </timeZone>
  ```

• startTime: The time the campaign starts dialing customer numbers. The format for this parameter is hours:minutes. Range is from 00:00 to 23:59. The default value is taken from the Blended_Agent_Options table column values for DialStartHours and DialStartMinutes.

• endTime: The time the campaign stops dialing customer numbers. The format for this parameter is hours:minutes. Range is from 00:00 to 23:59. The default value is taken from the Blended_Agent_Options table column values for DialEndHours and DialEndMinutes.

• enabled: Whether the dialer is available to call contacts. True or false. Default is false.

• startDate: The date that the campaign starts. The format is YYYY-MM-DD.

• endDate: The date that the campaign ends. The format is YYYY-MM-DD.

• campaignPrefix: Digits to prefix to each customer number dialed from this campaign. Maximum length of 15 digits.

• dialingMode: The dialing mode to use for the campaign skill groups. Valid values are as follows:
  ◦ INBOUND
<p>PREDICTIVEONLY</p><p>PREVIEWONLY</p><p>PROGRESSIVEONLY</p><p>PREVIEWDIRECTONLY</p>

- **reservationPercentage:** The percentage of agents to reserve within the skill groups associated with the campaign. Range is from 0 to 100. Default is 100.

- **callProgressAnalysis:** A collection of parameters for Call Progress Analysis (CPA). Any combination of parameters within this collection can be set. If none of the parameters are provided and CPA is enabled for the campaign by default, CPA recording is set to false and default parameter values are set from the Blended_Agent_Options table.

  - **enabled:** When set to false, CPA for all calls made from this Dialer is disabled on a campaign-by-campaign basis, including voice detection, fax/modem detection, and answering machine detection. True or false. Default is true.

  - **record:** If enabled is set to true, you can specify this parameter. If you set it to true, the gateway provides a media stream and the Dialer records .wav files. True or false. Default is false.

  - **minSilencePeriod:** The minimum silence period (in milliseconds) required to classify a call as voice detected. If many answering machine calls are being passed through to agents as voice, then increasing this value accounts for longer pauses in answering machine greetings. Range is from 100 to 1000. Default is 608.

  - **analysisPeriod:** The number of milliseconds spent analyzing this call. If there is a short agent greeting on an answering machine, then a longer value categorizes that answering machine call as voice. If the call is to a business where the operator has a longer scripted greeting, a shorter value categorizes the long, live greeting as an answering machine call. Range is from 1000 to 10000. Default is 2500.

  - **minimumValidSpeech:** Minimum number of milliseconds of voice required to classify a call as voice detected. Range is 50 to 500. Default is 112.

  - **maxTimeAnalysis:** The maximum number of milliseconds allowed for analysis before identifying a problem analysis as dead air/low volume. Range is 1000 to 10000. Default is 3000.

  - **maxTermToneAnalysis:** The maximum number of milliseconds the dialer analyzes an answering machine voice message looking for a termination tone. If the message has an odd tone and the analysis does not recognize it, the call is not transferred or dropped until this timeout occurs. Range is 1000 to 60000. Default is 30000.

- **skillGroupInfos:** A collection of information about the skill groups associated with the campaign.

  - **skillGroupInfo:** A collection of information about one skill group.

    - **skillGroup:** The name and the refURL of the skill group assigned to the campaign.

    - **overflowAgents:** This parameter ensures that at least one extra agent is reserved before the dialer begins dialing for a Progressive campaign. If the parameter is set to 1, at least two agents must be reserved before the dialer begins dialing. Range is 0 to 100. Default is 0.

    - **dialedNumber:** The digits that are dialed to reserve an agent in the configured skill group. This parameter can contain letters, numbers, periods (.), and underscores (_) and can be up to ten characters in length.
recordsToCache: The minimum number of dialing numbers that each dialer caches for each of the Outbound Option skill groups. Range is 1 to 100. Default is 1.

ivrPorts: The total number of IVR ports allocated for the skill group. This parameter indicates how many ports are available for the dialer to transfer customer calls. Because this value indicates the total number of ports supported by the IVR for the current skill group, multiple skill groups can make transfer to IVR calls. One IVR can also be used to play different messages based on the route point where the contact is transferred. If multiple dialers are associated with the skill group, each dialer dials a fraction of the total number of ports. Default is 0.

ivrRoutePoint: If the campaign is a Transfer to IVR campaign or is configured to transfer AMD calls to an IVR, this parameter indicates the route point required to run the transfer to IVR routing script. This parameter must coincide with a route point configured on Unified Communications Manager and be assigned to a PGUser. Contacts are transferred to the route point, which points to a routing script. The script transfers the call to an IVR. Maximum length of 32 characters.

abandonedRoutePoint: If the campaign is a Transfer to IVR campaign or is configured to transfer AMD calls to an IVR, this number allows the dialer to play a message to calls about to be disconnected because no agents are available. This number must coincide with a route point configured on Unified Communications Manager and be assigned to the agent PG's CTI application (for example, PGUser). Contacts are transferred to this route point, which points to a routing script. The script transfers the call to an IVR. Maximum length of ten characters.

• skillGroupInfosAdded: A collection of skill group references to be added to the campaign, including the skill group refURL. This parameter is update only. This parameter can be used with the skillGroupInfosRemoved parameter. See References, on page 6.

• skillGroupInfosRemoved: A collection of skill group references to be removed from the campaign, including the skill group refURL of each skill group. This parameter is update only. This parameter can be used with the skillGroupInfosAdded parameter. See References, on page 6.

Search and Sort Parameters
The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name (default)</td>
</tr>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

Disable All Campaigns (Emergency Stop)
The Outbound Campaign API provides a method to immediately disable all outbound campaigns that are currently enabled. To disable all of the campaigns, perform a PUT on the following URL:

https://<server>/unifiedconfig/config/campaign/disable
This request retrieves a list of all campaigns created using the Outbound Campaign API that are currently enabled and sets the enabled parameter to false. The Campaign Manager stops sending out records to the Dialer for these campaigns. All active records in the Dialer’s memory are removed.

**Example Get Response**

```xml
<campaign>
  <changeStamp>48</changeStamp>
  <refURL>/unifiedconfig/config/campaign/5168</refURL>
  <import>
    <refURL>/unifiedconfig/config/import/5000</refURL>
  </import>
  <abandonEnabled>true</abandonEnabled>
  <abandonPercent>3.0</abandonPercent>
  <amdTreatmentMode>abandonCall</amdTreatmentMode>
  <campaignPrefix>978</campaignPrefix>
  <campaignPurposeType>agentCampaign</campaignPurposeType>
  <dialingMode>PREVIEWONLY</dialingMode>
  <enabled>true</enabled>
  <endDate>2016-01-15</endDate>
  <endTime>17:00</endTime>
  <ipAMDEnabled>true</ipAMDEnabled>
  <ipTerminatingBeepDetect>false</ipTerminatingBeepDetect>
  <linesPerAgent>1.5</linesPerAgent>
  <maxAttempts>3</maxAttempts>
  <maximumLinesPerAgent>100.0</maximumLinesPerAgent>
  <minimumCallDuration>1</minimumCallDuration>
  <name>APIoct1</name>
  <noAnswerRingLimit>4</noAnswerRingLimit>
  <personalizedCallbackEnabled>false</personalizedCallbackEnabled>
  <predictiveCorrectionPace>70</predictiveCorrectionPace>
  <predictiveGain>1.0</predictiveGain>
  <retries>
    <answeringMachineDelay>60</answeringMachineDelay>
    <busySignalDelay>60</busySignalDelay>
    <customerAbandonedDelay>30</customerAbandonedDelay>
    <customerNotHomeDelay>60</customerNotHomeDelay>
    <dialerAbandonedDelay>60</dialerAbandonedDelay>
    <noAnswerDelay>60</noAnswerDelay>
  </retries>
  <skillGroupInfos>
    <skillGroupInfo>
      <ivrPorts>0</ivrPorts>
      <overflowAgents>0</overflowAgents>
      <recordsToCache>1</recordsToCache>
      <abandonedRoutePoint>12345</abandonedRoutePoint>
      <dialNumber>123</dialNumber>
      <ivrRoutePoint>91234</ivrRoutePoint>
      <skillGroup>
        <refURL>/unifiedconfig/config/skillgroup/5004</refURL>
        <name>errorDetailsRouteScript</name>
      </skillGroup>
    </skillGroupInfo>
  </skillGroupInfos>
  <startDate>2016-01-14</startDate>
  <startTime>09:00</startTime>
  <timeZone>
    <displayName>(UTC-05:00) Eastern Time (US & Canada)</displayName>
    <refURL>/unifiedconfig/config/timezone/Eastern%20Standard%20Time</refURL>
  </timeZone>
</campaign>
```
Example Create Request

```xml
<campaign>
  <name>APIOct1</name>
  <description>APIOct1</description>
  <dialingMode>PREVIEWONLY</dialingMode>
  <skillGroupInfos>
    <skillGroupInfo>
      <ivrPorts>0</ivrPorts>
      <overflowAgents>0</overflowAgents>
      <recordsToCache>1</recordsToCache>
      <skillGroup>
        <refURL>/unifiedconfig/config/skillgroup/5001</refURL>
        <name>sgcampaign</name>
      </skillGroup>
    </skillGroupInfo>
  </skillGroupInfos>
  <timeZone>
    <refURL>/unifiedconfig/config/timezone/UTC</refURL>
  </timeZone>
  <callProgressAnalysis>
    <enabled>true</enabled>
    <record>false</record>
    <minSilencePeriod>608</minSilencePeriod>
    <analysisPeriod>2500</analysisPeriod>
    <minimumValidSpeech>112</minimumValidSpeech>
    <maxTimeAnalysis>3000</maxTimeAnalysis>
    <maxTermToneAnalysis>30000</maxTermToneAnalysis>
  </callProgressAnalysis>
</campaign>
```

Campaign Status API

Use the Campaign Status API to get the real-time status of running Outbound Option campaigns.

**URL**

https://<server>/unifiedconfig/config/campaign/<campaign-id>/runtimestats

**Note**

You must specify the ID for a campaign that was created using the Outbound Campaign API (see Outbound Campaign API, on page 72). If you specify the ID for a campaign created using the Outbound Option Campaign tool, the request returns CceDBDataNotFoundException.

**Operations**

- **get**: Returns the runtime status of a campaign.

**Response Parameters**

- **abandonDetectCount**: The number of calls abandoned by the dialer.
- **abandonToIvrCount**: The number of calls that were abandoned by the dialer and transferred to IVR.
- **agentClosedCount**: The number of preview and callback calls that were closed by the agent.
• agentRejectedCount: The number of preview and callback calls that were rejected by the agent.
• answeringMachineCount: The number of calls that detected an answering machine.
• attemptedCount: Summary total of the number of calls attempted.
• busyCount: The number of calls that detected a busy signal.
• callBackCount: The number of callback contacts.
• cancelledDetectCount: The number of calls where the dialer canceled a ringing customer call.
• closedCount: The number of contacts closed for any reason other than reaching a live customer.
• customerAbandonDetectCount: The number of calls where the customer hung up immediately after picking up the phone.
• customerNotHomeCount: The number of contacts where the party who answered the phone was not the customer.
• dateTime: The date and time when this data was updated last. The format is yyyy-MM-ddTHH:mm:ss (for example, 2016-03-13T04:50:31).
• faxDetectCount: The number of calls that detected a fax machine.
• networkAnsMachineCount: The number of calls that detected a network answering machine.
• noAnswerDetectCount: The number of calls that were not answered.
• noDialToneDetectCount: The number of calls that did not detect a dial tone.
• noRingBackDetectCount: The number of calls that did not detect a ring back.
• personalCallbackCount: The number of callback contacts scheduled.
• sitToneDetectCount: The number of calls that detected a special information tone (SIT).
• talkTimeCount: The total number of seconds that agents spent talking on the phone since midnight.
• totalCount: The total number of records available to dial for the current campaign.
• totalVoiceCount: The number of live customers that have been reached for this campaign since the last time the imported dialing list was overwritten.
• voiceCount: The number of calls for the day that ended in successful customer contact.
• wrapupTimeCount: The number of seconds that agents spent in wrap-up mode since midnight.
• wrongNumberCount: The number of contacts where the party who answered the phone indicated that the customer did not live there.

Example Get Response

```xml
<runtimeStatus>
  <abandonDetectCount>0</abandonDetectCount>
  <abandonToIvrCount>0</abandonToIvrCount>
  <agentClosedCount>0</agentClosedCount>
  <agentRejectedCount>0</agentRejectedCount>
  <answeringMachineCount>0</answeringMachineCount>
  <attemptedCount>1</attemptedCount>
  <busyCount>0</busyCount>
  <callBackCount>0</callBackCount>
  <cancelledDetectCount>0</cancelledDetectCount>
  <closedCount>0</closedCount>
</runtimeStatus>
```
Do Not Call API

Use the Do Not Call API to set the Do Not Call (DNC) import rule configuration for Outbound Option so that the campaign dialer doesn't dial the numbers in the DNC list.

The DNC import is automatically scheduled to start when the file is available. After the import is complete, the file is RENAMED or DELETED based on the boolean value for the renameFileAfterImport field.

URL
https://<server>/unifiedconfig/config/dnc

Operations

• **create**: Sets the configuration in the database. Returns the refURL if the configuration is set.

• **delete**: Deletes the specific DNC configuration using the URL https://<server>/unifiedconfig/config/dnc/<id>. This does not delete the DNC import file that is present at the location of the filePath.

• **list**: Retrieves a list available DNC import rules.

• **get**: Returns the DNC import configuration from the database using the URL https://<server>/unifiedconfig/config/dnc/<id>.

• **update**: Updates the specific DNC import configuration with the new values using the URL https://<server>/unifiedconfig/config/dnc/<id>.

Parameters

• **name**: Name of the DNC import. See Shared Parameters, on page 9.

• **filePath**: Path to a file on the logger or accessible from the logger. Maximum length of 255 characters.

• **overwrite**: Whether the new DNC import overwrites the phone numbers from the previous DNC import. True or false. Default is false.

• **renameFileAfterImport**: Whether to delete the DNC import file after the import is complete. True or false. Default is true. If the parameter is set to true, the DNC file is renamed. If the parameter is set to false, the DNC import file is deleted.
Search and Sort Parameters

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

Example Get Response

```xml
<dnc>
  <refURL>/unifiedconfig/config/dnc/unifiedconfig/config/dnc/5001</refURL>
  <name>dnc1</name>
  <filePath>\CCE-ROGGER-A\CUsers\boston\Desktop\dnc.txt</filePath>
  <overwrite>false</overwrite>
  <renameFileAfterImport>true</renameFileAfterImport>
</dnc>
```

Create Do Not Call List

When creating a Do_Not_Call list file, format it correctly using the following instructions.

Procedure

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Using a text editor, create a text file that contains all the do-not-call phone numbers.</td>
</tr>
<tr>
<td>Step 2</td>
<td>Enter a phone number for each Do Not Call entry on a new line.</td>
</tr>
<tr>
<td>Step 3</td>
<td>Observe the following characteristics for each Do Not Call entry:</td>
</tr>
<tr>
<td></td>
<td>• Each phone number can be a maximum of 20 characters long.</td>
</tr>
<tr>
<td></td>
<td>• The Do Not Call table can support up to 60 million entries, but note that the information is stored in memory in the Campaign Manager process.</td>
</tr>
<tr>
<td></td>
<td>• Each Do Not Call entry uses 16 bytes of memory, so 60 million entries would require approximately 1 gigabyte of memory (960 million bytes) on the Logger Side A platform.</td>
</tr>
<tr>
<td>Step 4</td>
<td>Save the text file to the local server.</td>
</tr>
</tbody>
</table>

The following is an example of a Do_Not_Call list:

2225554444
2225556666
2225559999

To add a customer to this list, import a Do Not Call list.

The Campaign Manager reads the Do Not Call import files. Dialing List entries are marked as Do Not Call entries only when the Campaign Manager fetches the Dialing List entry and only when there is an exact,
**Digit-for-digit match.** This allows Do Not Call imports to happen while a Campaign is running without rebuilding the Dialing List.

**Note**

If the Dialing List includes a base number plus extension, this entry must match a Do Not Call entry for that same base number and same extension. The dialer will not dial the extension.

When the Campaign Manager starts it automatically imports from the DoNotCall.restore file that is stored in the `<drive>\icm\<instance>\la\bin` directory. When reading Do Not Call import files, the Campaign Manager appends the data to the DoNotCall.restore file. This restore file allows recovery of Do Not Call records after the Campaign Manager stops unexpectedly or for planned maintenance, such as a Service Release installation.

The restore file can grow to approximately 1 GB if 60 million DNC records are imported, each having ten-digit numbers plus five-digit extensions. Sufficient disk space must be available on LoggerAt to store the DoNotCall.restore file.

**Note**

To clear the Do Not Call list, import a blank file with the Overwrite table option enabled.

---

**Import API**

Use the Import API to import customer contact information for an outbound campaign.

You can import up to 10,000 records in one create request.

**Note**

You can queue up to 30 requests at a time. If you have 30 requests in queue and submit another request, the response is HTTP 503 Service Unavailable with the boundary error condition "Request processing queue is at capacity."

If you receive this error, wait a few seconds and then retry the request.

**URL**

https://<server>/unifiedconfig/config/campaign/<id>/import

**Note**

You must specify the ID for a campaign that was created using the Outbound Campaign API. If you specify the ID for a campaign created using the Outbound Option Campaign tool, the request returns CceDBDataNotFoundException.

**Operations**

- **create**: Imports customer contacts for a specific campaign.
- **delete**: Deletes the imported contacts for a specific campaign.
The delete operation deletes all imported contacts for the campaign. You cannot delete an individual record.

- **get (template):** Returns a sample CSV template for contacts, which is provided by the API, using the URL `https://<server>/unifiedconfig/config/campaign/import/template`. The response contains the CSV template as a file attachment.

- **get:** Retrieves the details of a single import record using the URL `https://<server>/unifiedconfig/config/campaign/<campaignId>/import/<id>`. The `<id>` is the ID of the imported contact.

- **list:** Retrieves a list of all imported contacts for the campaign.

  - **Query parameters**
    - Summary list: See list, on page 4.

### Parameters

#### Parameters for get:

- **accountNumber:** The customer's account number.
- **callsMade:** The number of calls made.
- **callStatus:** The call status of the last call placed for this record. Possible values include the following:
  - active (A)
  - callbackRequested (B)
  - closed (C)
  - agentRejected (J)
  - maxAttemptsReached (M)
  - pending (P)
  - retry (R)
  - personalCallbackRequested (S)
  - unknown (U)


- **callResultOverall:** The call result from the last call placed for this record.
- **firstName:** The customer's first name.
- **lastName:** The customer's last name.
- **importDate:** The date and time that the record was imported.
• phone01 through phone10: A collection of information about the customer's phones. Each phone contains the following parameters:
  * number: The phone number.
  * callResult: The call result from the last call placed for this phone number.
  
  
  * dstObserved: Whether DST is observed for this phone's location. True or false.
  * gmtOffset: The number of minutes that the customer's time zone is offset from GMT. When you create import records, you can optionally provide a TimeZoneBias. The TimeZoneBias parameter represents the way information about time zones is stored in the Windows registry. You can use the following formula to convert TimeZoneBias to gmtOffset:

    If timeZoneBias is positive:
      gmtOffset = 1440 - timeZoneBias (where 1440 is the number of minutes in 24 hours)
    Else if timeZoneBias is negative:
      gmtOffset = -1 * timeZoneBias
    Else if timeZoneBias is 0:
      gmtOffset = timeZoneBias

Parameters for create:

• fileContent: Required. Comma-separated or pipe-separated list of data embedded within a CDATA section.

• overwriteData: True or false. If set to true, the existing import data in the database is overwritten by the new import data. If set to false, the new import data is appended to the existing data. Default is false.

  **Note** You cannot modify or remove existing fields or append a new field to existing data.

• delimiter: Comma (,) or pipe (|). Default is comma (,).

• header fields: The following header fields are allowed:

  * AccountNumber: Can contain any characters, including internationalized characters, except the delimiter specified in the input XML. Maximum length of 30 bytes.

  * FirstName: Can contain any characters, including internationalized characters, except the delimiter specified in the input XML. Maximum length of 50 characters.

  * LastName: Can contain any characters, including internationalized characters, except the delimiter specified in the input XML. Maximum length of 50 characters.

  * Phone01 through Phone10: Can contain digits 0-9, pound sign (#), and asterisk (*). Maximum length of 20 bytes. At least one Phone field is required.

  * TimeZoneBias: Specifies the offset of the customer's time zone from UTC in minutes. Integer. Range is from –780 to 720 (–13 to 12 hours from UTC).

  * DstObserved: Specifies whether DST is observed for the customer's location. True or false. Default is false.
If the TimeZoneBias parameter is provided but the DstObserved parameter is not, DstObserved is set to false. If the DstObserved parameter is provided but not TimeZoneBias, a validation error is returned.

If TimeZoneBias is not provided, time zone and Daylight Saving Time information is assigned by matching phone numbers to region prefix strings. If a phone number for a contact does not match a configured region prefix, the default time zone for the campaign is used.

If a customer record contains multiple phone numbers, the value for TimeZoneBias and DstObserved are applied to all of that customer's phone numbers.

You can use the Time Zone API to look up the values for TimeZoneBias and DstObserved for the target time zone. For more information, see Time Zone API, on page 91.

### Search and Sort

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• accountNumber</td>
<td>• accountNumber (default)</td>
</tr>
<tr>
<td>• firstName</td>
<td>• firstName</td>
</tr>
<tr>
<td>• lastName</td>
<td>• lastName</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

### Example Get Response

```xml
<importContact>
  <refURL>/unifiedconfig/config/campaign/5000/import/1</refURL>
  <accountNumber>4019</accountNumber>
  <callsMade>1</callsMade>
  <callStatus>pending</callStatus>
  <callResult>0</callResult>
  <callResultOverall>0</callResultOverall>
  <firstName>Mir</firstName>
  <lastName>Ali</lastName>
  <importDate>2016-03-28T01:09:40</importDate>
  <phone01>
    <callResult>0</callResult>
    <dstObserved>true</dstObserved>
    <gmtOffset>720</gmtOffset>
    <number>9789360001</number>
  </phone01>
  <phone02>
    <callResult>0</callResult>
    <dstObserved>true</dstObserved>
    <gmtOffset>720</gmtOffset>
    <number>9789360002</number>
  </phone02>
  <phone03>...</phone03>
  <phone04>...</phone04>
</importContact>
```
Example Create Request

```
<import>
  <fileContent>
    <![CDATA[
      AccountNumber,FirstName,LastName,Phone01,Phone02,TimeZoneBias,DstObserved
      6782,Henry,Martin,2225554444,2225556262,720,false
      3456,Michele,Smith,2225559999,2225551234,540,true
      4569,Walker,Evans,2225552000,2225557890,-600,true
    ]]>}
  </fileContent>
</import>
```

Personal Callback API

Use the Personal Callback API to configure your Outbound Option campaign to handle personal callbacks. The Personal Callback feature allows an agent to schedule a callback to a customer for a specific date and time.

URL

https://<server>/unifiedconfig/config/personalcallback

Operations

- **create**: Creates a single PersonalCallback record and stores it in the BA database.
- **create (bulk)**: Creates PersonalCallback records in bulk and stores them in the BA database using the URL https://<server>/unifiedconfig/config/personalcallback/import.
- **delete**: Deletes one or more PersonalCallback records from the PersonalCallback list using the URL https://<server>/unifiedconfig/config/personalcallback/<id>.
- **get (template)**: Retrieves a sample CSV template for contacts, which is provided by the API, using the URL https://<server>/unifiedconfig/config/personalcallback/template.
- **get (record)**: Retrieves one PersonalCallback record from the database using the URL https://<server>/unifiedconfig/config/personalcallback/<id>.
- **list**: Retrieves a list of PersonalCallback records from the database.

*Query parameters:*

- agentld
- accountNumber
- `firstName`
- `lastName`

- update: Updates one PersonalCallback record in the database using the URL
  https://<server>/unifiedconfig/config/personalcallback/<id>.

**Parameters**

**Parameters for create (single record), get, and update operations:**

- campaign: A reference to a specific campaign, including the refURL and campaign ID.
- agent: A reference to a specific agent, including the refURL and agent ID.
- campaignId: Read only. The ID for the campaign.
- peripheralId: Read only. ID for the peripheral on which the agent would be available.
- agentId: Read only. ID of the agent to whom to connect the call.
- campaignDn: The Dialed Number (DN) to use if the original agent is not available.
- phone: Required. The phone number to call back. Can contain digits 0 to 9, pound sign (3), and asterisk (*). Maximum length of 20 bytes.
- accountNumber: The customer's account number. Internationalized characters are allowed. Maximum length of 30 bytes.
- maxAttempts: Required. The maximum number of times to attempt a call (decrements at each attempt). An attempt is the Dialer's attempt to reserve the agent and call the customer. Because the Dialer places multiple customer call attempts (such as busy, no answer), individual call attempts are not tracked here; only the result at the end of the callback time range. After this parameter is set to 0, no more attempts are made. Must be a positive integer value.
- callbackDateTime: Required. The time to attempt the customer callback is normalized to the logger GMT zone. For example, the Campaign Manager is in Boston and the customer is in California. The customer wants to be called back at 3 p.m. The time in this column is 6 p.m. The format for this parameter is yyyy-MM-ddTHH:mm:ss (for example, 2016-03-13T04:50:31).

The following rules apply to the callbackDateTime parameter:

- The callbackDateTime that is used is local to the time zone of the Logger. You need to consider this if the client machine on which you are using this API is in a different time zone than the Logger machine.
- The setting of the callbackDateTime by this API reads the CallbackDateTimeLimit in the registry. For more information, see the Registry Settings section of the Outbound Option Guide for Unified Contact Center Enterprise at http://www.cisco.com/c/en/us/support/customer-collaboration/unified-contact-center-enterprise/products-user-guide-list.html.
- The Campaign Manager uses the CallbackDateTimeLimit registry value to select Personal Callback records to queue to the Dialer.
- The default CallbackDateTimeLimit in the registry is 15 minutes. For create and update operations, you cannot set the callbackDateTime to 15 minutes or more before the current time. For example, if it is currently 4:00, you cannot set the callbackDateTime to 3:30.
For update operations, the new callbackDateTime value can only be set if the current callbackDateTime is 15 minutes or more after the current Logger time. For example, if the current Logger time is 4:00 and the current callbackDateTime is 4:30, the new callbackDateTime can be set to 4:10. However, if the current callbackDateTime is 4:10, it cannot be set to 4:00.

• callStatus: The status of the personal callback. Possible values include the following:
  - active (A)
  - callbackRequested (B)
  - closed (C)
  - agentRejected (J)
  - maxAttemptsReached (M)
  - pending (P)
  - retry (R)
  - personalCallbackRequested (S)
  - unknown (U)
  - agentNotAvailable (X)

For more information about these values, see the CallStatusZone Values section of the Outbound Option Guide for Unified Contact Center Enterprise at http://www.cisco.com/c/en/us/support/customer-collaboration/unified-contact-center-enterprise/products-user-guide-list.html.

• callResult: Read only. Corresponds to the CallResult value in the Dialer_Detail table.


• lastName: The customer's last name. Internationalized characters are allowed. Maximum length of 50 characters.

• firstName: The customer's first name. Internationalized characters are allowed. Maximum length of 50 characters.

**Bulk parameters:**

• fileContent: Comma-separated or pipe-separated list of data embedded within a CDATA section.

• overwriteData: True or false. If set to true, the existing personal callback data in the database is overwritten by the new personal callback data. If set to false, the new personal callback data is appended to the existing data. Default is false.

**Note**

You cannot modify or remove existing fields or append a new field to existing data. If overwriteData is set to true, any agent scheduled personal callback records are also overwritten.

• delimiter: Comma (,) or pipe (|). Default is comma (,).
Bulk fileContent parameters:

Note

The descriptions for these parameters are the same as the descriptions for the parameters for create (single record), get, and update operations. Only the case is different.

- CampaignId (must be a valid campaign ID in the database)
- AgentSkillTargetId (must be a valid SkillTargetID in the database)
- CampaignDn
- Phone
- AccountNumber
- MaxAttempts
- CallbackDateTime
- LastName
- FirstName

Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• agentId</td>
<td>• agentId</td>
</tr>
<tr>
<td>• accountNumber</td>
<td>• accountNumber</td>
</tr>
<tr>
<td>• firstName</td>
<td>• firstName</td>
</tr>
<tr>
<td>• lastName</td>
<td>• lastName</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

Advanced search parameters

There are several advanced searches you can perform on the Personal Callback API, including agentId, accountNumber, firstName, and lastName. All search terms are case-insensitive.

- `agentId:<ID>` finds all personal callback records where the agent ID contains a specified ID. For example, `agentId:123` returns all records where the agent ID contains the string "123".

- `lastName:<name>` finds all personal callback records where the last name contains the specified name. For example, `lastName:smith` returns all records where the last name includes "smith".

- `firstName:<name>` finds all personal callback records where the firstName contains the specified name. For example, `firstName:John` returns all records where the first name includes "john".
• **accountNumber:<number>** finds all personal callback records where the accountNumber contains the specified number. For example, **accountNumber:456** returns all records where the account number includes "456".

**Example Create Request (Single Record)**

```xml
<personalCallback>
  <campaign>
    <refURL>/unifiedconfig/config/campaign/5000</refURL>
  </campaign>
  <agent>
    <refURL>/unifiedconfig/config/agent/5050</refURL>
  </agent>
  <campaignDn>222222</campaignDn>
  <phone>999333</phone>
  <accountNumber>23334343334</accountNumber>
  <maxAttempts>1</maxAttempts>
  <callbackDateTime>2016-01-15T11:37:00</callbackDateTime>
  <callStatus>pending</callStatus>
  <lastName>Kumar</lastName>
  <firstName>Akshaya</firstName>
</personalCallback>
```

**Example Create Request (Bulk)**

```xml
<personalCallback>
  <fileContent>
    <![CDATA[
      AccountNumber,FirstName,LastName,Phone,AgentSkillTargetId,CampaignId,CampaignDn,
      CallbackDateTime,MaxAttempts
      6782,Henry,Martin,2225554444,1004,5000,2222222221,2016-01-25T05:12:00,1
      3444,Thomas,Edison,2225554555,1004,5000,2222222222,2016-01-25T05:23:00,1
      5444,Tom,Hilfiger,2225554666,1004,5000,2222222223,2016-01-25T05:37:00,1
    ]]> 
  </fileContent>
</personalCallback>
```

**Example Get (Record) Response**

```xml
<personalCallback>
  <changeStamp>48</changeStamp>
  <refURL>/unifiedconfig/config/campaign/personalcallback/2</refURL>
  <campaign>
    <refURL>/unifiedconfig/config/campaign/5000</refURL>
  </campaign>
  <agent>
    <refURL>/unifiedconfig/config/agent/5050</refURL>
  </agent>
  <campaignId>5000</campaignId>
  <peripheralId>5000</peripheralId>
  <agentId>1001</agentId>
  <campaignDn>222222</campaignDn>
  <phone>999333</phone>
  <accountNumber>23334343334</accountNumber>
  <maxAttempts>1</maxAttempts>
  <callbackDateTime>2016-01-15T11:37:00</callbackDateTime>
  <callStatus>pending</callStatus>
  <callResult>0</callResult>
  <lastName>Kumar</lastName>
  <firstName>Akshaya</firstName>
</personalCallback>
```
Time Zone API

Use the Time Zone API to list all available time zones and to get time zone information for a specified zone. Time zone information is stored in the registry of the Windows operating system.

**Important**

Microsoft periodically releases cumulative time zone updates. These updates include worldwide changes to time zone names, bias (the amount of time in minutes that a time zone is offset from Coordinated Universal Time (UTC)), and observance of daylight saving time. These patches update the information in the Windows registry. When these updates are available, apply them to all virtual machines in the deployment that are running a Microsoft Windows operating system.

Use this API with the Outbound Campaign API to set the default time zone for an Outbound Option campaign. An Outbound Option campaign uses the time zone when the location of the customer number being dialed is unknown.

This API is read-only.

**URL**

https://<server>/unifiedconfig/config/timezone

**Operations**

- **list**: Retrieves a list of available time zones. The list is sorted by UTC offset from the International Date Line from west to east.

- **get**: Returns information for a specific time zone using the URL
  https://<server>/unifiedconfig/config/timezone/<id>, where <id> is the URL-encoded version of the name parameter.

**Response Parameters**

- **name**: The name of the time zone.

- **displayName**: Specific bias and location information about the time zone, such as the offset from UTC and one or more places located within the time zone.
  
  Example: “(UTC+5:30) Chennai, Kolkata, Mumbai, New Delhi”

- **stdName**: The time zone name during standard time.
  
  Example: Malay Peninsula Standard Time

- **dstName**: The time zone name during daylight saving time.
  
  Example: Malay Peninsula Daylight Time

- **dstObserved**: Indicates whether daylight saving time is observed for the time zone. True or false.

- **bias**: The current bias for local time translation on the server (in minutes). That is, the number of minutes to add to the local time to yield UTC.
Example Get Response

<TimeZone>
  <refURL>/unifiedconfig/config/timezone/Pacific%20Standard%20Time%20(Mexico)%29</refURL>
  <name>Pacific Standard Time (Mexico)</name>
  <displayName>(UTC-08:00) Baja California</displayName>
  <stdName>Pacific Standard Time (Mexico)</stdName>
  <dstName>Pacific Daylight Time (Mexico)</dstName>
  <dstObserved>true</dstObserved>
  <bias>480</bias>
</TimeZone>
Precision Queue API

Precision queues help direct incoming callers to appropriate agents, as they match specific agent attributes with caller requirements. If a precision queue requires an agent who lives in Boston and who speaks fluent Spanish, then an agent with the attributes $\text{Boston} = \text{True}$ and $\text{Spanish} = \text{True}$ is a good match.

Use the Precision Queue API to list the precision queues currently defined in the database, define new precision queues, and view, edit, and delete existing precision queues.

**URL**

https://<server>/unifiedconfig/config/precisionqueue

**Operations**

- **create**: Creates one precision queue.
- **delete**: Marks one precision queue for deletion, but does not permanently delete it. Deleting a precision queue that is referenced dynamically in a script is allowed. No new calls are queued against it, but the precision queue remains operational until calls are no longer in the queue.
- **get**: Returns one precision queue, using the URL https://<server>/unifiedconfig/config/precisionqueue/<id>.
  - Query parameters:
    - agentcount: Use this query parameter to have the agent count parameter included in the response; for example, agentcount=true.
    - attributes: Use this query parameter to have the attribute parameter included in the response; for example, attributes=true.
    - skillgroups: Use this query parameter to augment the returned precision queue attributes with an id listing of all of the skillgroups that are associated with the precision queue; for example, skillgroups=true.
- **list**: Retrieves a list of precision queues. Query parameters described above for the get operation are also allowed for list.
- **update**: Updates one precision queue.
Parameters

Precision queue parameters:

• refURL: The refURL of the precision queue. See Shared Parameters, on page 9.
• name: The name of the precision queue. See Shared Parameters, on page 9.
• changeStamp: See Shared Parameters, on page 9.
• description: See Shared Parameters, on page 9.
• mediaRoutingDomain: A reference to the media routing domain (Media Routing Domain API, on page 63) including the name and refURL. See References, on page 6.
  ◦ Defaults to Cisco_Voice MRD if this parameter is not provided.
  ◦ This reference cannot be updated.
• bucketInterval: A reference to a bucket interval (Bucket Interval API, on page 31), including the refURL and name. See References, on page 6.
• agentCount: Returns agent count for the precision queue. Returned only when using the agentcount query parameter.
• agentOrdering: Determines the order in which agents receive calls from this queue.
  ◦ 1: LAA (Agent availability time)
  ◦ 2: Most skilled agent
  ◦ 3: Least skilled agent
• id: The database id of the precision queue. Read-only field. Used in scripting.
• attributes: A collection of attribute names (attribute1, attribute2, and so on) indicating all of the attributes used in this precision queue. Returned only when the query parameter attributes=true.
• serviceLevelThreshold: Maximum time in seconds that a caller should wait before being connected with an agent.
• serviceLevelType: This value indicates how the system calculates the service level.
  ◦ 1: Ignore abandoned calls.
  ◦ 2: Abandoned call has negative impact.
  ◦ 3: Abandoned call has positive impact.
• skillgroups: A collection of skill groups associated with this precision queue, including the id of each skill group. Returned only when the query parameter skillgroups=true.
• steps: Required. A collection of steps for this precision queue. You can have 1-10 steps. Returned only for get operation. See the Step parameters below.

Step parameters:

• waitTime: Time in seconds to wait before proceeding to the next step.
• considerIf: A Consider If expression which must be met to execute a particular step. Items used in the expression are case sensitive. You cannot add an expression to the last step.
• terms: Required. A collection of terms for this step. Each step can have 1-10 terms. See the Term parameters below.

**Term parameters:**

• attribute: A reference to the attribute ([Attribute API, on page 27](#)), including the refURL, name, description, and dataType. Multiple unique attributes can be used across all terms in a precision queue.

• parenCount: Denotes a parenthesis before or after this term. A value of 1 means a parenthesis before the current term, and a value of -1 means a parenthesis after the current term. The sum of all parenCount for all terms in a step must be equal to zero, meaning that all parenthesis in the expression are matched. For example, a step to check for agents that have (sales > 7 or expertSales = true) and english = true requires 3 terms with the parenCount set to 1 on the first term, -1 on the second term, and 0 on the last term.

• termRelation: Indicates the relationship of this term to the preceding term, using the following values:
  - 0: None. Valid only on the first term in a step.
  - 1: AND
  - 2: OR

• attributeRelation: Indicates what kind of comparison is done on the attribute, using the following values:
  - 1: Equal
  - 2: Not equal
  - 3: Less than
  - 4: Less than or equal
  - 5: Greater than
  - 6: Greater than or equal

• value1: The value that the attribute is tested against. For boolean attributes, this value must be true/false. For proficiency attributes, this value must be 1-10.

**Search and Sort Values**

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name (default)</td>
</tr>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
</tbody>
</table>

See [Search, on page 11](#) and [Sort, on page 12](#).

**Example Get Response**

```xml
<precisionQueue>
 <changeStamp>4</changeStamp>
</precisionQueue>
```
<precisionQueue>
  <agentOrdering>1</agentOrdering>
  <bucketInterval>
    <name>Default_Bucket_Intervals</name>
    <refURL>/unifiedconfig/config/bucketinterval/1</refURL>
  </bucketInterval>
  <mediaRoutingDomain>
    <name>Cisco_Voice</name>
    <refURL>/unifiedconfig/config/mediaroutingdomain/1</refURL>
  </mediaRoutingDomain>
  <description>This is a practice precision queue</description>
  <name>Practice_Queue</name>
  <serviceLevelThreshold>3</serviceLevelThreshold>
  <serviceLevelType>1</serviceLevelType>
  <steps>
    <step>
      <terms>
        <term>
          <attribute>
            <refURL>/unifiedconfig/config/attribute/5698</refURL>
            <name>test</name>
            <dataType>4</dataType>
          </attribute>
          <attributeRelation>5</attributeRelation>
          <parenCount>0</parenCount>
          <termRelation>0</termRelation>
          <value1>2</value1>
        </term>
      </terms>
      <waitTime>0</waitTime>
    </step>
  </steps>
</precisionQueue>
Scan API

The Scan API forces the validation of the SocialMiner external machine status rules. See Status API, on page 111.

URL

https://<server>/unifiedconfig/config/status/scan

HTTP method

PUT: Invoke the PUT method to force a scan to start immediately.
Single Sign-On Global State API

Use the Single Sign-On (SSO) Global State API to view or update the global status of SSO.

To retrieve the overall status of setting the SSO state or the status for a single component, see the Single Sign-On Status API, on page 103.

**URL**

https://<server>/unifiedconfig/config/sso/globalstate

**Operations**

- **get**: Returns the current global state of SSO in the database.
- **update**: Updates the global state of SSO in the database.

**Parameters**

- **refURL**: The RefURL. See Shared Parameters, on page 9.
- **changeStamp**: See Shared Parameters, on page 9.
- **permissionInfo**: Information about permissions.
  - **canUpdate**: Whether ssoState can be updated. True or false.
  - **role**: The role of the user.
- **state**: Required for update. Valid values are NON_SSO (SSO is disabled for all users), SSO (SSO is enabled for all users), and HYBRID (mix of enabled and disabled).

**Example Get Response**

```xml
<ssoState>
  <refURL>/ssostate</refURL>
  <changeStamp>227</changeStamp>
  <permissionInfo>
    <canUpdate>true</canUpdate>
    <role>Administrator</role>
  </permissionInfo>
  <state>SSO</state>
</ssoState>
```
**Single Sign-On Registration API**

Use the Single Sign-On (SSO) Registration API to register SSO-compatible components with the Cisco Identity Service. These components include AW, Finesse, and Unified Intelligence Center machines.

To retrieve the overall registration status or the status for a single component, see the [Single Sign-On Status API](#), on page 103.

**URL**

https://<server>/unifiedconfig/config/sso/register

**Operations**

- **update**: Registers all SSO-compatible components in the Machine Inventory with the Cisco Identity Service, using the URL `https://<server>/unifiedconfig/config/sso/register`. (See [Machine Inventory API](#), on page 55)

**Parameters**

None
CHAPTER 25

Single Sign-On Status API

Use the Single Sign-On (SSO) Registration API to get the current status of registering components with the Cisco Identity Service (IdS) and setting SSO state on the SSO-compatible components. These components include AW, Finesse, and Unified Intelligence Center machines.

URL

https://<server>/unifiedconfig/config/sso/status

Operations

- **get** component status: Returns the status of registering a specific component with the Cisco IdS and setting SSO state, using the URL
  https://<server>/unifiedconfig/config/sso/status/<id>. The <id> is the machine_id of the component.

- **list**: Retrieves a list with the overall and individual component statuses of registering SSO-compatible components with the Cisco Identity Service and setting SSO state, using the URL
  https://<server>/unifiedconfig/config/sso/status.

Parameters

- **globalSsoState**: The current SSO state as set in the AW database. The values are NON_SSO (SSO is disabled for all users), SSO (SSO is enabled for all users), and HYBRID (mix of enabled and disabled).

- **registrationState**: The overall status of registering components with the Cisco IdS. Values are the following:
  - SUCCEEDED: All of the components were successfully registered.
  - FAILED: Registration failed on one or more components. Error detail is set on each failed component.
  - PROCESSING: Registration started and is not complete.
  - NOT_STARTED: Registration has not started.

- **modeState**: The overall status of registering components with the Cisco IdS. Values are the following:
  - SUCCEEDED: The SSO state was successfully set on all of the components.
• FAILED: The SSO state failed to be set on one or more components. Error detail is set on each failed component.
• PROCESSING: The SSO state change has started and is not complete.
• NOT_STARTED: The SSO state change has not started.

• idSConfigurationState: Whether the Cisco IdS has been configured and is in service.
  • STATE_NOT_CONFIGURED: The Cisco IdS is not been configured.
  • STATE_IN_SERVICE: The Cisco IdS is configured and is in service.
  • STATE_OUT_OF_SERVICE: The Cisco IdS is configured and is out of service.
  • STATE_PARTIAL_SERVICE: The Cisco IdS is configured and is partially in service.
  • STATE_UNREACHABLE: The Cisco IdS cannot be reached.

• hasIdsCredentials: Whether the Machine Inventory has the necessary IdS credentials to register components (see Machine Inventory API, on page 55. Values are true or false. The default is false.
• idsBaseUrl: The base URL for accessing the Identity Service.
• ssoComponentStatuses: A collection of registration and SSO state status information for all of the individual components. Returned on a list operation.
• ssoComponentStatus: Registration and SSO state status information for an individual component. Includes the following parameters:
  • registrationState: The status of registering the component with the IdS. See the values for this parameter above.
  • modeState: The status of setting the SSO state for the component. See the values for this parameter above.
  • refURL: The refURL for the component machine. See Shared Parameters, on page 9.
  • name: The name of the component machine.

Example Get Response

Example URL: https://<server>/unifiedconfig/config/sso/status/21

<ssoComponentStatus>
  <registrationState>FAILED</registrationState>
  <modeState>NOT_STARTED</modeState>
  <refURL>/unifiedconfig/config/machineinventory/21</refURL>
  <name>PINESSE-A.boston.com</name>
</ssoComponentStatus>

Example List Response

<ssoStatus>
  <globalSsoState>HYBRID</globalSsoState>
  <registrationState>FAILED</registrationState>
  <idSConfigurationState>STATE_IN_SERVICE</idSConfigurationState>
  <hasIdsCredentials>true</hasIdsCredentials>
  <idsBaseUrl>https://<server>:<serverport></idsBaseUrl>
</ssoComponentStatuses>
<ssoComponentStatus>
  <registrationState>FAILED</registrationState>
  <modeState>NOT_STARTED</modeState>
  <refURL>/unifiedconfig/config/machineinventory/21</refURL>
  <name>FINESSE-A.boston.com</name>
</ssoComponentStatus>
<ssoComponentStatus>
  <registrationState>FAILED</registrationState>
  <modeState>NOT_STARTED</modeState>
  <refURL>/unifiedconfig/config/machineinventory/22</refURL>
  <name>FINESSE-B.boston.com</name>
</ssoComponentStatus>
<ssoComponentStatus>
  <registrationState>FAILED</registrationState>
  <modeState>NOT_STARTED</modeState>
  <refURL>/unifiedconfig/config/machineinventory/23</refURL>
  <name>CUIC-A.boston.com</name>
</ssoComponentStatus>
</ssoComponentStatuses>
Skill Group API

A skill group is a collection of agents who share a common set of competencies that equip them to handle the same types of requests. Some examples of skill groups are a collection of agents who speak a specific language or who can assist callers with billing questions.

Use the Skill Group API to list, view, or edit existing skill groups.

Note

Access to this API is different for supervisors and administrators. For more information, see Access, on page 4.

URL

https://<server>/unifiedconfig/config/skillgroup

Operations

• get: Returns one skill group, using the URL
  https://<server>/unifiedconfig/config/skillgroup/<id>.

• list: Retrieves a list of skill groups.
  * Query Parameters:
    * selectedAgentCount: Use this query parameter to augment skill group information about multiple agents. The selectedAgentCount parameter shows the number of specified agents belonging to that skill group. For example, to find out how many of agents 5000, 5001, 5002, and 5003 belong to each of the skill groups in the list, add selectedAgentCount=5000,5001,5002,5003.

  Note
  Using selectedAgentCount automatically sets the summary list query parameter to true.

  * Summary list: See list, on page 4.

• update: Updates one skill group.
**Parameters**

- **refURL:** The refURL of the skill group. See [Shared Parameters](#) on page 9.
- **name:** The name of the skill group. See [Shared Parameters](#) on page 9.
- **changeStamp:** See [Shared Parameters](#) on page 9.
- **description:** See [Shared Parameters](#) on page 9.
- **peripheral:** Contains the name and id of the peripheral.
- **mediaRoutingDomain:** A reference to the media routing domain ([Media Routing Domain API](#) on page 63) including the name and refURL. See [References](#) on page 6.
  - Defaults to Cisco_Voice MRD if this parameter is not provided.
  - This reference cannot be updated.
- **agents:** A collection of agents assigned to the skill group ([Agent API](#) on page 17). References also include firstName, lastName, agentId, and agentTeam (which includes the team name and refURL). See [References](#) on page 6.
  - **canRemove:** This parameter only appears for supervisors. It indicates whether or not the supervisor has permission to remove the agent from this skill group. The supervisor can remove the agent from the skill group if the agent belongs to a team of this supervisor.
- **agentsAdded:** A collection of agent references to be added to the skill group, including the refURL of each agent to be added. This parameter is update only, and cannot be used in conjunction with the agents parameter. This parameter can be used with the agentsRemoved parameter. See [References](#) on page 6.
- **agentsRemoved:** A collection of agent references to be removed from the skill group, including the refURL of each agent to be removed. This parameter is update only, and cannot be used in conjunction with the agents parameter. This parameter can be used with the agentsAdded parameter. See [References](#) on page 6.
- **agentCount:** Read-only parameter containing the number of agents having the skill.
- **selectedAgentCount:** Read-only field. Indicates the number of specified agents belonging to the skill group. Returned only when using the selectedAgentCount query parameter.
- **bucketInterval:** A reference to the bucket interval ([Bucket Interval API](#) on page 31). Includes the name and refURL. See [References](#) on page 6.
- **serviceLevelThreshold:** Maximum time in seconds that a caller should wait before being connected with an agent. Positive integers only, or blank.
- **serviceLevelType:** This value indicates how the system calculates the service level.
  - **1:** Ignore Abandoned Calls (default).
  - **2:** Abandoned Calls have Negative Impact.
  - **3:** Abandoned Calls have Positive Impact.
Search and Sort Values

The following table shows the parameters that are searched and the parameters that are sortable.

<table>
<thead>
<tr>
<th>Search parameters</th>
<th>Sort parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>• name</td>
<td>• name (default)</td>
</tr>
<tr>
<td>• description</td>
<td>• description</td>
</tr>
<tr>
<td></td>
<td>• serviceLevelThreshold</td>
</tr>
<tr>
<td></td>
<td>• serviceLevelType</td>
</tr>
<tr>
<td></td>
<td>• peripheralNumber</td>
</tr>
</tbody>
</table>

See Search, on page 11 and Sort, on page 12.

For more information on search restrictions, see Search, on page 11.

Advanced Search Parameters

- **peripherals**: (peri1|peri2|peri3, ...) which returns all SkillGroups that have any of the specified peripherals. Up to ten peripherals can be specified. The peripheral names are fully matched (case-insensitive, no partial matches).

Example Get Response

```xml
<skillGroup>
  <refURL>/unifiedconfig/config/skillgroup/(id)</refURL>
  <name>test</name>
  <description>test skill group</description>
  <changeStamp>0</changeStamp>
  <peripheral>
    <id>5001</id>
    <name>CUCM_PG1</name>
  </peripheral>
  <mediaRoutingDomain>
    <name>Cisco_Voice</name>
    <refURL>/unifiedconfig/config/mediaroutingdomain/1</refURL>
  </mediaRoutingDomain>
  <bucketInterval>
    <name>bucketIntervalName</name>
    <refURL>/unifiedconfig/config/bucketinterval/1</refURL>
  </bucketInterval>
  <serviceLevelThreshold>20</serviceLevelThreshold>
  <serviceLevelType>1</serviceLevelType>
  <agents>
    <agent>
      <refURL>/unifiedconfig/config/agent/5000</refURL>
      <firstName>Jane</firstName>
      <lastName>Doe</lastName>
      <userName>username</userName>
      <agentId>8007</agentId>
      <canRemove>true</canRemove>
    </agent>
    <agent>
      <refURL>/unifiedconfig/config/agent/5001</refURL>
      <firstName>John</firstName>
      <lastName>Smith</lastName>
      <userName>username2</userName>
      <agentId>8008</agentId>
      <agentTeam>
        <refURL>/unifiedconfig/config/agentteam/5000</refURL>
      </agentTeam>
    </agent>
  </agents>
</skillGroup>
```
<name>someTeam</name>
</agentTeam>
<canRemove>false</canRemove>
</agent>
<agent>...</agent>
<agent>...</agent>
</agents>
<agentCount>4</agentCount>
</skillGroup>
Status API

The Status API retrieves information about the state of rules that collectively determine if the system is working correctly.

- Configuration Rules, on page 112
- Operation Rules, on page 113

The system periodically checks the state of the rules. To force the status to be updated, refer to the Scan API, on page 97.

**URL**

https://<server>/unifiedconfig/config/status

**Operations**

- **get**: Returns the status rule results, using the URL https://<server>/unifiedconfig/config/status.

**Response Parameters**

- **name**: The name of the status. See the Configuration Rules, on page 112 and Operation Rules, on page 113.
- **category**: The status category of Configuration or Operation.
- **level**: The severity of the condition. Values include: OK, ERROR, and BLOCKED. The BLOCKED level indicates that the rule has not been processed yet or that the failure of another rule prevents this rule from running.
- **machine**: A collection for the machine (Machine Inventory API, on page 55) including the machine's name, type, and refURL. See References, on page 6.

**Example Get Response**

```
<results>
  <statuses>
    <status>
      <name>SOCIAL_MINER_MR_ENABLED</name>
      <category>CONFIGURATION</category>
    </status>
  </statuses>
</results>
```
Configuration Rules

Common Rules for All Machines

These rules show the potential configuration errors and warnings for all machines.

Table 1: Common Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_RESOLVE_ADDRESS_TO_FQDN</td>
<td>Machine address must be resolvable to a fully qualified domain name (FQDN).</td>
</tr>
</tbody>
</table>
Unified CCE AW, Unified Intelligence Center, CUIC-LD-IdS, and Finesse Machines

These rules show the potential configuration errors and warnings for Unified CCE AW, Unified Intelligence Center, CUIC-LD-IdS, and Finesse machines.

Table 2: Unified CCE AW, Unified Intelligence Center, CUIC-LD-IdS, and Finesse Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO_COMPONENT_STATUS_MATCHES_GLOBAL</td>
<td>The global SSO status must be the same as component SSO status.</td>
</tr>
<tr>
<td>SSO_VALID_IDS_REFERENCE</td>
<td>If single sign-on is enabled, this machine must be associated with a valid Cisco Identity Service.</td>
</tr>
</tbody>
</table>

Table 3: Unified CCE AW Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK_ROUTING_APP_PATHS_EXIST</td>
<td>Each Peripheral Gateway with a Communications Manager PIM must have an associated Task Routing Application Path.</td>
</tr>
</tbody>
</table>

SocialMiner Machines

These rules show the potential configuration errors and warnings for SocialMiner machines.

Table 4: SocialMiner Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCIAL_MINER_MR_ENABLED</td>
<td>Multichannel routing must be enabled on SocialMiner.</td>
</tr>
</tbody>
</table>

Operation Rules

Common Operation Rules

These rules show the potential operation errors and warnings for the following machines:

- All Unified CCE components
- Unified Intelligence Center
- CUIC-LD-IdS
- Finesse
- Live Data
- Identity Service
- SocialMiner
## Table 5: Common Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVER_CONNECTION</td>
<td>The Diagnostic Portal, AXL, REST, or SOAP service on this machine must be in service.</td>
</tr>
<tr>
<td>SERVER_CREDENTIALS</td>
<td>The inventory's credentials for the machine must be valid.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>  This rule does not apply to the Live Data machine.</td>
</tr>
</tbody>
</table>
INDEX

C
creating  81
Do Not Call list  81

D
Do Not Call list  81
creating  81