Web Services API Reference

Settings

4/22/2025

Settings

Contents

- 1 Settings
  - 1.1 Operations
  - 1.2 Attributes
  - 1.3 Example
  - 1.4 Supported Settings Groups
  - 1.5 User-Defined Setting Groups

The *settings* resource is intended for configuration tasks that modify the behavior of existing functionality. This resource contains a list of URIs that correspond to named settings groups. Each of these groups has its own attributes and security settings.

Operations

The following operations are supported by **/settings**:

Operation	Description	Permissions
GET	Returns a list of all available settings groups for the contact center	Contact Center Admin
POST	Creates a new settings group for the contact center	Contact Center Admin
DELETE	Removes the settings group from the contact center. Note that only settings groups that have been created via the API (for example, using `POST /settings`) can be deleted.	Contact Center Admin

Attributes

The following attributes are supported for each item that is returned by **GET /settings**:

Attribute	Type	Description	Access
uri	String	The URI to the settings group.	GET
displayName	String	Name that describes the settings group.	GET, POST
name	String	A URI-compatible name for the settings group. This name will be used as part of the URI used to access the group: (for example, `GET /settings/my-settings-group`)	GET, POST
key	String	The name of the key attribute for this group's settings. Whenever an individual setting needs to be modified, this key attribute will be used to identify the setting. The	POST

Attribute	Type	Description	Access
		value of the key attribute must be unique for every setting and is read-only after the setting has been created. A setting may not be created without this attribute.	

Example

The following sample returns a list of all settings groups for the contact center:

```
GET /api/v2/settings

{
        settings:[{
                "displayName":"Agent States",
                "uri":"http://.../api/v2/settings/agent-states"
        },{
                "displayName":"Dispositions",
                "uri":"http://.../api/v2/settings/dispositions"
        }]
}
```

To create a new setting group:

```
POST /cloud-web/api/v2/settings

{
    "displayName":"My Setting Group",
    "name":"my-setting-group",
    "key":"name" //specifies that each setting in this group must have a "name" attribute
with a unique value
}
```

Important

Settings groups are different from features. Enabling a feature for the contact center may result in particular settings groups becoming available.

Each setting group may return the attribute "key" along with a setting array. This attribute specifies which of the setting attributes should be used as a key to identify the setting during modification (PUT) requests. If the "key" attribute is not present, "name" is the default identifying attribute.

Supported Settings Groups

The supported Settings Groups are General and Agent States.

General

This group is available under the following URI: `http://<host:port>/api/v2/settings/general-settings`. It contains the following attributes:

Attribute	Type	Description
countryCode	String	A two-character country code for the Contact Center.
countryDigits	String	A numerical country prefix for phone numbers.
countryName	String	Country name.

Real-Time Reporting

The real-time reporting settings group is available under the following URI: `/api/v2/settings/reporting`. It contains following settings described below:

Setting	Description	Default Value	Permitted Values
defaultServiceLevelInterval	Specifies the maximum time (in seconds) it should take an agent to answer a call. The percentage of answered calls that were answered under this threshold can be retrieved via the statistics API. This settings has the following rule of evaluation: if setting not set explicitly, the default value from statistics.yaml is returned. If corresponding statistic/property not found - the serviceLevel threshold property is considered to be not set.	`TimeRangeRight` property of `statisticDefinitionEx` of ServiceLevel statistic defined for QUEUE object type. It is set to `60` (seconds) in `statistics.yaml` file shipped within IP. If set, this setting affects `ServiceLevel` statistic defined for all queues and skills in contact center.	Any positive integer. **Note:** changing this setting will cause stat server to calculate a new statistic, resetting statistics.
defaultTargetServiceLevelPercentage	this setting allows to store/retrieve value to be used in UI as default target service level. Does not affects reporting functionality on server side.	If not explicitly set, default value is `80` (not configurable).	Any integer between 1 and 100 (inclusive).

Supported operations:

Operation	Description
PUT	updates the value of option
GET	returns the list of settings

Note: `POST` and `DELETE` are not supported.

GET Example:

```
GET .../api/v2/settings/reporting
returns:
    {
       "statusCode": 0,
       "settings":
       [
           {
               "name": "defaultTargetServiceLevelPercentage",
               "value": "80"
           },
           {
               "name": "defaultServiceLevelInterval",
               "value": "60"
           }
       ],
       "key": "name"
    }
```

PUT Example for `defaultServiceLevelInterval`

```
PUT .../api/v2/settings/reporting
with body:
{
           "name": "defaultServiceLevelInterval",
           "value": "115"
}
```

PUT sample for defaultServiceLevelPercentage

```
PUT .../api/v2/settings/reporting
with body:
{
           "name": "defaultTargetServiceLevelPercentage",
           "value": "99"
}
```

Agent States

Agent state is accessible by using **/settings/agent-states**. It allows the Contact Center Admin to define custom agent states that include reason codes.

Operations

The following operations are supported for the **/settings/agent-states** resource:

Operation	Description	Permissions
GET	Returns a list of all available agent states for the contact center	Contact Center Admin, Agent
POST	Creates a new agent state description.	Contact Center Admin
PUT	Modifies an existing agent state.	Contact Center Admin
DELETE	Removes an agent state description from the system.	Contact Center Admin

Attributes

The following attributes are supported for each agent state descriptor:

Attribute	Type	Description	Access	Required
id	String	The unique ID (GUID) for the agent state. This ID is included in the userState of device change messages when an agent state is matched.	GET	Yes
operationName	String	The unique operation name that is used to set this state (for example, OutToLunch).	GET, POST, PUT	Yes
displayName	String	The name for the state.	GET, POST, PUT	Yes
state	Enum	The actual T-Server state (Ready/NotReady).	GET, POST, PUT	Yes
workMode	Enum	An after call work mode. Note that modes are applicable to particular states. For Ready: ManualIn/AutoIn/ReturnBack. For NotReady: AfterCallWork/AuxWork/LegalGuard/NoCallDisconnect/	GET, POST, PUT	N

Attribute	Type	Description	Access	Required
		WalkAway. This should be enforced by the API.		
reason	String	The reason for the agent's state (if specified, it must be unique as it is used as a reason code).	GET, POST, PUT	N

Examples

Each contact center initially has five agent state descriptions for the basic Ready/Not Ready/Offline operations that are currently in use. These five operations are read only. They cannot be deleted or modified:

```
{ "key" : "operationName",
  "settings" : [ { "displayName" : "AfterCallWork",
        "id" : "D3663509-3D82-4DD3-A82E-2EA8EFA02AEF",
        "operationName" : "AfterCallWork",
        "state" : "NotReady",
        "workMode" : "AfterCallWork"
      },
      { "displayName" : "AuxWork",
        "id" : "2B36138D-C564-4562-A8CB-3C32D564F296",
        "operationName" : "AuxWork",
        "state" : "NotReady",
        "workMode" : "AuxWork"
      },
      { "displayName" : "Not Ready",
        "id" : "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
        "operationName" : "NotReady",
        "state" : "NotReady"
      },
      { "displayName" : "Offline",
        "id" : "0F7F5003-EF26-4D13-A6Ef-D0C7EC819BEB",
        "operationName" : "Offline",
        "state" : "Logout"
      },
      { "displayName" : "Ready",
        "id" : "9430250E-0A1B-421F-B372-F29E69366DED",
        "operationName" : "Ready",
        "state" : "Ready"
      }
    ],
  "statusCode" : 0
}
```

To add a new "Not Ready" state called "Out to lunch":

```
POST /settings/agent-states

{
    "operationName":"OutToLunch",
    "displayName":"Not Ready - Out to lunch",
    "state":"NotReady",
    "reason":"OutToLunch"
}
```

To modify the display name of the "OutToLunch" state described above:

```
PUT /settings/agent-states

{
    "operationName":"OutToLunch",
    "displayName":"Not Ready - Lunch!"
}
```

User-Defined Setting Groups

Operations

The following operations are available for user-defined groups via the **/settings/{group-name}** URI:

Operation	Description	Permissions
GET	Retrieves an array of settings in this group	Contact Center Admin, Supervisor, Agent
POST	Creates a new setting in this group. Must include the "key" attribute specified when the group was created	Contact Center Admin
PUT	Updates a specific setting. Must include the "key" attribute specified when the group was created in order to identify the setting to update.	Contact Center Admin
DELETE	Removes a setting. Must include the "key" attribute to identify the setting.	Contact Center Admin

Attributes

The attributes for each setting group vary. There is no limitation as to the number of attributes defined or the values they contain -- beyond that the values must contain valid json. One important thing to note is that if you have an attribute which holds a json object, you will not be able to modify the individual fields in the object. To modify a specific field, the whole object must be passed via **PUT** overwriting the existing value (see examples below).

Storage

User-defined settings groups created using this API are only stored in Cassandra and are not synchronized to Configuration Server. Configuration Server-defined settings groups will continue to be imported.

Examples

We will use disposition codes as an example of a custom setting group. Disposition codes are used by agents to specify how a given call was completed. Note that this illustrates a sample configuration that might be used by a client to configure disposition codes. Disposition code configuration is not included in the API.

For the purposes of this example let's assume that a group called "dispositions" has been created. The "key" attribute is "name".

1. Create a disposition code with an attribute that contains a complex structure (possibleValues)

```
POST /settings/dispositions
{
        "name":"department" //key attribute
        "displayName":"Department"
        "possibleValues":[
             {
                 "name":"tech_support"
                 "displayName":"Tech Support"
                 "possibleValues":[
                      {
                           "displayName":"Computers"
                           "name":"computers"
                      },
                      {
                           "displayName":"Network"
                           "name":"network"
                      }
                  ]
             },
             {
                 "displayName":"Sales"
                 "name":"sales"
             }
        ]
}
```

2. Update the displayName "Computers" to "Computers!!!"

```
PUT /settings/dispositions
{
        "name":"department" //must include key attribute to identify setting
        "possibleValues":[
             {
                 "name":"tech_support"
                 "displayName":"Tech Support"
                 "possibleValues":[
                      {
                           "displayName":"Computers!!!"
                           "name":"computers"
                      },
                      {
                           "displayName":"Network"
                           "name":"network"
                      }
                  ]
             },
             {
                 "displayName":"Sales"
                 "name":"sales"
             }
        ]
}
```

Note we must re-send the entire structure with the updated `displayName`.

3. Update displayName "Department" to "Department!"

```
PUT /settings/dispositions
{
        "name":"department", //key attribute
        "displayName":"Department!"
}
```

For top level settings it's enough to include the attribute (for example, `displayName`) and its new value along with the key to identify the setting.

4. Create another disposition...

```
POST /settings/dispositions
{
        "name":"helplevel", //key attribute
        "displayName":"Help Level"
        "possibleValues":[{"displayName":"Fixed!", "name":"fixed"}, {"displayName":"Not
Fixed", "name":"not_fixed"}]
}
```

...and GET everything:

```
GET /settings/dispositions
{
"statusCode":0,
"key":"name",
"settings":[
    {
        "name":"department"
        "displayName":"Department!"
        "possibleValues":[
             {
                 "name":"tech_support"
                 "displayName":"Tech Support"
                 "possibleValues":[
                      {
                           "displayName":"Computers!!!",
                           "name":"computers"
                      },
                      {
                           "displayName":"Network"
                            "name":"network"
                      }
                  ]
           },
           {
               "displayName":"Sales"
               "name":"sales"
           }]
    },
    {
        "name":"helplevel",
        "displayName":"Help Level"
        "possibleValues":[{"displayName":"Fixed!", "name":"fixed"}, {"displayName":"Not
Fixed", "name":"not_fixed"}]
    }]
}
```