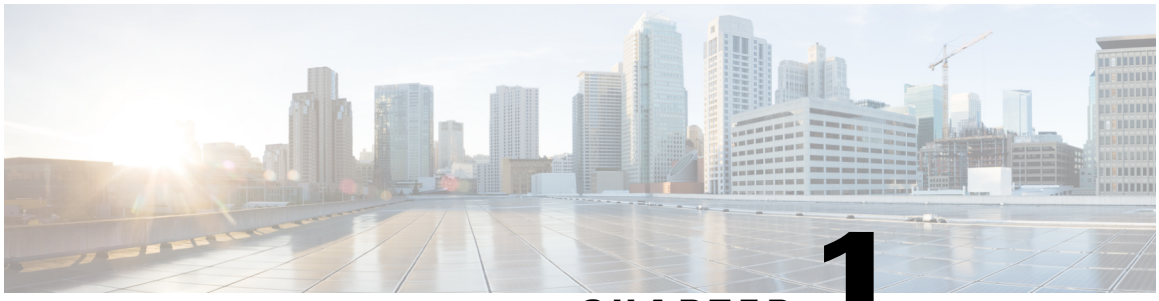
# Cisco Unified Contact Center Enterprise Developer Reference Release 10.0(x)

**First Published:** December 12, 2013

**Americas Headquarters**
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

# CONTENTS



CHAPTER 1

# 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 1.

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

To review examples on how to interact with the REST web service in Java, see the CCE Config Sample REST Toolkit.



# 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 3.

### 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.

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>:<serverport>/unifiedconfig/config/bucketinterval`. See also Permissions, on page 6, Pagination, on page 4, Search, on page 8, and Sort, on page 8.

**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>:<serverport>/unifiedconfig/config/bucketinterval/(id)` with a the following body:

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

# Usage and behavior

### Duplicate parameters

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

### Read-only fields

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

### References

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

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

```
<agent>
   <team>
     <refURL>/config/team/5000</refURL>
     <name>NameOfTeam</name>
   </team>
  ...
</agent>
```

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

```
<agent>
  <refURL>https://10.10.10.5/unifiedconfig/config/agent/5000</refURL>
  <firstName>Jane</firstName>
  <lastName>Doe</lastName>
  <userName>username</userName>
  <agentId>8007</agentId>
  <canRemove>true</canRemove>
</agent>
```

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

```
<agent>
   <team>
     <refURL>/config/team/5000</refURL>
   </team>
  ...
</agent>
```

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

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

### XML

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

# Error responses

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

### Parameters

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

### Example error response

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

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

# Pagination

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

### Query parameters

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

- resultsPerPage: Specifies the number of items to retrieve. Minimum:1. Default: 25. Maximum: 100.

### Returned parameters

- totalResults: Total number of items.
- resultsPerPage: Number of items requested per page.
- startIndex: The index of the first item returned. If you request a startIndex that is greater than total items, a full last page is returned.
- nextPage: The URL to get the next page. This parameter is not returned if you are on the last page.
- prevPage: The URL to get the previous page. This parameter is not returned if you are on the first page.
- firstPage: The URL to get the first page.
- lastPage: The URL to get the last page.
- searchTerm: The value specified in the search query parameter. See Search, on page 8.
- sortTerm: The value specified in the sort query parameters. See Sort, on page 8.

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

### Example response

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

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

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

# Shared parameters

### changeStamp

- The version of the item. Initially set during a create (create, on page 1) operation.
- A changeStamp is a required parameter for the body of a PUT (update, on page 2) 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 Installing and Configuring Cisco Packaged Contact Center Enterprise.
- Maximum length of 255 characters.

### name

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

### refURL

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

# Permissions

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

### Parameters

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

### Example get response

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

# Synchronous vs. Asynchronous Writes

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

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

### Performing asynchronous operations

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

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

### Asynchronous result parameters

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

### Polling the asynchronous request status

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

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

# Search

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

### Default search parameters

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

Values:

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

### Advanced search

Advanced search parameters allow specific parameters to be searched. Refer to each API section for the advanced search parameters permitted. Advanced search parameters can be combined with a default search value. For example, applying the search query parameter of `q=abc routingType:1` to a dialed number list operation returns results where the routingType is set to one, and one of the default search parameters contains **abc**. Advanced search also has the following restrictions:

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

# Sort

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

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

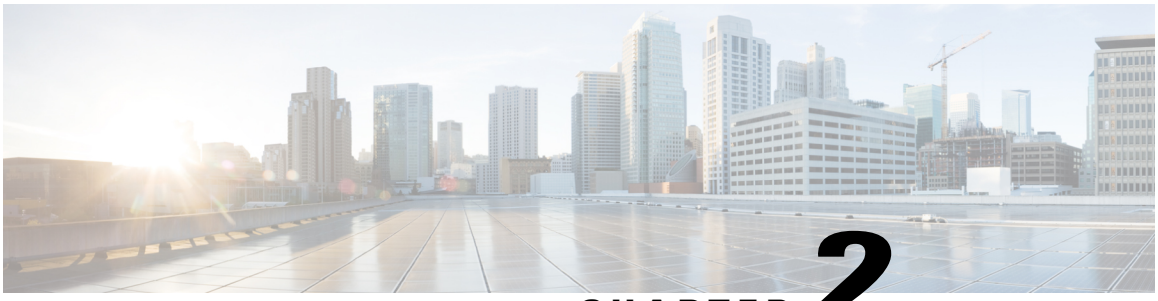
- parameterName: The name of the parameter that you want to sort on. This is case sensitive, so it must match the parameter in the API exactly.
- order: Specifies the order of the sort. Values are as follows:
  - asc: Perform an ascending sort. This is the default if no order is specified.
  - desc: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

### Example

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

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

Sort



CHAPTER 2

# 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

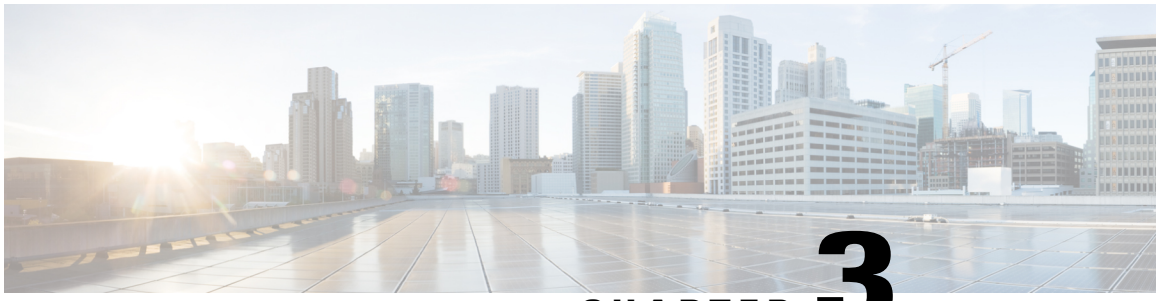
- list: Retrieves a list of active directory domains.

### Parameters

- activeDirectoryDomains: A collection of activeDirectoryDomain items, including a name parameter.

### Example get response

```
<results>
  <activeDirectoryDomains>
    <activeDirectoryDomain>
       <name>boston.com</name>
    </activeDirectoryDomain>
    <activeDirectoryDomain>
       <name>cisco.com</name>
    </activeDirectoryDomain>
  </activeDirectoryDomains>
</results>
```



CHAPTER 3

# 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.

### 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:**
        - Summary list: See list, on page 2.
- update: Updates one agent.

### Parameters

- refURL: The refURL for the agent. See Shared parameters, on page 5.
- agentId: The unique peripheral number. Maximum length of 11 characters allowed. Default is an auto-generated 7 digit number.
- changeStamp: See Shared parameters, on page 5.
- description: See Shared parameters, on page 5.
- agentStateTrace: Indicates if agent state tracing is turned on for the agent. Values are true/false.
- agentDeskSettings: A reference to the agent's agentDeskSettings, including the refURL and name. See References, on page 3.
- 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.

- userName: Agent's user name. Maximum of 32 alphanumeric characters.
- 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.

- supervisor: Required. Indicates whether the agent is marked as supervisor. Values are true/false.
- supervisorUserInfo: Required if supervisor is set to true. User information about an existing Active Directory account for the supervisor. Includes the following parameters:
  - userName: Supervisor's Active Directory user name.
  - domainName: Supervisor's Active Directory (Active Directory Domain API, on page 11) domain name. If domainName is empty, system uses default domain name.
- agentAttributes: A collection of agent attribute (Attribute API, on page 17) 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/false or 1-10) of the attribute for this agent. See References, on page 3.
- skillGroups: A collection of skill group references for this agent, including the refURL and name of each associated skill group. See References, on page 3.
- 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 in conjunction 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 3.
- 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 in conjunction 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 3.
- 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 3.
- agentTeam: A reference to the agent's team, including the refURL and name. See References, on page 3.
- 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 3.

### Search and sort values

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

| Search parameters | Sort parameters |
|---|---|
| • agentId<br>• description<br>• person.firstName<br>• person.lastName<br>• person.userName | • agentId<br>• description<br>• supervisor<br>• agentStateTrace<br>• person.firstName<br>• person.lastName<br>• person.userName<br>• person.loginEnabled |

See Search, on page 8 and Sort, on page 8.

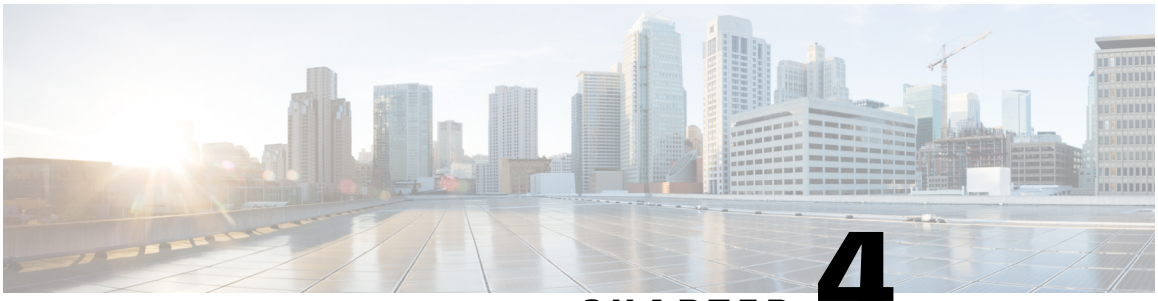
### Example get response

```
<agent>
 <changeStamp>2877</changeStamp>
        <refURL>/unifiedconfig/config/agent/5017</refURL>
        <agentId>8006</agentId>
        <agentStateTrace>false</agentStateTrace>
        <description>an agent</description>
        <person>
             <firstName>Agent2</firstName>
             <lastName>Agent2</lastName>
             <loginEnabled>true</loginEnabled>
             <userName>Agent2</userName>
             <password>mypassword</password>
       </person>
        <agentDeskSettings>
             <name>test2</name>
             <refURL>/unifiedconfig/config/agentdesksetting/5434</refURL>
        <supervisor>true</supervisor>
        </agentDeskSettings>
        <supervisorUserInfo>
              <userName>boston</userName>
              <domainName>boston.com</domainName>
        </supervisorUserInfo>
        <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>
         <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>
```



CHAPTER 4

# 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 for deletion, but does not permanently delete it.
- get: Returns one attribute, using the URL `https://<server>:<serverport>/unifiedconfig/config/attribute/<id>`.
- list: Retrieves a list of attributes.
- update: Updates one attribute.

### Parameters

- refURL: The refURL of the attribute. See Shared parameters, on page 5.
- name: The name of the attribute. See Shared parameters, on page 5.
- changeStamp: See Shared parameters, on page 5.
- description: See Shared parameters, on page 5.
- 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.

### Search and sort values

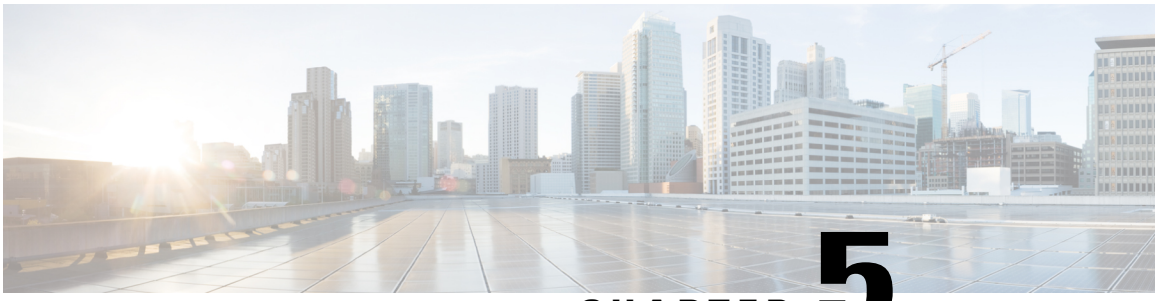
The following table shows the parameters that are searched and the parameters that are sortable.

| Search parameters | Sort parameters |
|---|---|
| • name<br>• description | • name (default)<br>• dataType<br>• defaultValue<br>• description |

See Search, on page 8 and Sort, on page 8.

### Example get response

```
<attribute>
      <changeStamp>0</changeStamp>
      <refURL>/unifiedconfig/config/attribute/5000</refURL>
      <dataType>3</dataType>
      <defaultValue>true</defaultValue>
      <name>chinese</name>
</attribute>
```



CHAPTER 5

# 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 5.
- name: The name of the bucket interval. SeeShared parameters, on page 5.
- changeStamp: See Shared parameters, on page 5.
- 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

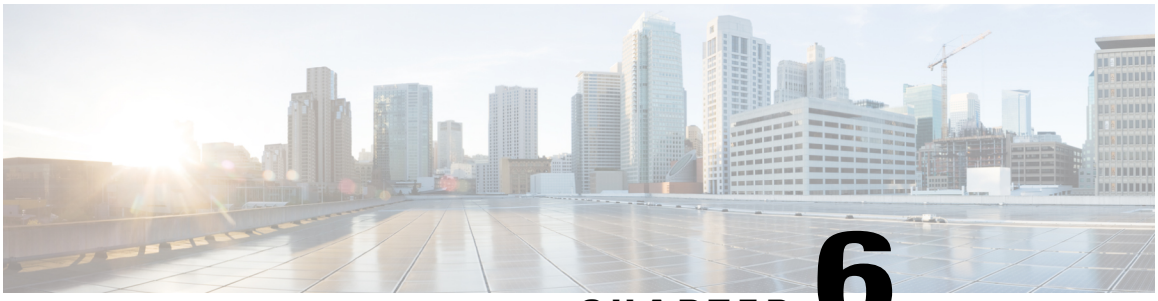
The following table shows the parameters that are searched and the parameters that are sortable.

| Search parameters | Sort parameters |
|---|---|
| • name | • name (default)<br>• upperBound 1-9 |

See Search, on page 8 and Sort, on page 8.

### Example get response

```
<bucketInterval>
    <refURL>http://***.***.***.***/unified/config/bucketInterval/(id)</refURL>
    <name>test</name>
    <department>
        <name>sales</name>
        <refURL>https://host/unifiedconfig/config/department/5003</refURL>
    </department>
    <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>
```



CHAPTER 6

# 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

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

### Operations

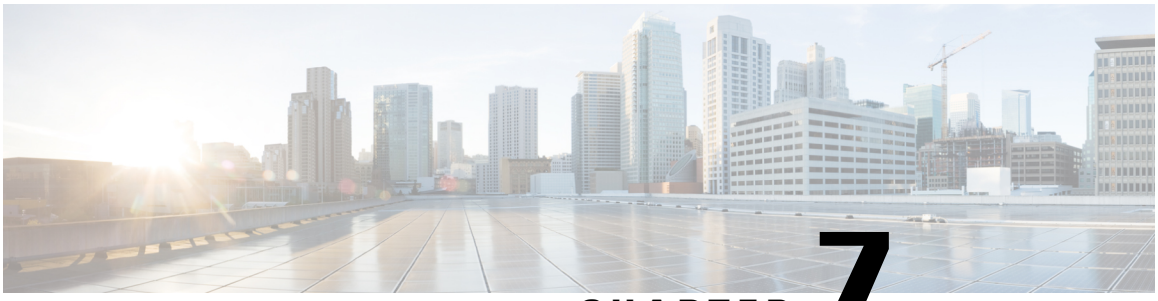
- get: Returns the congestion control parameters, using the URL `https://<server>:<serverport>/unifiedconfig/config/congestioncontrol`.
- update: Updates the congestion control parameters.

### Parameters

- deploymentType: The type of deployment. See Deployment Type Info API, on page 25.
- 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

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



CHAPTER 7

# 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>:<serverport>/unifiedconfig/config/deployment`

### Parameters

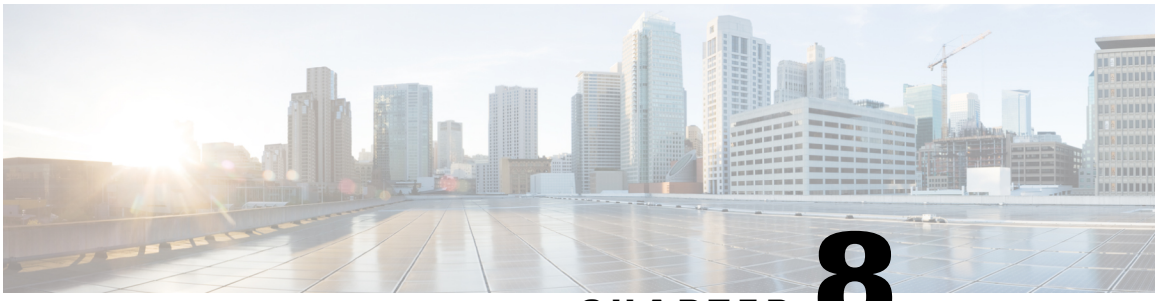
deploymentType: See Deployment Type Info API, on page 25.

### Operations

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

### Example get response

```
<deployment>
     <deploymentType>7</deploymentType>
</deployment>
```



CHAPTER 8

# 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

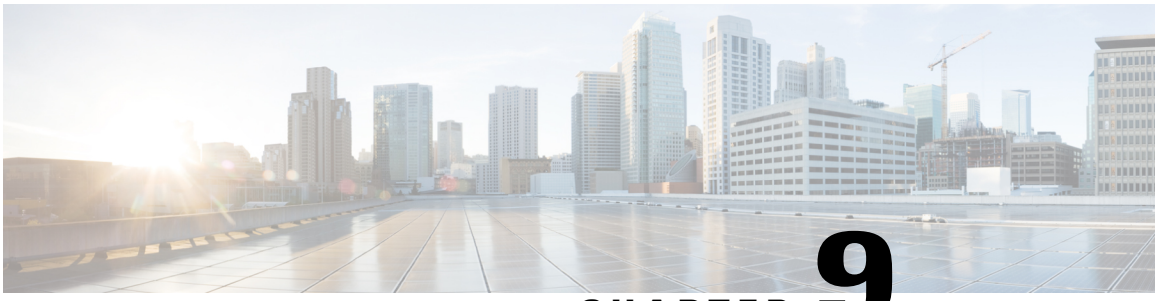
- changeStamp: See Shared parameters, on page 5.
- vmHosts: vmHost information, including name, address, username, and password parameters of Side A and Side B. Only required when switching to Packaged CCE, to allow access to the ESX servers for VM validation.
- permissionInfo: See Permissions, on page 6.
- 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
    - 2: IVR-ICM
    - 3: NAM Rogger
    - 4: ICM Router/Logger
    - 5: UCCE 8000 Agents Router/Logger
    - 6: UCCE 12000 Agents Router/Logger
    - 7: Packaged CCE: CCE-PAC-M1
    - 8: ICM Rogger
    - 9: UCCE 4000 Agents Rogger
    - 10: Packaged CCE: CCEPACM1 Lab only
    - 11: HCS-CC 1000 Agents
    - 12: HCS-CC 500 Agents
    - 13: UCCE 450 Agents Progger
    - 14: HCS-CC 4000 Agents

### 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>
      <userName>root</userName>
      <password>pwexample</password>
    </vmHost>
  </vmHosts>
</deploymentTypeInfo>
```



CHAPTER 9

# Dialed Number API

Dialed numbers are string values used to select the appropriate routing script so that a voice call or a non-voice task (such as an email or a request for a web chat) can be delivered to an agent.

Use the Dialed Number API to list the dialed numbers currently defined in the database, define new dialed numbers, and view, edit, and delete existing dialed numbers.

### URL

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

### Operations

- get: Returns one dialed number, using the URL `https://<server>/unifiedconfig/config/dialednumber/<id>`.
- list: Retrieves a list of Multichannel dialed numbers.

### Parameters

- dialedNumberString: Required. Value used to route the call or direct the non-voice task. A unique string for the routing type. Maximum of 25 characters.
- changeStamp: See Shared parameters, on page 5.
- description: See Shared parameters, on page 5.
- routingType: Specifies where a call or non-voice task request originates.
    - 1: External Voice. Calls come from Unified CVP. When creating a Dialed Number using this type, a dialed number database record is created for each Unified CVP routing client.
    - 2: Internal Voice. Calls come from a Unified CM phone.
    - 3: Outbound. Calls that come from the Outbound Option Dialer.
    - 4: Multichannel. Requests that come from an EIM/WIM or SocialMiner.
- dialedNumberRecords: A collection of dialed number record entries each containing the id and name of a dialed number database record. Read-only.

- mediaRoutingDomain: A reference to the media routing domain (Media Routing Domain API, on page 29) for the dialed number. See References, on page 3.

### Search and sort values

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

| Search parameters | Sort parameters |
|---|---|
| • dialedNumberString<br>• description | • dialedNumberString (default)<br>• description |

See Search, on page 8 and Sort, on page 8.

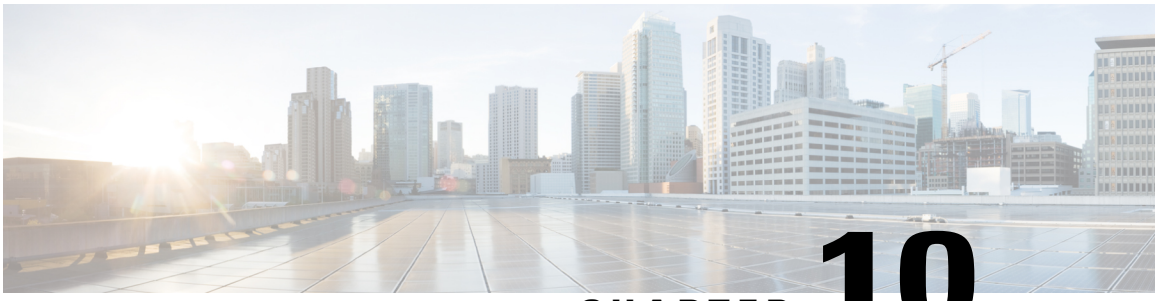
**Advanced search parameters**

The Dialed Number API also supports advanced search parameters, such as routing type.

- **routingType:<type>** Finds all dialed numbers with the specified routing type value. Valid types match those in the routingType parameter.
  - **routingType:1** Returns all dialed numbers with an external voice routing type.

### Example get response

```
<dialedNumber>
  <refURL>[/unifiedconfig/config/dialedNumber/(id)]</refURL>
  <description>test dialed number</description>
  <dialedNumberString>8885551212</dialedNumberString>
  <routingType>1</routingType>
  <changeStamp>0</changeStamp>
  <mediaRoutingDomain>
    <refURL>[/unifiedconfig/config/mediaroutingdomain/1]<refURL>
    <name>Cisco_Voice</name>
  </mediaRoutingDomain>
  <callType>
    <refURL>[/unifiedconfig/config/calltype/(id)]</refURL>
    <name>calltype name</name>
  </callType>
  <dialedNumberRecords>
    <dialedNumberRecord>
      <id>10</id>
      <name>cvp1rc.8885551212</name>
    </dialedNumberRecord>
    <dialedNumberRecord>
      <id>11</id>
      <name>cvp2rc.8885551212</name>
    </dialedNumberRecord>
    <dialedNumberRecord>
      <id>12</id>
      <name>cvp3rc.8885551212</name>
    </dialedNumberRecord>
    <dialedNumberRecord>
      <id>13</id>
      <name>cvp4rc.8885551212</name>
    </dialedNumberRecord>
  </dialedNumberRecords>
</dialedNumber>
```



CHAPTER 10

# 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.

### URL

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

### Operations

- list: Retrieves a list of media routing domains.
- get: Returns one media routing domain using the URL `https://<server>/unifiedconfig/config/mediaroutingdomain/<id>`.

### Parameters

- refURL: The refURL of the media routing domain. See Shared parameters, on page 5.
- name: Name of the media routing domain. See Shared parameters, on page 5.
- description: See Shared parameters, on page 5.
- mediaClass: Includes the following parameters:
  - name: The name of the media class.
  - id: The ID of the media class.
- serviceLevelThreshold: Value in seconds within which calls must be answered.
- interruptible: Indicates if an agent can be interrupted by assigned tasks from another MRD. Values are true/false.
- maxCallsInQueue: The maximum number of calls allowed to be queued at one time.
- maxCallsInQueuePerCallType: The maximum number of calls allowed to be queued, per call type.
- maxTimeInQueue: The maximum amount of time, in seconds, a call can queued.

### Search and sort values

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

| Search parameters | Sort parameters |
|---|---|
| • name<br>• description | • name (default)<br>• description<br>• interruptible<br>• maxCallsInQueue<br>• maxCallsInQueuePerCallType<br>• maxTimeInQueue |

See Search, on page 8 and Sort, on page 8.

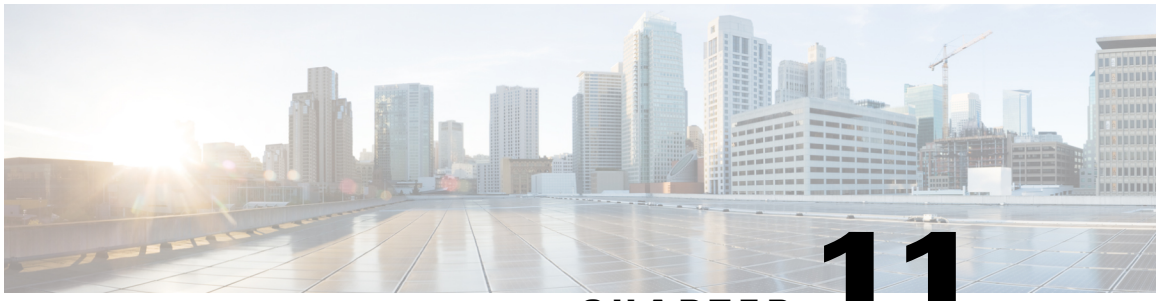
**Advanced search parameters**

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 media routing domain. For example, **q=nonVoiceOnly:true**.

### Example get response

```
<mediaRoutingDomains>
  <mediaRoutingDomain>
     <changeStamp>0</changeStamp>
     <refURL>/unifiedconfig/config/mediaroutingdomain/5001</refURL>
     <description>System-provided media routing domain for Cisco_Chat</description>
     <interruptible>false</interruptible>
     <maxCallsInQueue>1000</maxCallsInQueue>
     <maxCallsInQueuePerCallType>1000</maxCallsInQueuePerCallType>
     <maxTimeInQueue>1000</maxTimeInQueue>
     <mediaClass>
       <name>Cisco_Chat</name>
       <id>6</id>
     </mediaClass>
     <name>Cisco_Chat</name>
     <serviceLevelThreshold>30</serviceLevelThreshold>
  </mediaRoutingDomain>
  <mediaRoutingDomain>
     <changeStamp>0</changeStamp>
     <refURL>/unifiedconfig/config/mediaroutingdomain/1</refURL>
     <description>Default Media Routing Domain for Cisco_Voice</description>
     <interruptible>false</interruptible>
     <mediaClass>
       <name>Cisco_Voice</name>
       <id>4</id>
     </mediaClass>
     <name>Cisco_Voice</name>
     <serviceLevelThreshold>30</serviceLevelThreshold>
   </mediaRoutingDomain>
 </mediaRoutingDomains>
```



CHAPTER 11

# 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 **Boston = True** and **Spanish = 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>:port/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>:port/unifiedconfig/config/precisionqueue/<id>`.
    - **Query parameters:**
        - agentcount: Use this query parameter to have the agent count parameter included in the response.
        - attributes: Use this query parameter to have the attribute parameter included in the response.
- 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 5.

- name: The name of the precision queue. See Shared parameters, on page 5.
- changeStamp: See Shared parameters, on page 5.
- description: See Shared parameters, on page 5.
- bucketInterval: A reference to a bucket interval (Bucket Interval API, on page 19), including the refURL and name. See References, on page 3.
- 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.
- 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 17), including the refURL, name, description, and dataType. A maximum of 5 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.

| Search parameters | Sort parameters |
|---|---|
| • name<br>• description | • name (default)<br>• description |

See Search, on page 8 and Sort, on page 8.

### Example get response

```
<precisionQueue>
    <changeStamp>4</changeStamp>
    <refURL>/unifiedconfig/config/precisionqueue/5002</refURL>
    <agentOrdering>1</agentOrdering>
    <bucketInterval>
        <refURL>/unifiedconfig/config/bucketinterval/1</refURL>
        <name>Default_Bucket_Intervals</name>
    </bucketInterval>
     <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>
            <step>
                <terms>
                    <term>
                        <attribute>
                        <refURL>/unifiedconfig/config/attribute/5698</refURL>
                        <name>test</name>
                        <dataType>4</dataType>
                        </attribute>
                        <attributeRelation>3</attributeRelation>
                        <parenCount>0</parenCount>
                        <termRelation>0</termRelation>
                        <value1>2</value1>
                    </term>
                </terms>
                <waitTime>-1</waitTime>
            </step>
        </steps>
    </precisionQueue>
```