



# API Reference

Workspace Web Edition & Web Services 8.5.2

# Table of Contents

<b>Web Services API Reference</b>	<b>10</b>
<b>Samples</b>	<b>12</b>
Getting Started	13
Get Version	14
Get User Information	17
Cross Site Request Forgery Protection Example	21
<b>General</b>	<b>34</b>
What Are Operations?	0
What Are Capabilities?	0
Agent State Operations	41
Asynchronous Events	46
Error Handling	57
Recovering Existing State	58
Working With Settings	63
Toast And Case Data	73
Disposition codes	79
Request Parameters	81
Return Values	94
Subresources	88
Filtering	108
Filtering and sorting users by fields and subresources	110
Cross Origin Resource Sharing (CORS) Settings	112
Cross Site Request Forgery Protection	114
Features	116
Services	119
<b>Voice API</b>	<b>123</b>
Calls	123
Dial	125
Answer	130
Reject	132

Hold	134
Retrieve	136
Hangup	138
SendDTMF	140
MuteCall	142
UnmuteCall	143
SetCallDisposition	144
Call Forwarding	123
ForwardCallsOn	147
ForwardCallsOff	149
Conferences and Transfers	151
SingleStepConference	151
InitiateConference	156
CompleteConference	160
SingleStepTransfer	164
InitiateTransfer	167
CompleteTransfer	178
RemoveParticipantFromConference	188
SwapCalls	190
MergeWithOtherCall	203
Call Data	124
AttachUserData	204
UpdateUserData	207
DeleteUserData	211
DeleteUserDataPair	213
Agent State	124
Ready	216
Not Ready	218
Aux Work	219
After Call Work	221
Offline	223
DoNotDisturbOn	225

DoNotDisturbOff	227
Supervisor	124
ListenIn	229
Coach	232
BargeIn	234
CancelSupervisionMonitoring	237
SwitchToBargeIn	239
SwitchToListenin	242
MuteMonitoredUser	244
UnmuteMonitoredUser	247

Session Management	125
StartContactCenterSession	261
EndContactCenterSession	263
Mobile Push Notifications	264

<b>Hierarchical Dispositions API</b>	<b>599</b>
Retrieve all configured dispositions in a specific disposition group	601
Retrieve full contents of a specific disposition category	602
Retrieve full disposition and category tree	603
Create a new disposition	604
Remove a disposition	605
Create sub-category of dispositions	605
Create top-level disposition group	606
Remove sub-category of dispositions	607
Remove root-level disposition group	607
Configuration Manager Scenarios	608

# Web Services API Reference

## RESTRICTED

### Important

- This is **restricted release** documentation, and therefore is subject to change and is not complete. Some features that are described in this section might not be fully implemented in the application.

## Overview

Welcome to the Genesys Web Services Application Programming Interface (API) Reference.

This Genesys Web Services REST API can be used to write agent applications that provide a variety of contact center-related features. These features include agent state management, call control, supervisor monitoring.

This reference explains the resources and methods available to developers.

Each category presents information about relevant operations, related resources, request parameters, and return values.

**Samples (*restricted*)** Simple examples showing how to create applications to access the Genesys Web Services REST API.

**General (*restricted*)** Concepts employed by the various portions of the Genesys Web Services REST API.

**Voice API (*restricted*)** API for developers building voice-related client applications.

**Hierarchical Dispositions API (*restricted*)** API for developers building dispositions-related client applications.

# Samples

This is the Getting Started section of the [Genesys Web Services REST API](#).

## RESTRICTED

### Important

This is **restricted release** documentation, and therefore is subject to change and is not complete. Some features that are described in this section might not be fully implemented in the application.

## Introduction

The [Genesys Web Services REST API](#) provides developers with the capability to build applications that can interact with the Genesys environment through a simple REST API, rather than requiring the use of a complicated software framework.

REST stands for Representational State Transfer, and is stateless, and Genesys uses a Hypertext Transfer Protocol (HTTP) REST API.

REST uses a client-server model, with cacheable communications protocol.

REST uses **resources** and **actions** instead of **methods** and **data types**. Resources are URLs that reference the object where you want to perform an action.

## Requests

These HTTP requests are used for all four standard CRUD (Create/Read/Update/Delete) operations by way of the following HTTP Requests:

Request	Description
POST	Create a new resource.
GET	Read a specific resource or collection of resources by sending an identifier.
PUT	Update a specific resource or collection of resources by sending an identifier.



**DELETE** Remove/delete a specific resource by sending an identifier.

## Responses

Server responses are formatted as JavaScript Object Notation (JSON).

## What is included in the Genesys Web Services REST API?

Currently, the following Genesys components are implemented (partially or completely) in the **Genesys Web Services REST API**:

- **Voice Interactions**

- **Hierarchical Dispositions**

More components and additional functionality are being added all the time.

---

## Getting Started

Any developer with a working knowledge of REST-based application development and who has successfully deployed the **REST API** that comes with Genesys Web Services can use this section to get started and explore the API.

## Testing the API

Several tools are available for simple browser-based testing. Our recommendations include one of the following Google Chrome extensions:

- [Advanced Rest Client](#) or
- [REST Console](#).

---

## Get Version

When you are using the **REST API** that is part of Genesys Web Services, you may want to determine which version of the Genesys Web Services software is deployed in your installation. This is about the simplest thing you can do with this API and is the *only* thing you can do without being authenticated.

In this sample, we will show you the three different methods you can use for getting that information:

- [Javascript](#)
- [REST Client](#)
- [Web Browser](#)

### Javascript

This sample uses JavaScript to find the version of the installed Genesys Web Services software. (For simplicity the server location is hard coded.)

```
<!doctype html>
<html>
  <head>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/
1.11.1/jquery.min.js"></script>
    <script>
      $(document).ready(function() {

        // Add a click handler to the
        getVersion button.

        $('#getVersion')
          .click(function() {

            // Create and configure the
            request.

            var request = {
              url: 'http://localhost:8080/
api/v2/diagnostics/version',
              type: 'GET', crossDomain:
```

```
true, success: function (result) {
    // Update the label
    with the result.
    $('#version').text(result.version);
},
error: function (result) {
    alert('Failed to get
version.');
```

```
});
$.ajax(request);
});
});
</script>
</head>
<body>
  <div>
    <button id='getVersion'>Get Version</button>
    <br/>
    <label id='version'>--</label>
  </div>
</body>
</html>
```

## Response

```
{
  "statusCode":0,
  "version":"8.5.200.23"
}
```

## REST Client

Instead of writing a Javascript or other client to test your API calls, you can use a **REST client** embedded in your web browser to test REST API calls.

## Request

In this example, we retrieve the software version from the REST API using a REST client:

---

### Target

<b>Target</b> <b>Request URI</b> <input type="text" value="http://localhost/api/v2/diagnostics/version"/> <small>Universal Resource Identifier. ex: https://www.sample.com:9000</small>	<b>Accept</b> <b>Content-Type</b> <input type="text" value="example: text/plain"/> <small>Content-Types that are acceptable.</small>
<b>Request Method</b> <input type="button" value="GET"/> <small>The desired action to be performed on the identified resource.</small>	<b>Language</b> <input type="text" value="example: en-US"/> <small>Acceptable languages for response.</small>
<b>Request Timeout</b> <input type="text" value="60"/> seconds <small>Timeout in seconds before aborting.</small>	

## Response

The Response should look similar to this response:

### Response

**Response Body** | RAW Body | Response Headers | Response Preview | Request Body | Request Headers

Color Theme:  | Force Syntax Highlighting:  Auto  JSON  XML  HTML  CSS

```
{
  "statusCode": 0,
  "version": "8.5.200.23"
}
```

## Web Browser

You can also use a web browser to perform a simple test and find out which version of the product we are interacting with.

This call is the only REST API call that can be made in a web browser, because it does not require authentication. This means that using a web browser to test most API calls is not possible, but we can use it to confirm the API is working and responding to calls.

---

## Request

In a web browser, enter and navigate to the following (where `<server name>` is the server where the **Genesys Web Services** is installed, for example: `test.com`):

```
http://<server name>/api/v2/diagnostics/version
```

### Important

If you are not using the standard port 80 for your web traffic, append `<server name>` with the port. For example: `test.com:8080`

## Response

If successful, the response (result) should look like this:

```
{"statusCode":0,"version":"8.5.200.23"}
```

---

# Get User Information

This is part of the **Samples** section of the **Genesys Web Services REST API**.

## Audience

Developers building applications to utilize the **Genesys Web Services REST API**.

## Prerequisites

You have successfully deployed the **Genesys Web Services REST API**.

## GET User Information

The following is an example showing how to implement the REST API calls.

```
<!doctype html>
<html>
  <head>
```

---

```

        <script src='//ajax.googleapis.com/ajax/libs/jquery/
1.11.1/jquery.min.js'></script>
        <script>
            $(document).ready(function() {

                $('#getMe')
                .click(function() {

                    // Read the values from the
input boxes
                    var username =
$('#username').val();
                    var pw =
$('#password').val();
                    var uri =
$('#baseUri').val();
                    uri += 'api/v2/me';

                    // Create and configure the
request
                    var request = {
                        url: uri,
                        type: 'GET',
                        crossDomain: true,
                        success: function (result) {
                            // Update the textarea with
a string version of the resulting JSON
                            $('#result').text(JSON.stringify(result
null, 4));
                        },
                        error: function (result) {
                            alert('Failed to get my user
info.');
```

```
};
```

```

                // This adds the Authorization
header. The call to btoa base 64 encodes the username and
                // password separated by a ':'. For
more info on Basic authentication check the RFC.
                request.beforeSend = function (xhr) {
```

```

            }
        }
    }
};
```

---

---

```

                                xhr.setRequestHeader('Authorizati
'Basic ' + window.btoa(username + ':' + pw));
                                };

                                $.ajax(request);
                                });
                                </script>
</head>
<body>
    <div>
        <input id='baseUri' type='text' style='margin-bottom:
5px; width: 200px;' placeholder='GWS Base Uri'
value='http://localhost:8080/'>
        <br/>
        <input id='username' type='text' style='margin-bottom:
5px; width: 200px;' placeholder='Username'
value='pavel@redwings.com'>
        <br/>
        <input id='password' type='password'
placeholder='Password' value='password'>
        <br/>
        <button id='getMe'>Get Me</button>
        <br/>
        <br/>
        <textarea id='result' rows='20'
cols='100'></textarea>
    </div>
</body>
</html>

```

## Using a REST Console

Using a REST API extension, you can test the API call to GET user information.

## Authentication

To retrieve agent information, we must authenticate, as shown here (in our case, `cspencer` is the user and there is no password associated with her account):

### Basic Authorization ✕

**Username**

**Password**

Set Header
Reset

Choose `Set Header`, and you should see something similar to the following:

### ☑ Authorization

**Authorization Header:**

example: Basic QWxhZGRpbjpvYy5GVulHNlc2FiZQ==
 
Basic Auth
Setup OAuth
Refresh OAuth

Authentication credentials for HTTP authentication.

## Request

Set up the API call as shown in this example:

### ☑ Target

<p><b>Target</b></p> <p><b>Request URI</b></p> <input style="width: 100%;" type="text" value="http://localhost/api/v2/me"/> <p style="font-size: x-small; margin-top: 5px;">Universal Resource Identifier. ex: https://www.sample.com:9000</p> <p><b>Request Method</b></p> <div style="border: 1px solid #ccc; padding: 2px; background-color: #fff9c4; display: inline-block;">GET</div> <p style="font-size: x-small; margin-top: 5px;">The desired action to be performed on the identified resource.</p> <p><b>Request Timeout</b></p> <input style="width: 50px;" type="text" value="60"/> seconds <p style="font-size: x-small; margin-top: 5px;">Timeout in seconds before aborting.</p>	<p><b>Accept</b></p> <p><b>Content-Type</b></p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;"> <input type="checkbox"/> example: text/plain         </div> <p style="font-size: x-small; margin-top: 5px;">Content-Types that are acceptable.</p> <p><b>Language</b></p> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;"> <input type="checkbox"/> example: en-US         </div> <p style="font-size: x-small; margin-top: 5px;">Acceptable languages for response.</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

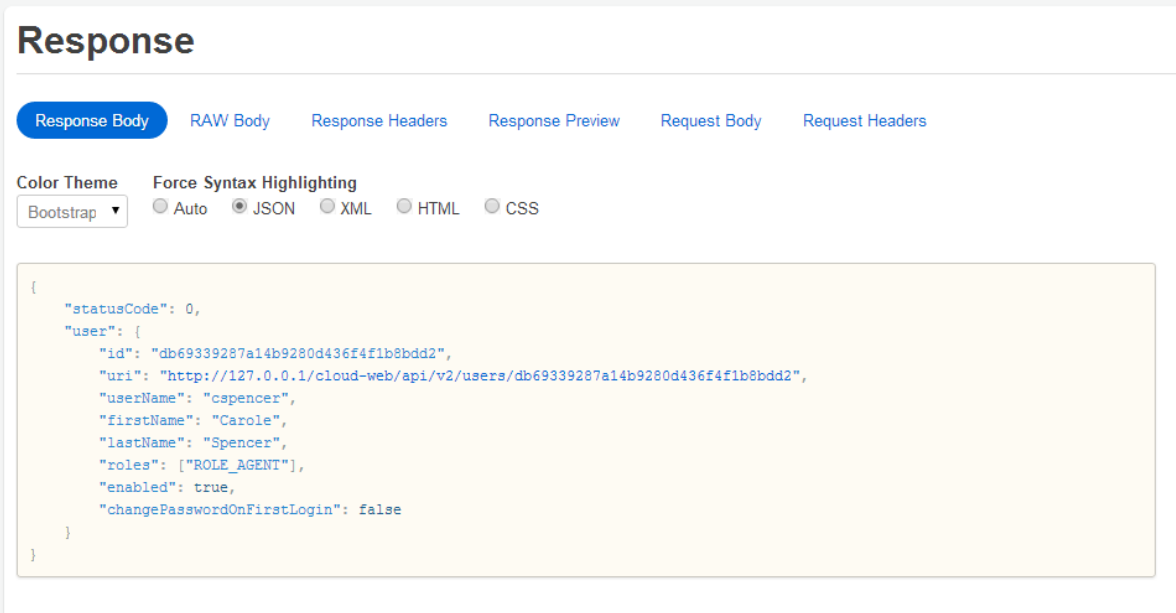


## Important

- If you are not using the standard port 80 for your web traffic, append `<server name>` with the port. For example: `test.com:8080`

## Response

If successful, the response should look similar to this:



The screenshot shows a web interface titled "Response" with several tabs: "Response Body" (selected), "RAW Body", "Response Headers", "Response Preview", "Request Body", and "Request Headers". Below the tabs, there are settings for "Color Theme" (set to "Bootstrap") and "Force Syntax Highlighting" (with radio buttons for "Auto", "JSON" (selected), "XML", "HTML", and "CSS"). The main content area displays a JSON response with syntax highlighting:

```
{
  "statusCode": 0,
  "user": {
    "id": "db69339287a14b9280d436f4f1b8bdd2",
    "uri": "http://127.0.0.1/cloud-web/api/v2/users/db69339287a14b9280d436f4f1b8bdd2",
    "userName": "cspencer",
    "firstName": "Carole",
    "lastName": "Spencer",
    "roles": ["ROLE_AGENT"],
    "enabled": true,
    "changePasswordOnFirstLogin": false
  }
}
```

# Cross Site Request Forgery Protection Example

## Overview

Genesys Web Services provides protection against Cross Site Request Forgery (CSRF) attacks.

Setting up CSRF is covered in the Deployment Guide.

In these samples, we will show you two different methods you can use for getting that information:

- [Javascript](#)
- [Python](#)

## Notes

When CSRF is enabled:

- The token is retrieved prior to starting Cometd.
- When tokens appear in the response header for a `GET` request they are saved.
- When token values are saved, they are included in requests. This sample sends headers for all, including `GET`.
- When `EndContactCenterSession` is called, token values are cleared.

## JavaScript Example

This sample uses JavaScript to .

```
<html>
  <head>

    <script type="text/javascript" src="./org/
cometd.js"></script>
    <script type="text/javascript" src="./org/cometd/
ReloadExtension.js"></script>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/
1.11.1/jquery.min.js"></script>
    <script src="./jquery.cometd.js"></script>

    <script>

    ////////////////////////////////////////////////////////////////////
    // Initialization
    ////////////////////////////////////////////////////////////////////
    var baseUrl = 'http://127.0.0.1:8080';
    var username = 'paveld@redwings.com';
    var password = 'password';

    var csrfHeaderName;
    var csrfToken;
    var cometd;

    $.ajaxSetup({
beforeSend: function(xhr) {
```

---

```

        if (csrfHeaderName && csrfToken) {
            xhr.setRequestHeader(csrfHeaderName,
csrfToken);
        }
    });

$(document).ready(function() {
    $('#getMeButton').click(getMe);
    $('#startCometdButton').click(connectCometD);
    $('#startSessionButton').click(startContactCenterSession);
    $('#readyButton').click(ready);
    $('#stopCometdButton').click(disconnectCometD);
    $('#endSessionButton').click(endContactCenterSession);

    cometd = $.cometd;

    cometd.addListener('/meta/handshake',
onHandshake);
    cometd.addListener('/meta/connect', onConnect);
    cometd.addListener('/meta/disconnect',
onDisconnect);

    $(window).unload(function() {
        cometd.disconnect();
    });
});

////////////////////////////////////
// HTTP Helpers
////////////////////////////////////
var get = function(params)
{
    var request = {
        url: baseUri + params.uri,
        type: 'GET',
        crossDomain: true,
        xhrFields: {
            withCredentials: true
        },
        success: function (data, textStatus, response) {
            console.log(response.getAllResponseHeaders());
        }
    };
};

```

---

```
                if
(response.getResponseHeader('X-CSRF-HEADER') &&
response.getResponseHeader('X-CSRF-TOKEN')) {
                    csrfHeaderName =
response.getResponseHeader('X-CSRF-HEADER');
                    csrfToken =
response.getResponseHeader('X-CSRF-TOKEN');

                                console.log('csrfHeaderName: ' +
csrfHeaderName);
                                console.log('csrfToken: ' +
csrfToken);
                }

                if (params.callback) {
                    params.callback(data);
                }
            },
            error: function (result) {
                console.log(result);

                if (params.error) {
                    params.error(result);
                }
            }
        };

        if (params.includeCredentials) {
            request.beforeSend = function (xhr) {
                xhr.setRequestHeader('Authorization',
'Basic ' + window.btoa(username + ':' + password));
            };
        }

        $.ajax(request);
    };

    var post = function(params)
    {
        var data = JSON.stringify(params.json,
undefined, 2);
```

---

```
var request = {
  url: baseUri + params.uri,
  type: 'POST',
  data: data,
  headers: {
    'Content-Type' : 'application/json'
  },
  crossDomain: true,
  xhrFields: {
    withCredentials: true
  },
  handleAs: 'json',
  success: function(data) {
    if (params.callback) {
      params.callback(data);
    }
  },
  error: function (req, err, exception) {
    console.log('Error! (' + req.status + ') : ' + err + ' ' + exception);
    if (params.error) {
      params.error(result);
    }
  }
};

$.ajax(request);
}

////////////////////////////////////
// API Functions
////////////////////////////////////
var getMe = function() {
  get({
    uri: '/api/v2/me',
    includeCredentials: true
  });
};

var startContactCenterSession = function() {
```

---

---

```
        post({
            uri: '/api/v2/me',
            json: {
                operationName:
'StartContactCenterSession',
                channels: ['voice']
            }
        });
};

var ready = function() {
    post({
        uri: '/api/v2/me',
        json: {
            operationName: 'Ready'
        }
    });
};

var endContactCenterSession = function() {
    post({
        uri: '/api/v2/me',
        json: {
            operationName:
'SendContactCenterSession'
        },
        callback:
onEndContactCenterSessionComplete
    });
};

////////////////////////////////////
// Callbacks
////////////////////////////////////
var onEndContactCenterSessionComplete = function() {
    csrfHeaderName = null;
    csrfToken = null;
}

////////////////////////////////////
// CometD
////////////////////////////////////
```

---

```
var connected = false;
var subscription;

var onConnect = function(message) {
    if (cometd.isDisconnected()) {
        return;
    }

    var wasConnected = connected;
    connected = message.successful;
    if (!wasConnected && connected) {
        console.log('Cometd connected.');
```

disconnected...');

```
    } else if (wasConnected && !connected) {
        console.log('Cometd
};

var onDisconnect = function(message) {
    if (message.successful) {
        connected = false;
        console.log('Cometd disconnected.');
```

JSON.stringify(message, null, 2));

```
    }

var onMessage = function(message) {
    console.log('Cmetd message received:\n' +
};

var onHandshake = function(handshake) {
    if (handshake.successful === true) {
        if (subscription) {
            console.log('unsubscribing:
' + subscription);
            cometd.unsubscribe(subscription);
        }

        console.log('Subscribing to
channels...');
        subscription =
```

---

---

```
cometd.subscribe('/v2/me/*', onMessage);
    }
    };

    var connectCometD = function() {

        var reqHeaders = {};
        reqHeaders[csrfHeaderName] = csrfToken;

        cometd.unregisterTransport('websocket');
        cometd.unregisterTransport('callback-polling');

        cometd.configure({
            url: baseUrl + '/api/v2/
notifications',
            logLevel: "info",
            requestHeaders: reqHeaders
        });

        cometd.handshake();
    };

    var disconnectCometD = function() {
        cometd.disconnect();
    };

</script>
</head>

<body>
    <button id='getMeButton'>Get Me</button>
    <br/>
    <button id='startCometdButton'>Start CometD</button>
    </br>
    <button id='startSessionButton'>Start Contact
Center Session</button>
    <br/>
    <button id='readyButton'>Ready</button>
    <br/>
    <button id='stopCometdButton'>Stop CometD</button>
    </br>
    <button id='endSessionButton'>End Contact Center
```

---



```
Session</button>
    </body>
</html>
```

## Python Example

Working Python client sample.

```
import base64;
import httplib2;
import json;

GWS_BASE_URI = "http://127.0.0.1:8080/api/v2"
ADMIN_USERNAME = "mikeb@redwings.com"
ADMIN_PASSWORD = "password"

CONTACT_CENTER_USERS = [
    {
        "userName": "bobp@redwings.com",
        "firstName": "Bob",
        "lastName": "Probert",
        "password": "password",
        "phoneNumber": "5019",
        "role": "ROLE_AGENT"
    }
]

X_CSRF_HEADER = "x-csrf-header"
X_CSRF_TOKEN = "x-csrf-token"

jsessionId = None
csrfHeaderName = None
csrfTokenValue = None

http = httplib2.Http(".cache")

def create_request_headers():
    request_headers = dict()
    request_headers["Content-Type"] = "application/json"
    request_headers["Authorization"] = "Basic " +
base64.b64encode(ADMIN_USERNAME + ":" + ADMIN_PASSWORD)

    if jsessionId:
```

---

```
    request_headers["Cookie"] = jsessionid;
    print "Using JSESSIONID %s" % jsessionid;

    if csrfHeaderName and csrfTokenValue:
        print "Adding csrf header [%s] with value [%s]..." %
(csrfHeaderName, csrfTokenValue)
        print
        request_headers[csrfHeaderName] = csrfTokenValue
    else:
        print "No csrf token, skipping..."
        print

    return request_headers

def post(uri, content):
    request_headers = create_request_headers()
    body = json.dumps(content, sort_keys=True, indent=4)

    print "POST %s (%s/%s)..." % (uri, ADMIN_USERNAME,
ADMIN_PASSWORD)
    print body
    print

    response_headers, response_content = http.request(uri, "POST",
body = body, headers = request_headers)
    status = response_headers["status"]

    ugly_response = json.loads(response_content)
    pretty_response = json.dumps(ugly_response, sort_keys=True,
indent=4)

    print "Response: %s" % (status)
    print "%s" % (pretty_response)
    print

    return response_headers, ugly_response

def get(uri):

    global csrfHeaderName
    global csrfTokenValue
    global jsessionid
```

---

```
    request_headers = create_request_headers()
    print "GET %s (%s/%s)..." % (uri, ADMIN_USERNAME,
ADMIN_PASSWORD)
    print

    response_headers, response_content = http.request(uri, "GET",
headers = request_headers)
    status = response_headers["status"]
    if response_headers["set-cookie"]:
        jsessionid = response_headers["set-cookie"]
        print "Set JSESSIONID %s..." % jsessionid

    ugly_response = json.loads(response_content)
    pretty_response = json.dumps(ugly_response, sort_keys=True,
indent=4)

    print "Response: %s" % (status)
    print "%s" % (pretty_response)
    print

    if X_CSRF_HEADER in response_headers:
        csrfHeaderName = response_headers[X_CSRF_HEADER]
        print "Saved csrf header name [%s]" % csrfHeaderName

    if X_CSRF_TOKEN in response_headers:
        csrfTokenValue = response_headers[X_CSRF_TOKEN]
        print "Saved csrf token value [%s]" % csrfTokenValue
        print

    return response_headers, ugly_response

def check_response(response_headers, expected_code):
    if response_headers["status"] != expected_code:
        print "Request failed."
        exit(-1)

def create_user(user_info):
    user_name = user_info["userName"]
    print "Creating user [%s]..." % (user_name)

    uri = "%s/users" % (GWS_BASE_URI)
```

---

```
user = {
    "userName": user_name,
    "password": user_info["password"],
    "firstName": user_info["firstName"],
    "lastName": user_info["lastName"],
    "roles": [user_info["role"]]
}

response_headers, response_content = post(uri, user)
check_response(response_headers, "200")

user_id = response_content["id"]
print "User [%s] created. User id [%s]." % (user_name, user_id)

return user_id

def assign_device_to_user(user_id, phone_number):
    print "Creating device [%s] and assigning to user [%s]..." %
(phone_number, user_id)

    uri = "%s/users/%s/devices" % (GWS_BASE_URI, user_id)

    device = {
        "phoneNumber": phone_number
    }

    response_headers, response_content = post(uri, device)
    check_response(response_headers, "200")

    device_id = response_content["id"]
    print "Device [%s] created and assigned to user id [%s]." %
(device_id, user_id)

def create_users_and_devices():
    for user_info in CONTACT_CENTER_USERS:
        user_id = create_user(user_info)
        assign_device_to_user(user_id, user_info["phoneNumber"])

def getToken():
    uri = "%s/diagnostics/version" % (GWS_BASE_URI)
```

---

```
    response_headers, response_content = get(uri)
    check_response(response_headers, "200")

if __name__ == "__main__":
    getToken()
    create_users_and_devices()
```

# General

## RESTRICTED

### Important

- This is **restricted release** documentation, and therefore is subject to change and is not complete. Some features that are described in this section might not be fully implemented in the application.

This document describes the general concepts involved in the **Genesys Web Services REST API** and provides guidance for developers building related applications.

### General

---

- What Are Operations?
- What Are Capabilities?
- Agent State Operations
- Asynchronous Events
- Error Handling
- Recovering Existing State
- Working With Settings

- 
- Toast And Case Data
  - Disposition codes
  - RequestTypes
  - ReturnValues
  - Subresources
  - Filtering
  - Filtering and sorting

- 
- Cross Domain (CORS) Filter
  - Cross Site Request Forgery Protection
  - Features
  - Services

# What Are Operations?

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

## Overview

The typical core requests in a REST API include:

- `create`
- `read`
- `update`
- `delete`

There are several areas of the Genesys Web Services REST API where the design departs from the core requests on resources.

In areas such as the [Voice API](#), where there are many differing requests for a particular resource that must be processed asynchronously, the notion of an "operation" is introduced. Resources that support this concept allow an operation to be requested by name by sending a `POST` with the name of the operation to be executed along with any supporting parameters that are required.

The majority of the traditional voice functionality, such as agent state manipulation and call control, is implemented in this manner.

Both device and call resources support operations that fall under the [Voice API](#).

## Operations that can be performed on the call resource

- [Dial](#)
- [Answer](#)
- [Hangup](#)
- [SingleStepConference](#)

## Operations for the device resource

- [Ready](#)

- [NotReady](#)
- [AfterCallWork](#)
- [DoNotDisturbOn](#)

## Example

The operation name is included as a property in the POST body:

```
POST api/v2/me/calls/0071023821aec011
{
  "operationName": "Answer"
}
```

---

# What Are Capabilities?

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

## Overview

Capabilities is a complimentary concept that Genesys Web Services employs to assist client developers in deciding when it is appropriate to allow various operations to be performed.

In the [Voice API](#), both the device resource and the call resource include a "capabilities" property. This property provides a list of operation names that are valid given the current state of the device or call.

When processing call or device resources either via state recovery, or processing CometD events the client can enable or disable user interface elements based on the contents of the capabilities array.

## Examples

### Device Capabilities Example

DeviceStateChangeMessage:

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
```



```
"devices": [
  {
    "id": "9c14cad7-17c4-48d0-8492-7cf0ff92c224",
    "deviceState": "Active",
    "userState": {
      "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
      "displayName": "Not Ready",
      "state": "NotReady"
    },
    "phoneNumber": "5001",
    "e164Number": "5001",
    "telephonyNetwork": "Private",
    "doNotDisturb": "Off",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn",
      "DoNotDisturbOn"
    ]
  }
],
"channel": "/v2/me/devices"
}
```

The capabilities property on the device above includes two operation names:

- ForwardCallsOn, and
- DoNotDisturbOn

Based on this, the client application can enable the UI elements for invoking these operations while disabling others, for example: DoNotDisturbOff.

After sending a DoNotDisturbOn request, the following DeviceStateChangeMessage is received:

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "9c14cad7-17c4-48d0-8492-7cf0ff92c224",
```

```

    "deviceState": "Active",
    "userState": {
      "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
      "displayName": "Not Ready",
      "state": "NotReady"
    },
    "phoneNumber": "5001",
    "e164Number": "5001",
    "telephonyNetwork": "Private",
    "doNotDisturb": "On",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/
voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn",
      "DoNotDisturbOff"
    ]
  }
}
],
},
"channel": "/v2/me/devices"
}

```

### Important

In this example, the `DoNotDisturbOn` operation is no longer present in the capabilities array and has been replaced by `DoNotDisturbOff`. Also note the inclusion of the `doNotDisturb` property.

Agent state operations are not governed by capabilities due to the ability to define custom operations. Any agent state operation can be sent at any time. Genesys Web Services and Genesys TServer are generally tolerant of duplicate requests. For example, if you send a request for `NotReady` and the current user is already in the `NotReady` state, the request will be ignored.

## Call Capabilities Example

Like devices, the call resource provides a capabilities property as shown in the `CallStateChangeMessage`:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e00a",
      "state": "Ringing",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000B",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/74152ed8-858f-4a33-9e96-36213a678d30",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e00a",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5000",
      "callType": "Internal",
      "capabilities": [
        "AttachUserData",
        "Answer",
        "UpdateUserData",
        "DeleteUserData",
        "DeleteUserDataPair"
      ],
      "duration": "0",
      "mute": "Off",
      "supervisorListeningIn": false,
      "monitoredUserMuted": false
    },
    "phoneNumber": "5000"
  },
  "channel": "/v2/me/calls"
}
```

---

Since this call is 'ringing', a limited set of capabilities are provided, including the `Answer` operation.

Once a call is answered, another notification is sent with updated state and list of capabilities:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e00a",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000B",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/74152ed8-858f-4a33-9e96-36213a678d30",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e00a",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5000",
      "callType": "Internal",
      "capabilities": [
        "AttachUserData",
        "InitiateConference",
        "UpdateUserData",
        "Hold",
        "SingleStepTransfer",
        "DeleteUserData",
        "SingleStepConference",
        "Hangup",
        "DeleteUserDataPair",
        "SendDtmf",
        "InitiateTransfer"
      ]
    }
  }
}
```

```
    ],
    "duration": "5",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5000"
},
"channel": "/v2/me/calls"
}
```

### Important

- The set of available capabilities is greatly expanded once a call has been established.

For additional capabilities examples, see the individual operation pages for operation specific samples and related scenarios.

---

## Agent State Operations

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

An agent state operation is a resource that defines the state, workmode, and reason code that should be used when sending a state change to Genesys TServer. Genesys Web Services defines a basic set of system agent state operations (see below) and additional agent state operations can be defined.

Once a custom agent state operation has been defined, it will be returned to the client application when the set of available agent state operations are queried (**GET `api/v2/me/settings/agent-states`**).

To request that the current user change state using the parameters defined in a specific

agent state operation, the client application can send a **POST** to `/api/v2/me/channels/voice` and include the operation name as a parameter.

On startup, client applications should query for the set of available agent state operations and allow the user to choose any of returned options. Each agent state operation also defines a display name that can be used in UI elements. A unique id is also provided to allow clients to more easily identify a state.

## System Defined Agent State Operations

The following agent state operations are system defined and are always available for use. They cannot be modified or deleted.

operationName	displayName	state	workMode
Ready	Ready	Ready	-
NotReady	NotReady	NotReady	-
AuxWork	AuxWork	NotReady	AuxWork
AfterCallWork	AfterCallWork	NotReady	AfterCallWork
Offline	Offline	Logout	-

## Creating Custom Agent State Operations

A custom agent state operation can be created by using the provisioning API:

```
POST /api/v2/settings/agent-states
{
  "operationName": "Training",
  "displayName": "Training",
  "state": "NotReady",
  "workMode": "AuxWork",
  "reason": "Training"
}
```

### Important

- Creating agent states through the API requires the Administrator role.

---

Agent state operations can also be created by adding a new Action Code using Configuration Manager or Genesys Administrator. See the Deployment and Configuration documentation for additional details.

## Retrieving Available Agent State Operations

The set of available agent state operations for the current user can be read as shown in this example.

### Request

```
GET api/v2/settings/agent-states
```

### Response

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

```

---

```

    "displayName": "Offline",
    "state": "Logout"
  },
  {
    "id": "9430250E-0A1B-421F-B372-F29E69366DED",
    "operationName": "Ready",
    "displayName": "Ready",
    "state": "Ready"
  },
  {
    "id": "90b2b0c6-8395-4d7b-b94a-ed634ac612b3",
    "operationName": "Training",
    "displayName": "Training",
    "state": "NotReady",
    "workMode": "AuxWork",
    "reason": "Training"
  }
],
"key": "operationName"
}

```

## Invoking a Custom Agent State Operation

Requesting a custom agent state operation works the same way it does for system defined agent states.

Send a POST to `/api/v2/me/channels/voice` with the `operationName` of the agent state operation, as shown in this example:

```

POST api/v2/me/channels/voice
{
  "operationName": "Training"
}

```

A `DeviceStateChangeMessage` will be sent with the updated state:

```

{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "9c14cad7-17c4-48d0-8492-7cf0ff92c224",
        "deviceState": "Active",

```



```

    "userState": {
      "id": "90b2b0c6-8395-4d7b-b94a-ed634ac612b3",
      "displayName": "Training",
      "state": "NotReady",
      "workMode": "AuxWork",
      "reason": "Training.detroit"
    },
    "phoneNumber": "5001",
    "e164Number": "5001",
    "telephonyNetwork": "Private",
    "doNotDisturb": "Off",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn",
      "DoNotDisturbOn"
    ]
  }
}
},
"channel": "/v2/me/devices"
}

```

## Agent State Matching in Device State

When the state of a device resource is updated by events received from TServer, Genesys Web Services will attempt to match the `state`, `workmode`, and `reason` in the current state with an existing agent state operation. If a match is found, Genesys Web Services will include the `id` and `displayName` of the corresponding agent state operation as part of the `userState` device property for the convenience of the user interface:

```

"userState": {
  "id": "9430250E-0A1B-421F-B372-F29E69366DED",
  "displayName": "Ready",
  "state": "Ready"
}

```

**Important**

An `id` and `displayName` may not be included in all scenarios. Take appropriate steps to handle this possibility.

---

## Asynchronous Events

This is part of the **General** section of the **Genesys Web Services REST API**.

### Overview

All requests in the **Voice API** are asynchronous. When sending a request, you can expect a response with a status code like other **Genesys Web Services REST APIs**, however, a success response will indicate only that the request was processed and sent to T-Server. When the T-Server finishes processing the request and notifies Genesys Web Services of any resulting changes in state (or errors), the updated state or error details are delivered to the client application as **CometD** messages.

### CometD

Genesys Web Services uses **CometD** to deliver unsolicited notifications to clients. **CometD** is a library that allows the server to deliver messages to a web-based client with low-latency using a variety of transports. The transport used to deliver messages is negotiated between the client and server based on what the client supports running in a particular browser. Example transports include long polling and web sockets. **CometD** also provides a basic infrastructure for publishing and subscribing to messages. For more information about **CometD**, or for information on where to obtain client-side **CometD** libraries for various platforms, see the official CometD site.

#### Important

A basic understanding of **CometD** is a prerequisite to developing a voice application using the Genesys Web Services REST API.

## Topics

Once a **CometD** session has been established by the client, a subscription to one or more topics must be created. What subscriptions are created will depend on what functionality the client application intends to provide.

The following **CometD** topics are used by the **Genesys Web Services REST API**:

Topic	Description	MessageTypes
<i>/v2/me/devices</i>	<p>Messages related to device are published to this topic. Examples include</p> <p>changes to agent state, do-not-disturb, call forwarding, and supervisor monitoring.</p>	<ul style="list-style-type: none"> <li>• DeviceStateChangeMessage</li> <li>• ErrorMessage</li> </ul>
<i>/v2/me/calls</i>	<p>Messages related to calls are published to this topic. Examples include</p> <p>changes to call state, updates to call participant information, and updates to call data.</p>	<ul style="list-style-type: none"> <li>• CallStateChangeMessage</li> <li>• ErrorMessage</li> </ul>
<i>/notifications/services</i>	<p>Messages relating to the state of different services are published to this topic. If the connection to T-Server is lost, or T-Server's connection to the CTI link is broken, a message is delivered to the client.</p>	<ul style="list-style-type: none"> <li>• ServiceStateChangeMessage</li> </ul>

## Messages

### [+] DeviceStateChangeMessage

#### DeviceStateChangeMessage

Property	Description
data	The data element is present in all CometD notifications and is the root JSON element. The <code>data.messageType</code> property can always be used to identify the message and determine what other properties should be present.
data.messageType	This property identifies the message type and will have a value of <code>DeviceStateChangeMessage</code> .
data.devices	An array of <b>device resources</b> .
channel	The topic to which the message was published.

#### Example

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "efe1ab32-53f9-43ce-b65e-5768c61f7d4a",
        "deviceState": "Active",
        "userState": {
          "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
```

```

        "displayName": "Not Ready",
        "state": "NotReady"
    },
    "phoneNumber": "5005",
    "e164Number": "5005",
    "telephonyNetwork": "Private",
    "doNotDisturb": "Off",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/
voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
        "DoNotDisturbOn",
        "ForwardCallsOn",
        "ListenIn",
        "Coach",
        "BargeIn"
    ]
}
]
},
"channel": "/v2/me/devices"
}

```

## [+] CallStateChangeMessage

### CallStateChangeMessage

Property	Description
data	The data element is present in all <b>CometD</b> notifications and is the root JSON element. The <code>data.messageType</code> property can always be used to identify the message and determine what other properties should be present.
data.messageType	This property identifies the message type and will have a value of <code>CallStateChangeMessage</code> .
data.notificationType	This property further identifies the type of notification. See the table below for more details.
data.call	A <b>call resource</b> with the updated state and capabilities.
channel	The topic the message was published to.

**notificationType** The `notificationType` property can have one of the following values:

Type	Description
StatusChange	The status of the call has changed.
ParticipantsUpdated	The call participants have changed.
AttachedDataChanged	The call data has changed.
DtmfSent	This is sent as confirmation that the <code>SendDtmf</code> operation was successful.

**MonitoredUserMutedStateChange** The monitored user muted state has changed.

### Example

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec021",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES000015",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec021",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5000",
      "callType": "Internal",
      "capabilities": [
```

```

    "UpdateUserData",
    "SingleStepConference",
    "DeleteUserData",
    "Hangup",
    "SendDtmf",
    "Hold",
    "AttachUserData",
    "SingleStepTransfer",
    "InitiateConference",
    "DeleteUserDataPair",
    "InitiateTransfer"
  ],
  "duration": "1",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

## [+] ErrorMessage

### ErrorMessage

Property	Description
data	The data element is present in all <b>CometD</b> notifications and is the root JSON element. The <code>data.messageType</code> property can always be used to identify the message and determine what other properties should be present.
data.messageType	This property identifies the message type and will have a value of <code>ErrorMessage</code> .
data.deviceUri	The URI of the device the error message is related to.
data.callUri	The URI of the call the error message is related to. May not be present if the error is not related to a call.

---

<code>data.errorMessage</code>	The description of the error. If no error description is provided by T-Server, this may not be present.
<code>channel</code>	The topic to which the message was published.

## Examples

```
{
  "data": {
    "messageType": "ErrorMessage",
    "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "errorMessage": "Invalid Called Dn"
  },
  "channel": "/v2/me/devices"
}
```

```
{
  "data": {
    "messageType": "ErrorMessage",
    "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "callUri": "http://127.0.0.1:8080/api/v2/calls/
0071023821aec021",
    "errorMessage": "Incorrect object state"
  },
  "channel": "/v2/me/calls"
}
```

## [+] ServiceStateChangeMessage

### ServiceStateChangeMessage

Property	Description
<code>data</code>	The data element is present in all <b>CometD</b> notifications and is the root JSON element. The <code>data.messageType</code> property can always be used to identify the message and determine what other properties should be present.

---



<code>data.messageType</code>	This property identifies the message type and will have a value of <code>ServiceStateChangeMessage</code> .
<code>data.service</code>	A JSON object that describes the service the notification relates to and it's current state.
<code>data.service.type</code>	The type of service: The <b>Voice API</b> requires only the <code>Voice</code> and <code>Provisioning</code> services.
<code>data.service.state</code>	The state of the service: <code>ReadOnly</code> , <code>Inactive</code> , <code>Active</code> , or <code>Unknown</code>
<code>channel</code>	The topic to which the message was published.

The following example message is received when the connection to T-Server is unavailable

```
{
  "data": {
    "messageType": "ServiceStateChangeMessage",
    "service": {
      "id": "370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
      "name": "SIPS",
      "type": "Voice",
      "state": "Inactive"
    }
  },
  "channel": "/notifications/services"
}
```

This second example message is received when the connection to T-Server is restored.

```
{
  "data": {
    "messageType": "ServiceStateChangeMessage",
    "service": {
      "id": "370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
      "name": "SIPS",
      "type": "Voice",
      "state": "Active"
    }
  },
}
```

```
"channel": "/notifications/services"  
}
```

---

## Error Handling

This is part of the **General** section of the **Genesys Web Services REST API**.

### Overview

In the **Voice API**, errors are handled in one of two ways depending on the nature of the error.

Each HTTP request may return an error code with supporting error details. This will happen if the requested operation could not be processed and sent to TServer.

If the requested operation was processed and the corresponding request(s) were successfully sent to TServer, then the HTTP request will return successfully:

```
HTTP 200
{
  "statusCode": 0
}
```

The following response is received when, for example, the TServer connection is unavailable:

```
HTTP 503
{
  "statusCode":4,
  "statusMessage": "Voice service is not available at this time"
}
```

---

If TServer is unable to process the request and returns an error event to Genesys Web Services, the error will be delivered to the client asynchronously using a [CometD message](#). Errors received from TServer will always relate to a device but may not relate to a call, depending on the specific scenario.

See the [ErrorMessage documentation](#) for additional details and examples of the CometD error messages for the [Voice API](#).

---

## Recovering Existing State

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

Often with voice interactions there is an existing state prior to the client application starting. This could be caused by a failover/recovery scenario or because the agent logged in and started using a physical device prior to opening the application.

The Genesys Web Services REST API provides several means for client applications to discover existing state. For voice applications, the key state to be examined are the state of any devices assigned for the current user and any active calls.

These details can either be queried independently or in one request as shown in the examples below. Once any existing device and call state has been recovered, the application is able to update their UI appropriately and move forward with processing of new unsolicited messages and agent initiated actions.

### Examples

#### [+] Reading existing device state

#### Reading existing device state

The current state of any devices assigned to the current user can be read by sending a **GET** request to **api/v2/me/devices?fields=\***:

```
GET api/v2/me/devices?fields=*
{
  "statusCode": 0,
  "devices": [
    {
```

---

```

    "id": "9c14cad7-17c4-48d0-8492-7cf0ff92c224",
    "deviceState": "Active",
    "userState": {
      "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
      "displayName": "Not Ready",
      "state": "NotReady"
    },
    "phoneNumber": "5001",
    "e164Number": "5001",
    "telephonyNetwork": "Private",
    "doNotDisturb": "Off",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/
voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn"
    ]
  }
]
}

```

## [+] Reading active calls

### Reading active calls

The active calls for the current user can be read by sending a **GET** request to **api/v2/me/calls?fields=\***:

```

{
  "statusCode": 0,
  "calls": [
    {
      "id": "007102385535e008",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES000008",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
9c14cad7-17c4-48d0-8492-7cf0ff92c224",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e008",
      "participants": [

```

```
    "5000"
  ],
  "participantsInfo": [
    {
      "digits": "5000",
      "e164Number": "5000",
      "formattedPhoneNumber": "5000"
    }
  ],
  "dnis": "5001",
  "callType": "Internal",
  "capabilities": [
    "AttachUserData",
    "InitiateConference",
    "UpdateUserData",
    "Hold",
    "SingleStepTransfer",
    "DeleteUserData",
    "SingleStepConference",
    "Hangup",
    "DeleteUserDataPair",
    "SendDtmf",
    "InitiateTransfer"
  ],
  "duration": "363",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
}
]
```

## [+] Reading device state and active calls together

### Reading device state and active calls together

This example shows the results of getting all information for the current user.

---

This includes both device state and active calls in one request, but also includes other details, such as settings and skills.

```
GET api/v2/me?subresources=*
{
  "statusCode": 0,
  "user": {
    "id": "8eb52b0724344f67a44389db5aa5f192",
    "userName": "jsmith@uppercape.ca",
    "firstName": "John",
    "lastName": "Smith",
    "roles": [
      "ROLE_AGENT"
    ],
    "devices": [
      {
        "id": "9c14cad7-17c4-48d0-8492-7cf0ff92c224",
        "deviceState": "Active",
        "userState": {
          "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
          "displayName": "Not Ready",
          "state": "NotReady"
        },
        "phoneNumber": "5001",
        "e164Number": "5001",
        "telephonyNetwork": "Private",
        "doNotDisturb": "Off",
        "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
        "capabilities": [
          "ForwardCallsOn"
        ]
      }
    ],
    "skills": [],
    "settings": {
      "htcc": {
        "roles": "Agent"
      },
      "provisioning_flags": {
        "modified_At": "4c3fccdb-f942-4c99-a823-6cab68852ef7"
      }
    }
  }
}
```

---

```
    },
    "calls": [
      {
        "id": "007102385535e008",
        "state": "Established",
        "callUuid": "011DJV5JI898NB2L04000VTAES000008",
        "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
9c14cad7-17c4-48d0-8492-7cf0ff92c224",
        "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e008",
        "participants": [
          "5000"
        ],
        "participantsInfo": [
          {
            "digits": "5000",
            "e164Number": "5000",
            "formattedPhoneNumber": "5000"
          }
        ],
        "dnis": "5001",
        "callType": "Internal",
        "capabilities": [
          "AttachUserData",
          "InitiateConference",
          "UpdateUserData",
          "Hold",
          "SingleStepTransfer",
          "DeleteUserData",
          "SingleStepConference",
          "Hangup",
          "DeleteUserDataPair",
          "SendDtmf",
          "InitiateTransfer"
        ],
        "duration": "5",
        "mute": "Off",
        "supervisorListeningIn": false,
        "monitoredUserMuted": false
      }
    ],
    "changePasswordOnFirstLogin": false
```

---



```
}  
}
```

---

## Working With Settings

This is part of the **General** section of the **Genesys Web Services REST API**.

### Overview

Genesys Web Services provides two types of settings that can be accessed and manipulated by API users:

- **System Settings** - System settings are those defined by Genesys Web Services that can be adjusted by clients through the API. The various system settings sections below outline what settings are available and their purpose.
- **Custom Settings Groups** - Genesys Web Services also provides a means for clients to define their own settings groups and settings. These can be used for any purpose the client chooses.

### System Settings

- **General Settings**
- 
- **Voice Settings**
- **Voice Operations Settings**

### General Settings

The general-settings group provides read-only access to several key contact center properties:

Setting	Description
countryCode	The country code of the contact center.
countryDigits	The digit used to dial the country.
countryName	The name of the country of the contact center.

General settings can be read as shown in the following example:

#### Request:

```
GET /api/v2/settings/general-settings
```

**Response:**

```
{ "settings" : { "countryCode" : "US",  
                "countryDigits" : "1",  
                "countryName" : "United States"  
            },  
  "statusCode" : 0  
}
```

## Voice Settings

The voice group includes settings relating to agents and call activity. See below for descriptions of available settings and examples of manipulating them via the API.

The key for the voice settings group is **name**.

Setting	Description
case-filter	Defines the subset of <code>userData</code> that should be provided in the case property for established calls.. See the toast and case data page for additional details.
defaultTelephonyNetworkType	Public, to be used when creating new devices if no <code>telephonyNetworkType</code> is provided in the creation request.

---

Note that this property is a default property and can only be updated - not created or deleted.

**defaultWrapupTime** Defines the default amount of time that agents are given between ending a call and the system making them ready for the next call.

**outboundCallerId** Sets the number to be used in the caller id override for outbound calls. This can be used to protect the agent identity and provide a corporate number to show up for customers.

**toast-filter** Defines the subset of `userData` that should be provided in the toast property for ringing or dialing calls.. See the toast and case data page for additional details.

## Get voice settings

### Request:

```
GET /api/v2/settings/voice
```

Response:

```
{ "key" : "name",
  "settings" : [ { "name" : "defaultTelephonyNetworkType",
                  "value" : "Private"
                } ],
  "statusCode" : 0
}
```

## Create a setting

### Request:

```
POST /api/v2/settings/voice
```

```
{
  "name": "outboundCallerId",
  "value": "18883695555"
}
```

### Response:

```
{ "statusCode" : 0 }
```

### Update a setting

#### Request:

```
PUT /api/v2/settings/voice
{
  "name": "outboundCallerId",
  "value": "18883691212"
}
```

#### Response:

```
{ "statusCode" : 0 }
```

### Delete a setting

#### Important

Default settings cannot be created or deleted, only updated.

#### Request:

```
DELETE /api/v2/settings/voice
{
  "name": "outboundCallerId"
}
```

#### Response:

```
{ "statusCode" : 0 }
```

## Voice Operations Settings

The voice-operations setting group can be used to configure a default set of user attached data to be provided with call related operations. At present, default user data is only supported for the Dial operation. The `userData` configured via the voice-operations settings

---

will be added to any `userData` provided in the API request to dial the call. If there are duplicate keys between the two, the values provided in the API request will be used.

## Examples

### Create the setting

#### Request:

```
POST /api/v2/settings/voice-operations
{
  "operationName": "Dial",
  "userData": {
    "subscriberId": "1234567890"
  }
}
```

#### Response:

```
{ "statusCode" : 0 }
```

### Reading the existing settings

#### Request:

```
GET /api/v2/settings/voice-operations
```

#### Response:

```
{ "key" : "operationName",
  "settings" : [ { "operationName" : "Dial",
                  "userData" : { "subscriberId" : "1234567890" }
                } ],
  "statusCode" : 0
}
```

### Updating the setting

#### Request:

---

```
PUT /api/v2/settings/voice-operations
{
  "operationName": "Dial",
  "userData": {
    "subscriberId": "1234567890",
    "region": "EU"
  }
}
```

**Response:**

```
{ "statusCode" : 0 }
```

**Deleting the setting****Request:**

```
DELETE /api/v2/settings/voice-operations
{
  "operationName": "Dial"
}
```

**Response:**

```
{ "statusCode" : 0 }
```

## Custom Settings Groups

In addition to system defines settings groups and settings, Genesys Web Services allows custom groups and settings to be created. These can be used for any purpose the client application chooses. Storing user preferences is a common example. The key used in the setting group can be defined when creating the group, and the structure of the setting property values themselves (beyond the key property) can have any structure as long as it is valid JSON.

The following examples show how to work with custom settings groups.

**Reading available settings groups****Request:**

---

---

GET http://127.0.0.1:8080/api/v2/settings

Response:

```
{ "settings" : [ { "displayName" : "interaction-workspace",
  "key" : "name",
  "uri" : "http://127.0.0.1:8080/api/v2/settings/
interaction-workspace"
  },
  { "displayName" : "Voice",
  "key" : "name",
  "uri" : "http://127.0.0.1:8080/api/v2/settings/voice"
  },
  { "displayName" : "Voice Operations",
  "key" : "operationName",
  "uri" : "http://127.0.0.1:8080/api/v2/settings/
voice-operations"
  },
  { "displayName" : "dispositions",
  "name" : "dispositions",
  "uri" : "http://127.0.0.1:8080/api/v2/settings/dispositions"
  },
  { "displayName" : "General Settings",
  "name" : "general-settings",
  "uri" : "http://127.0.0.1:8080/api/v2/settings/
general-settings"
  },
  { "displayName" : "Agent States",
  "name" : "agent-states",
  "uri" : "http://127.0.0.1:8080/api/v2/settings/agent-states"
  }
  ],
  "statusCode" : 0
}
```

## Creating a new settings group

### Request:

```
POST /api/v2/settings
{
  "name": "client-settings",
  "displayName": "Client Settings",
```



```
    "key": "name"
  }
```

**Response:**

```
{ "statusCode" : 0 }
```

**Creating a new setting****Request:**

```
POST /api/v2/settings/client-settings
{
  "name": "bgColor",
  "value": "blue"
}
```

**Response:**

```
{ "statusCode" : 0 }
```

**Create a setting with a complex property value****Request:**

```
POST /api/v2/settings/client-settings
{
  "name": "screenPopCoordinates",
  "value": {
    "x": "100",
    "y": "250"
  }
}
```

**Response:**

```
{ "statusCode" : 0 }
```

**Reading settings in the group****Request:**

---

```
GET /api/v2/settings/client-settings
```

Response:

```
{ "key" : "name",
  "settings" : [ { "name" : "bgColor",
                  "value" : "blue"
                },
                { "name" : "screenPopCoordinates",
                  "value" : { "x" : "100",
                              "y" : "250"
                            }
                }
            ],
  "statusCode" : 0
}
```

### Updating a setting

#### Request:

```
PUT /api/v2/settings/client-settings
{
  "name": "bgColor",
  "value": "red"
}
```

#### Response:

```
{ "statusCode" : 0 }
```

### Deleting a setting

#### Request:

```
DELETE /api/v2/settings/client-settings
{
  "name": "bgColor"
}
```

#### Response:

```
{ "statusCode" : 0 }
```

---

---

## Deleting a settings group

### Request:

```
DELETE /api/v2/settings/client-settings
```

### Response:

```
{ "statusCode" : 0 }
```

Reading settings for the current user

---

# Toast And Case Data

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

## Overview

Genesys Web Services provides two different optional filtered subsets of the data found in the `userData` property of the call resource. Toast data is intended to be used for screen-pop (or "toast") and is included for ringing or dialing calls. Case data is included for established calls and is intended to represent the business-relevant subset of data included on the call.

The set of data that is included in the toast and case properties of the call resource are configured using the `toast-filter` and `case-filter` voice settings.

In addition to the key and value properties entries in the toast and case properties include the `displayName` property configured in the filter. The overall intention is to provide API users all the information required to build a toast/popup and/or call data display without having to manage filtering and metadata on their own.

See the sections below for examples of configuring the `toast-filter` and `case-filter` settings as well as call resource examples that include the toast and case properties.

### Tip



The `userData` property of the call resource will always contain the full set of data related to the call,

regardless of whether a toast/case filter has been configured.

## Toast Data

Toast data is intended to support screen pop or "toast" UI elements that alert the agent to a new call. Any desired set of toast data can be configured using the toast-filter setting. It is common to configure a minimal subset of the business data to allow the agent to see the information required to greet the customer.

If a `toast-filter` is configured, the toast property is provided on the call resource when a call is in the Ringing or Dialing state.

## Examples

### [+] Reading voice settings

#### Request:

```
GET http://127.0.0.1:8080/api/v2/settings/voice
```

Response:

```
{ "key" : "name",
  "settings" : [ { "name" : "case-filter",
                  "value" : [ { "attachedDataKey" : "CustomerName",
                              "displayName" : "Customer"
                            },
                            { "attachedDataKey" : "POC",
                              "displayName" : "Purpose of Call"
                            }
                          ]
                },
                { "name" : "toast-filter",
                  "value" : [ { "attachedDataKey" : "CustomerName",
                              "displayName" : "Customer"
                            }
                          ]
                }
  ],
  "statusCode" : 0
}
```

## [+] Setting a toast-filter

### Request

```
POST /api/v2/settings/voice
{
  "name": "toast-filter",
  "value": [
    {
      "attachedDataKey": "CustomerName",
      "displayName": "Customer"
    }
  ]
}
```

### Response

```
{ "statusCode" : 0 }
```

## [+] Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023c62a4a02a",
      "state": "Ringing",
      "callUuid": "01A4HVK9AG9CP40B04000VTAES00001A",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/d41b41c3-33fa-4051-a650-511c6a2e2131",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023c62a4a02a",
      "participants": [
        "5005"
      ],
      "participantsInfo": [
        {

```

```
        "digits": "5005",
        "e164Number": "5005",
        "formattedPhoneNumber": "5005"
    }
],
"dnis": "5000",
"callType": "Internal",
"capabilities": [
    "DeleteUserDataPair",
    "AttachUserData",
    "Answer",
    "DeleteUserData",
    "UpdateUserData",
],
"userData": {
    "POC": "New Service",
    "CustomerName": "Chris"
},
"toast": [
    {
        "key": "CustomerName",
        "displayName": "Customer",
        "value": "Chris"
    }
],
"duration": "0",
"mute": "Off",

"supervisorListeningIn": false,
"monitoredUserMuted": false,
"monitoring": false
},
"phoneNumber": "5000"
},
"channel": "/v2/me/calls"
}
```

## Case Data

Case data is intended to provide a business-relevant set of call data that API developers can use to populate a call data user interface. Case data is only provided after the agent has

---

accepted the call and it is in the Established state. The set of `userData` to be included in the case property can be configured via the `case-filter` setting as shown in the examples below.

## Examples

### [+] Setting a case-filter

#### Request

```
POST /api/v2/settings/voice
{
  "name": "case-filter",
  "value": [
    {
      "attachedDataKey": "CustomerName",
      "displayName": "Customer"
    },
    {
      "attachedDataKey": "POC",
      "displayName": "Purpose of Call"
    }
  ]
}
```

#### Response

```
{ "statusCode" : 0 }
```

#### Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023c62a4a02a",
      "state": "Established",
      "callUuid": "01A4HVK9AG9CP40B04000VTAES00001A",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/"
    }
  }
}
```

---

```
d41b41c3-33fa-4051-a650-511c6a2e2131",
  "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023c62a4a02a",
  "participants": [
    "5005"
  ],
  "participantsInfo": [
    {
      "digits": "5005",
      "e164Number": "5005",
      "formattedPhoneNumber": "5005"
    }
  ],
  "dnis": "5000",
  "callType": "Internal",
  "capabilities": [
    "DeleteUserDataPair",
    "SingleStepTransfer",
    "AttachUserData",
    "Hold",
    "InitiateTransfer",
    "Hangup",
    "InitiateConference",
    "DeleteUserData",
    "SingleStepConference",
    "UpdateUserData",
    "SendDtmf",

  ],
  "userData": {
    "POC": "New Service",
    "CustomerName": "Chris"
  },
  "duration": "4",
  "mute": "Off",

  "supervisorListeningIn": false,
  "monitoredUserMuted": false,
  "monitoring": false,
  "case": [
    {
      "key": "CustomerName",
```



```
    "displayName": "Customer",
    "value": "Chris"
  },
  {
    "key": "POC",
    "displayName": "Purpose of Call",
    "value": "New Service"
  }
]
},
"phoneNumber": "5000"
},
"channel": "/v2/me/calls"
}
```

---

## Disposition codes

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

Genesys Web Services provides a means for clients to read and manage disposition codes. Disposition codes are a set of possible outcomes for a call that the agent can select from. These are typically business-specific, but can include things like whether the customer issue was resolved, or whether the customer purchased the service/product being offered.

The APIs provide for disposition codes are intended to allow clients to read the set of possible codes so that they can be displayed to the user. When a disposition code is selected, it can be set for a specific call.

### Examples

The examples below show how to manage the set of possible disposition codes available to the agent.

#### Create a new disposition code

##### Request:

```
POST /api/v2/settings/dispositions
{
```

---

```
    "name": "IssueResolved",
    "displayName": "Issue Resolved"
  }
```

**Response:**

```
{ "statusCode" : 0 }
```

**Update an existing disposition code****Request:**

```
PUT /api/v2/settings/dispositions
{
  "name": "IssueResolved",
  "displayName": "Resolved"
}
```

**Response:**

```
{ "statusCode" : 0 }
```

**Delete a disposition code****Request:**

```
DELETE /api/v2/settings/dispositions
{
  "name": "IssueResolved"
}
```

**Response:**

```
{ "statusCode" : 0 }
```

**Read all defined disposition codes****Request:**

```
GET /api/v2/settings/dispositions
```

---

---

**Response:**

```
{ "key" : "name",
  "settings" : [ { "displayName" : "Issue Escalated",
                  "name" : "EscalationCreated"
                },
                { "displayName" : "Issue Resolved",
                  "name" : "IssueResolved"
                },
                { "displayName" : "Update Provided",
                  "name" : "UpdateProvided"
                },
                { "displayName" : "Information Requested",
                  "name" : "InfoRequested"
                }
              ],
  "statusCode" : 0
}
```

---

## Request Parameters

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

This outlines the request parameters for the [Genesys Web Services REST API](#).

### Object Fields

#### Requesting Devices

When making `list` requests for any kind of object, Genesys Web Services returns a list of the corresponding object URIs.

#### [+] Requesting Devices - Example

##### Request:

```
GET .../api/v2/me/devices
```

**Response:**

```
{
  "statusCode" : 0,
  "uris" : [
    "http://127.0.0.1:8080/api/v2/devices/
ba0f987f-15b4-42c7-bed0-5f302259f9db"
  ]
}
```

**Requesting a list of objects with their actual devices**

In order to receive a list of objects with their actual fields, you will need to provide the `fields` request parameter.

**[+] Requesting a list of objects with their actual devices - Example****Request:**

```
GET .../api/v2/me/devices?fields=*
```

**Response:**

```
{ "devices" : [ { "capabilities" : [ "ForwardCallsOn",
  "DoNotDisturbOn"
],
  "deviceState" : "Active",
  "doNotDisturb" : "Off",
  "e164Number" : "5001",
  "id" : "ba0f987f-15b4-42c7-bed0-5f302259f9db",
  "phoneNumber" : "5001",
  "telephonyNetwork" : "Private",
  "userState" : { "displayName" : "Ready",
    "id" : "9430250E-0A1B-421F-B372-F29E69366DED",
    "state" : "Ready"
  },
  "voiceEnvironmentUri" : "http://127.0.0.1:8080/api/v2/
```

---

```
voice-environments/a481cd8e-7b6a-4466-af88-db3471ac909e"  
    } ],  
    "statusCode" : 0  
}
```

## Specify data fields when requesting an object

When requesting an object from the Web Services server, it is possible to specify which data fields you receive by providing the `fields` request parameter.

### [+] Specify data fields when requesting an object - Example

#### Request:

```
GET .../api/v2/queues/<queue_id>?fields=id,name
```

#### Response:

```
{  
    "id":<queue_id>,  
    "name":<queue_name>  
}
```

## Requesting all field of an object

To request all fields of an object, set the `fields` property to `*`.

### [+] Requesting all field of an object - Example

#### Request:

```
GET .../api/v2/queues/<queue_id>?fields=*
```

#### Response:

```
{  
    "id":<queue_id>,  
    "name":<queue_name>,  
    "description":<queue_description>,  
    ...  
}
```

## Requesting Queues

Note that when making "list" requests for any kind of object, Web Services returns a list of the corresponding object URIs.

### [+] Requesting Queues - Example

#### Request:

```
GET .../api/v2/queues
```

#### Response:

```
{
  "statusCode":0,
  "uris":[
    "http://.../api/v2/queues/<queue_1_id>",
    ...
    "http://.../api/v2/queues/<queue_N_id>"
  ]
}
```

## Request a list of objects with their actual fields

In order to receive a list of objects with their actual fields, you need to provide the `fields` request parameter and have it set either to `*`, or to a list of data fields of interest.

### [+] Request a list of objects with their actual fields - Example

#### Request:

```
GET .../api/v2/queues?fields=id,name
```

#### Response:

```
{
  "statusCode":0,
  "queues":[{"
    "id":<queue_1_id>,
    "name":<queue_1_name>
  }],
}
```

```
...
{
    "id":<queue_N_id>,
    "name":<queue_N_name>
}]
}
```

## Object Filtering

It is possible to filter objects using request parameters when doing "list" requests.

For example:

### Request:

```
GET .../api/v2/queues?fields=id,name,channel&channel=voice
```

### Response:

```
{
    "statusCode":0,
    "queues":[{
        "id":<queue_1_id>,
        "name":<queue_1_name>,
        "channel":"voice"
    },
    ...
    {
        "id":<queue_N_id>,
        "name":<queue_N_name>,
        "channel":"voice"
    }
    ]
}
```

### Important

- Note that the filtering parameter must be exactly the same as the name of the corresponding object field.

You can also combine several filtering parameters to make even more constraints:

### Request:

```
GET .../api/v2/system/
routing-templates?fields=*&channel=voice&version=1.0.0
```

**Response:**

```
{
  "statusCode":0,
  "routingTemplates":[{"
    "id":"00_RouteToSpecDestination",
    "name":"Route Call to Specified Destination",
    "description":"Routes calls to a skill or queue",
    "version":"1.0.0",
    "channel":"voice",
    "dependencies":["media", "destination"],
    "enabled":true,
    "schema": [...]
  },
  ...
  {
    "id":"07_SegmentCallerRouteToSpecDestination",
    "name":"Play Greeting, Segment Caller, and Route To
Specified Destination",
    "description":"Plays a user-configured greeting,
...",
    "version":"1.0.0",
    "channel":"voice",
    "dependencies":["media", "destination",
"data_record_type"],
    "enabled":false,
    "schema": [...]
  }
}]
}
```

**Important**

- Note that some "list" requests may make some of the filtering parameters mandatory.

**Pagination**

The following pagination-related request parameters can be used with REST API requests.



### Important

- Pagination and sorting functionality is only enabled if Elastic Search indexing is enabled.

Name	Description	Request	Resources	Example
offset	Specifies the index of the first record to be returned. <ul style="list-style-type: none"> <li>• Defaults to 0.</li> </ul>	GET	All "plural" resources	The following request will return the first 100 users in the contact center:  <pre>GET /api/v2/users?offset=0&amp;limit=100</pre>
limit	Specifies the number of records to be returned. <ul style="list-style-type: none"> <li>• Maximum allowed value is 500.</li> <li>• Parameter is <b>MANDATORY</b> for all resources which support pagination.</li> </ul>	GET	All "plural" resources	The following request will return the second page of 25 users in the contact center:  <pre>GET /api/v2/users?offset=25&amp;limit=25</pre>

Read requests with pagination return an extra field called `totalCount` containing the total count of objects satisfying the request criteria.

```
{
  "statusCode": 0,
  "users": [...],
  "totalCount": 2
}
```

The following API resources support sorting and pagination:

- **users**
- **groups/<id>/users**
- **contacts**

## Sorting

The following sorting-related request parameters can be used with REST API requests.

Name	Description	Request	Resources
sortBy	Specifies a comma separated list of object properties to be used for sorting. GET All "plural" resources The following request will sort users by their last names first and then by their first names:	GET	<code>/api/v2/users?sortBy=lastName,firstName&amp;limit=100</code>
order	Specifies sorting order to be used, can be either "Ascending" or "Descending", defaults to "Ascending".	GET	All "plural" resources  The following in a desc  GET /ap users?sc

## Subresources

The subresources feature allows you to read subresources of an object together with the object itself. If you have a user object that has one or more skills and one or more devices, you can read all skills and devices of that user with the following request:

### Request:

```
GET .../api/v2/users/<user_id>?subresources=*
```

**Response:**

```
{
  "id":<user_id>,
  "firstName":<first_name>,
  ...
  "skills":[
    {
      "id":<skill_1_id>,
      ...
    },
    ...
    {
      "id":<skill_N_id>,
      ...
    }
  ],
  "devices":[
    {
      "id":<device_1_id>,
      ...
    },
    ...
    {
      "id":<device_M_id>,
      ...
    }
  ]
}
```

If you do not include the `subresources` parameter in the request, you will get everything except the "skills" collection and "devices" collection.

**Important**

- It is also possible to apply the subresources feature to object settings and request both an object and its settings in one request.

**Selecting Subresources**

In the example above, "`subresources=*`" was specified in order to get all available subresources. If the object you are interested in has several types of subresources, it is

---

possible to choose whether you want all subresources to be returned or just some of them. This can be achieved by specifying a comma-separated list of subresources.

### Example 1

To receive a list of skills and devices associated with an agent, use the following.

#### Request:

```
GET .../api/v2/users/<user_id>?subresources=skills,devices
```

#### Response:

```
{
  "id":<user_id>,
  "firstName":<first_name>,
  ...
  "skills":[
    {
      "id":<skill_1_id>,
      ...
    },
    ...
    {
      "id":<skill_N_id>,
      ...
    }
  ],
  "devices":[
    {
      "id":<device_1_id>,
      ...
    },
    ...
    {
      "id":<device_M_id>,
      ...
    }
  ]
}
```

### Example 2

To receive a list of skills associated with an agent, use the following.

---

**Request:**

```
GET .../api/v2/users/<user_id>?subresources=skills
```

**Response:**

```
{
  "id":<user_id>,
  "firstName":<first_name>,
  ...
  "skills":[{
    "id":<skill_1_id>,
    ...
  },
  ...
  {
    "id":<skill_N_id>,
    ...
  }
}]
}
```

## Resolving URIs

### Introduction

This feature is called "resource link resolution", which allows you to read an object and all other objects it is associated with in one request. For example, if we have a device object associated with a phone number object and we want to read both of them in one request, we need to do the following:

**Request:**

```
GET .../api/v2/devices/<device_id>?resolveUri=*
```

**Response:**

```
{
  "id":<device_id>,
  "phoneNumberUri":"http://...",
  ...
  "phoneNumber":{
```

```
        "id":<phone_number_id>,
        ...
    }
}
```

In comparison, if you do not include the "resolveUris" parameter in the request, you will get everything except the "phoneNumber" object. In the example above, we specify "resolveUris=\*" to resolve all URIs. It is possible to choose whether you want all URIs to be resolved or just some of them. This can be achieved by specifying a comma-separated list of property names referring to URIs.

## Examples

### Example 1

To resolve all URIs, use "resolveUris=\*" as shown below.

#### Request:

```
GET .../api/v2/queues/<queue_id>?resolveUris=*
```

#### Response:

```
{
    "id":<queue_id>,
    "name":<queue_name>,
    ...
    "routingTemplateUri":"http://...",
    "phoneNumberUri":"http://...",
    ...
    "phoneNumber":{
        "id":<phone_number_id>,
        ...
    },
    "routingTemplate":{
        "id":<routing_template_id>,
        ...
    }
}
```

---

## Example 2

To resolve a specific URI, use "resolveUri=<uri>" as shown below

### Request:

```
GET .../api/v2/queues/<queue_id>?resolveUri=phoneNumberUri
```

### Response:

```
{
  "id":<queue_id>,
  "name":<queue_name>,
  ...
  "routingTemplateUri":"http://...",
  "phoneNumberUri":"http://...",
  ...
  "phoneNumber":{
    "id":<phone_number_id>,
    ...
  }
}
```

## Example 3

### Request:

```
GET .../api/v2/
queues/<queue_id>?resolveUri=phoneNumberUri,routingTemplateUri
```

### Response:

```
{
  "id":<queue_id>,
  "name":<queue_name>,
  ...
  "routingTemplateUri":"http://...",
  "phoneNumberUri":"http://...",
  ...
  "phoneNumber":{
    "id":<phone_number_id>,
    ...
  }
}
```

---

```
    },
    "routingTemplate": {
      "id": <routing_template_id>,
      ...
    }
  }
}
```

## User Authentication

Basic HTTP Authentication is used. Please see RFC 2617 Section 2 for reference.

## Supported Requests

The following requests are supported at this time:

- /devices: fields=\*
- /features: fields=\*
- /me: subresources=\*
- /me/calls: fields=\*
- /me/devices: fields=\*
- /me/skills: fields=\*
- /skills: fields=\*
- /system/features: fields=\*
- /system/routing-templates: channel, version (these are query parameters), fields=\*
- /users: fields=\*, subresources=\*
- /users/{id}: subresources=\*
- /users/{id}/devices: fields=\*
- 

---

## Return Values

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

All [Genesys Web Services REST API](#) methods return a result for each operation in addition to the HTTP status code. The results are different depending on the type of operation.



---

## All Methods

All methods always return the `statusCode` attribute . If an error occurs, that is, if the `statusCode` is not 0, the response includes error details in the `statusMessage` attribute.

The following status codes are supported:

Code	Description
0	The operation is successful. No <code>statusMessage</code> is provided.
1	A required parameter is missing in the request.
2	A specified parameter is not valid for the current state.
3	The operation is forbidden.
4	An internal error occurred. This could occur if an internal error occurred with Genesys Web Services or with one of the servers working with Genesys Web Services (for example: Cassandra or a Genesys Framework component).
5	The user does not have permission to perform this operation.
6	The requested resource could not be found.
7	The operation was partially successful. Returned if at least one action in a bulk operation succeeded. More information is available in the <a href="#">Partial Success</a> section.
8	Change password demanded. Genesys Web Services requested a password change for the user.
9	Processing incomplete
10	Input validation error - the provided value is not within the range of valid values
11	User requested to change read-only property
12	Unable to retrieve resource error
13	Unable to create resource error
14	Unable to delete resource error
15	Unable to update resource error
16	Unable to assign resource error
17	Unable to unassign resource error
18	Resource already exists
19	Resource already in use
20	User is not authenticated. Any subsequent request should provide credentials

---

If an error occurs during an operation, the response includes `statusCode` and `statusMessage` to clarify the error. No other attributes are included.

Note that if an error occurs during a request, you can assume that the request failed to modify the data of the contact center.

## GET

GET requests are used to retrieve a variety of information and the response body will depend on what is being requested as well as the request parameters.

These are the possible scenarios:

1. If retrieving a collection of URIs, the response will include the array attribute `uris` which will hold the requested collection and collection or relative uris with array attribute `paths`.
2. If retrieving a collection of resources, the response will include an array attribute named after the requested resource (for example: `GET /users?fields=*` will contain `"users":[ {..user1..}, {...user2...}, and so forth]`
3. If the URI is a singular resource (for example: `GET /users/{id}`) the response will include an attribute named after the singular of the requested resource which will contain the requested value. (for example: `GET /users/{id}` will return `"user":{...user..}`)

## Example

If retrieving a collection of URIs, the response will include the array attribute `uris` which will hold the requested collection.

```
GET /skills
{
  "statusCode": 0,
  "uris": [
    "http://../api/v2/skills/123",
    "http://../api/v2/skills/456",
    ...
  ],
  "paths": [
    "/skills/123",
    "/skills/456",
  ]
}
```

```
        ...
    ]
}
```

## Example

If retrieving a collection of resources, the response will include an array attribute named after the requested resource. For example, `GET /users?fields=*` will contain `"users": [ {..user1..}, {...user2...}, etc]`.

```
GET .../users?fields=*
{
  "statusCode":0,
  "users":[
    {
      "userName":"..",
      "firstName":"...",
      etc},
    {
      "userName":"..",
      "firstName":"...",
      etc
    }
  ]
}
```

## Example

If the URI is a "singular" resource such as `GET /users/{id}`, the response includes an attribute named after the singular form of the requested resource. This attribute contains the requested value. For example, `GET /users/{id}` will return `"user":{...user..}`.

```
GET /devices/{id}
{
  "statusCode": 0,
  "device": {
    "vendor": "...",
    "phoneNumber": "...",
    ...
  }
}
```

```
}  
}
```

## POST to Create Resource

When a POST request is successful, the following extra attributes will be included:

1. `id`—the ID of the newly-created object.
2. `uri`—The URI to access the newly-created object.
3. `path`—The relative URI to access the newly created object.

## Example

### Request:

```
POST /users  
{  
  ... some user data  
}
```

### Response:

```
{  
  "statusCode":0  
  "id":"12345",  
  "uri":"http://...api/v2/users/12345"  
  "path": "/users/12345"  
}
```

## POST to Assign Resource

POST can also be used to assign one resource to another's collection, such as when assigning a skill to a user. When this is the case, no extra attributes are returned and only `statusCode:0` will be returned on success.

## DELETE

The DELETE operation does not have any extra attributes. Only `statusCode:0` will be returned on success.

## DELETE to Unassign Resource

DELETE can also be used to unassign one resource from another's collection, such as when unassigning a skill from a user. No extra attributes are returned and only `statusCode:0` will be returned on success.

## PUT

The PUT operation does not have any extra attributes. Only `statusCode:0` will be returned on success.

## Asynchronous Operations

Web Services supports many operations that are performed using POST on an existing resource and the response for which is sent via CometD. When POST is used to perform one of these operations, `statusCode:0` will be returned on success.

## Hybrid Operations

In order to increase API usability and minimize network traffic, multi-step operations are occasionally implemented. For instance, it is possible to create a device and assign it to a user with one operation. When hybrid operations are implemented, the methods will return all of the values required for each operation being performed. For example, POST to create a resource requires a return value of "uri" and "id" whereas POST to assign does not have any extra return values. Implementing a multi-step "create and assign" POST returns "uri", "id", and "statusCode" on successful completion.

## Partial Success

Some operations may be considered successful if they are able to perform some of their work. These operations are considered "bulk" operations and are different from "transactions", which involve multiple steps that possibly use multiple servers. An example of a transaction is "create user" which involves creating some data in Cassandra as well as Configuration Server. If one of these actions fails, Web Services considers the whole operation a failure. In contrast, an operation such as "assign multiple skills to user" is a bulk operation which consists of a series of transactions (for example, each individual skill assignment is a transaction). The general rule is that if a step of a transaction fails, Web Services considers the whole operation a failure. If at least one transaction in a bulk operation succeeds, Web Services considers this a "partial success." Note that for bulk GETs (for example, GET /users) if the result is a partial list, the response includes `statusCode:7` instead of 0. The rest of the result looks the same. For POST, PUT, and DELETE, the partial success returns have the following attributes:

Attribute	Value
-----------	-------

statusCode Always 7

succeeded An array of resource descriptors (see below). Each represents a resource for which the transaction was successful.

failed An array of failure descriptors (see below). Each represents a resource for which the transaction failed.

Attribute	Value
uri	The URI of a resource from request parameters for which the transaction succeeded. For example, if assigning multiple skills to a user, this is the URI of a skill).
path	The relative path of the resource
id	The unique identifier of the resource above.

Attribute	Value
<uids>	The attributes which uniquely identify the resource for which this transaction failed. For example, if assigning skill uris, this will be "uri." If creating a user this will be "userName." If a resource has more than one identifying attribute all should be present.
statusCode	The status code describing the reason for failure.
statusMessage	The message describing the reason for failure.

## Examples

### Assign:

```
POST /users/{id}/skills
{
  "uris":["uri1", "uri2"], "paths": ["uri3"]
}

{
  "statusCode":7 (partial success)
  "succeeded":[
    {
      "id":<id1>,
      "uri":"uri1",
      "path":"path1",
    },
    {
```

```
        "id":<id2>,
        "uri":"uri2",
        "path":"path2"
    }
  ]
  "failed":[
    {
      "statusCode":X,
      "statusMessage":"msg",
      "uri":"uri3",
      "path": "path2"
    }
  ]
}
```

**Create:**

```
POST /users
{
  "users":[
    {
      "firstName":"..",
      "lastName":"..",
      "userName":"u1", etc
    },
    {
      "firstName":"..",
      "lastName":"..",
      "userName":"u2", etc
    },
    {
      "firstName":"..",
      "lastName":"..",
      "userName":"u3", etc
    }
  ]
}

{
  "statusCode":7 (partial success)
  "succeeded":[
    {
```

---

```

        "id":<id>,
        "uri":"uri1",
        "path":"path"
    },
    {
        "id":<id>,
        "uri":"uri2",
        "path":"path2"
    }
]
"failed":[
    {
        "statusCode":3,
        "statusMessage":"Operation forbidden,
username already exists",
        "userName":"u3"
    }
]
}

```

**Delete:**

```

DELETE /users
{
    "uris":["uri1", "uri2"],"paths": ["uri3"]
}

{
    "statusCode":7 (partial success)
    "succeeded":[
        {
            "id":<id1>,
            "uri":"uri1",
            "path":"path1"
        },
        {
            "id":<id2>,
            "uri":"uri2",
            "path": "path2"
        }
    ]
    "failed":[
        {

```



```
        "statusCode":X,  
        "statusMessage":"...",  
        "uri":"uri3",  
        "path": "path2"  
    }  
}
```

---

## Subresources

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

In Version 2 of the [Genesys Web Services REST API](#), we are introducing a feature called `subresources` which allows reading the subresources of an object.

### Example - Show all subresources

For example, if we have a user object which has one or more skills and one or more devices associated twith it, and we want to read all of those in one request we need to do the following:

#### Request

```
GET .../api/v2/users/<user_id>?subresources=*
```

#### Response

```
{  
  "id":<user_id>,  
  "firstName":<first_name>,  
  ...  
  "skills":[{  
    "id":<skill_1_id>,  
    ...  
  }],  
  ...  
  {
```

```
        "id":<skill_N_id>,
        ...
    }],
    "devices":[
        {
            "id":<device_1_id>,
            ...
        },
        ...
        {
            "id":<device_M_id>,
            ...
        }
    ]
}
```

If the `subresources` parameter is not included in the request, you receive everything except the "skills" collection and "devices" collection.

### Important

- It is also possible to apply subresources feature to object settings and request both an object and its settings in one request.

## Selecting Subresources

In the example above, we specify `subresources=*` to get all available subresources.

If the object we are interested in has several types of subresources, we can choose which subresources to be returned. This could be achieved by specifying a comma-separated list of subresources.

## Example

### Request

```
GET .../api/v2/users/<user_id>?subresources=skills,devices
```

## Response

```
{
  "id":<user_id>,
  "firstName":<first_name>,
  ...
  "skills":[
    {
      "id":<skill_1_id>,
      ...
    },
    ...
    {
      "id":<skill_N_id>,
      ...
    }
  ],
  "devices":[
    {
      "id":<device_1_id>,
      ...
    },
    ...
    {
      "id":<device_M_id>,
      ...
    }
  ]
}
```

## Example 2

### Request

```
GET .../api/v2/users/<user_id>?subresources=skills
```

### Response

```
{
  "id":<user_id>,
  "firstName":<first_name>,
  ...
  "skills":[
    {
      "id":<skill_1_id>,
```

---

```
        ...
    },
    ...
    {
        "id":<skill_N_id>,
        ...
    }
}
```

## Filtering and Sorting by Subresource Properties

If an object has subresources, it is possible to filter and sort by their properties. For example: show all users with skill "Sales". This could be achieved by using subresource name (e.g. "skills") followed by a dot and its property (e.g. ".name").

### Example - Sorting by Skill Name

#### Request

```
GET .../api/v2/
users?subresources=skills&fields=id,userName&skills.name=Sales
```

#### Response

```
{
  "statusCode": 0,
  "users": [{
    "id": "3454353254324",
    "userName": "cmburns@springfieldnclear.com",
    "skills": [{
      "id": "0890689",
      "name": "Sales",
      "level": 2
    }
  ]
},
{
  "id": "3567365736736",
  "userName": "hsimpson@springfieldnclear.com,
```

```
        "skills":[{
            "id": "0890689",
            "name": "Sales",
            "level": 4
        }
    ]
}
```

## Example 2

### Request

```
GET .../api/v2/
users?subresources=skills&fields=id,userName&sortBy=skills.level&order=Descen
```

### Response

```
{
    "statusCode": 0,
    "users": [{
        "id": "3567365736736",
        "userName": "hsimpson@springfieldnclear.com",
        "skills":[{
            "id": "0890689",
            "name": "Sales",
            "level": 4
        }
    ],
    {
        "id": "3454353254324",
        "userName": "cmburns@springfieldnclear.com",
        "skills":[{
            "id": "0890689",
            "name": "Sales",
            "level": 2
        }
    ]
}
}
```

---

# Filtering

This is part of the **General** section of the **Genesys Web Services REST API**.

## Overview

This version of the **Genesys Web Services REST API** makes it possible to filter objects using request parameters when doing a `list` request.

## Example 1

### Request

```
GET .../api/v2/queues?fields=id,name,channel&channel=voice
```

Response:

```
{
  "statusCode":0,
  "queues":[{
    "id":<queue_1_id>,
    "name":<queue_1_name>,
    "channel":"voice"
  },
  ...
  {
    "id":<queue_N_id>,
    "name":<queue_N_name>,
    "channel":"voice"
  }
  ]
}
```

### Important



The filter parameter must be the same as the name of the corresponding object field.

## Example 2

You may also combine several filtering parameters to make even more constraints.

## Request

```
GET .../api/v2/system/
```

```
routing-templates?fields=*&channel=voice&version=1.0.0
```

Response:

```
{
  "statusCode":0,
  "routingTemplates":[{"
    "id":"00_RouteToSpecDestination",
    "name":"Route Call to Specified Destination",
    "description":"Routes calls to a skill or queue",
    "version":"1.0.0",
    "channel":"voice",
    "dependencies":["media", "destination"],
    "enabled":true,
    "schema": [...]
  },
  ...
  {
    "id":"07_SegmentCallerRouteToSpecDestination",
    "name":"Play Greeting, Segment Caller, and Route To
Specified Destination",
    "description":"Plays a user-configured greeting,
...",
    "version":"1.0.0",
    "channel":"voice",
    "dependencies":["media", "destination",
"data_record_type"],
    "enabled":false,
    "schema": [...]
  }
}]
}
```

### Important

- Some `list` requests may require mandatory filter parameters.

---

## Filtering and sorting users by fields and subresources

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

This section provides additional details on the filtering functionality described in [Pagination](#) and [Subresources](#).

#### Important

- The `limit` parameter in this request is **mandatory**.

The request accepts comma-delimited list of fields as a `sortBy` parameter.

This allows an API user to specify multiple sorting fields so that sorting is done by the first field specified, then by the second field specified, and so on.

#### Important

- Limitation: An API user can specify the order (Ascending/Descending) for all fields at once but it is not possible to specify ascending for first field, descending for the second, or similar.

### Example

Here is the sample request which returns users sorted by voice channel state and inside users with same state sorted by lastName.

#### Tip

- 💡 Default sort order is Ascending.

```
GET ...api/v2/users?subresources=statistics&limit=100&sortBy=statistics.channels.voice.stat
```



The list of user properties which can be used for filtering:

- firstName
- lastName
- userName
- enabled
- roles

The list of subresources fields what can be used for filtering:

Subresource	Description
devices.phoneNumber	Search for user which has assigned device with provided phone number. Please note that server stores for search purposes phone numbers in fixed format which contains digits only, thus value should not contain any non-digits.
skills.name	Search for user(s) which have an assigned skill with the name provided.
skills.level	Search for users(s) which have assigned skills with provided <code>skillLevel</code> .
statistics.channels.voice.state	Search for users with specified voice channel states. <ul style="list-style-type: none"> <li>• Accepts a comma-delimited list of values.</li> </ul>
statistics.channels.voice.displayNameSearch	for users with specified voice channel display names. <ul style="list-style-type: none"> <li>• Accepts a comma-delimited list of values.</li> </ul>
statistics.channels.voice.activity	Search for users with specified voice channel activities. <ul style="list-style-type: none"> <li>• Accepts a comma-delimited list of values.</li> </ul>
statistics.channels.voice.workMode	Search for users with specified voice channel workModes. <ul style="list-style-type: none"> <li>• Accepts a comma-delimited list of values.</li> </ul>

Here is the list of available sort fields:

- `userName`
- `firstName`
- `lastName`
- `enabled`
- `roles`
- `devices.phoneNumber`
- `skills.name`
- `skills.level`
- `statistics.channels.voice.state`
- `statistics.channels.voice.activity`
- `statistics.channels.voice.displayName`
- `statistics.channels.voice.workMode`
- `statistics.<statisticName>.value`

The `statistics.<statisticName>.value` is the value of simple statistic with name `<statisticName>` as defined in `statistics.yaml`.

---

## Cross Origin Resource Sharing (CORS) Settings

This is part of the **General** section of the **Genesys Web Services REST API**.

### Overview

Cross-Origin Resource Sharing (CORS) is a specification that enables open access across domain-boundaries.

Each contact center can define their own allow origin list through Genesys Web Services access control settings.

Genesys Web Services will filter an incoming request by merging global `allowOrigins` and contact center access control settings by using an `Admin` account.

### Operations

The following operations are available for this group:

Operation	Description	Permissions
GET	Retrieves an array of settings	Contact Center Admin

POST	Creates a new setting in this group. <code>allowedOrigins</code> is the only valid setting.	Contact Center Admin
PUT	Updates a setting.	Contact Center Admin
DELETE	Removes a setting.	Contact Center Admin

## Settings

Attribute name	Description
<code>allowedOrigins</code>	An array of valid "origins" for this contact center. The CORS filter will use this list to validate incoming requests.

### Tip



Wildcards are allowed in the context of a domain name for `allowedOrigins`, but "\*" by itself is not permitted.

## Examples

### Retrieve access control settings

```
GET /settings/access-control {
  settings: [
  {
    "name": "allowedOrigins",
    "value": ["https://cloud.genhtcc.com", "https://*.genhtcc.com"]
  }
  ]
}
```

### Add "genesys.com" to the list of domains

```
PUT /settings/access-control {
  settings:
  {
```

```
  "name": "allowedOrigins",
  "value": ["https://cloud.genhtcc.com", "https://*.genhtcc.com",
"https://*.genesys.com"]
}
```

### Important

- When sending the above, the entire array must be sent

---

## Cross Site Request Forgery Protection

### Overview

Genesys Web Services prevents Cross Site Request Forgery (CSRF) attacks by requiring a token in a custom header for all requests that modify data: PUT, POST, DELETE.

Genesys Web Services generates and stores a token along with the HTTP session. The token shares the life cycle of the HTTP session.

To obtain the CSRF token and the expected header name from Genesys Web Services, a client application sends a GET request. The expected header name and token value will be provided to the client in two custom headers on the HTTP response: X-CSRF-HEADER and X-CSRF-TOKEN.

After obtaining the token value, subsequent API requests that use PUT, POST, DELETE should include the token. If the token is not included, the API will response with an HTTP 403 response and include a message stating that the CSRF token is missing.

If CSRF is disabled in configuration, no token will be returned and no validation will take place on requests.

### Example - Authorized request returning token headers

#### Request

```
GET /api/v2/me
```

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/
```

537.36

```
Authorization: Basic cGF2ZWxkQHJlZHdpbmdzLmNvbTpwYXNzd29yZA==
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US, en; q=0.8
Cookie: JSESSIONID=hac082exio454jccqk6ieqm4j
```

## Response

```
200 - OK
Date: Mon, 23 Jun 2014 02:00:15 GMT
X-CSRF-HEADER: X-CSRF-TOKEN
Set-Cookie: JSESSIONID=1h49t997p4mgc1e108bz0cjntr; Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-CSRF-TOKEN: e2fcfafd-c600-4156-88ae-ca56babd24e1
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Transfer-Encoding: chunked
```

## Example - POST request including CSRF token

### Request

```
POST /api/v2/me

Origin: chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo
X-CSRF-TOKEN: e2fcfafd-c600-4156-88ae-ca56babd24e1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/
537.36
Content-Type: application/json
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US, en; q=0.8
Cookie: JSESSIONID=1h49t997p4mgc1e108bz0cjntr

{ "operationName": "Ready" }
```

### Response

---

---

```
200 - OK
Date: Mon, 23 Jun 2014 02:02:51 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(8.1.14.v20131031)
```

```
{ "statusCode": 0 }
```

[Javascript](#) and [Python](#) examples are also available in our [Getting Started](#) section.

---

## Features

This is part of the [General](#) section of the [Genesys Web Services REST API](#).

### Overview

The features resource allows the client application to determine which functionality is available in the current contact center. This data can then be used to draw the UI as appropriate for the feature set that is supported for the current contact center.

A *feature* represents a set of functionality that may include channels, services, resources, sets of operations, settings groups, and so on. Anything that is needed for the feature to function successfully should be available when a feature is enabled for the contact center. When a feature is disabled, the API behaves as if this set of functionality does not exist. This returns results such as 404 errors when relevant resources are accessed, settings groups are not visible in lists, and operations return with `invalid operation` errors.

### Operations

Two resources are available in the API to support this functionality:

- `api/v2/system/features` represents all features available in the system.
- `api/v2/features` represents the set of features for a given contact center.

The following operations are available for `/features`

Operation	Description	Permissions
-----------	-------------	-------------

---

	Returns a list of URIs for the features assigned to this contact center.	
GET	The parameter <code>fields=*</code> causes full feature descriptions to be returned instead of URIs.	<ul style="list-style-type: none"> <li>• Contact Center Admin</li> <li>• Agent</li> </ul>

The following operations are available for `/system/features`

Operation	Description	Permissions
GET	Returns a list of URIs for all of the features available in the system.  The parameter <code>fields=*</code> causes full feature descriptions to be returned instead of URIs.	<ul style="list-style-type: none"> <li>• Contact Center Admin</li> </ul>

### Important

- The full feature set is defined by Web Services and is not modifiable.

The following operations are available for `/features/{id}`

Operation	Description	Permissions
GET	Returns the full feature description.	<ul style="list-style-type: none"> <li>• Contact Center Admin</li> <li>• Agent</li> </ul>
DELETE	Removes the feature from the contact center.	<ul style="list-style-type: none"> <li>• Cloud Admin</li> </ul>

## Attributes

The following attributes are supported for each feature:

Attribute	Type	Description	Access
<code>id</code>	String	The name of the feature (this is also the unique identifier and should be in a URI-compatible format).	GET
<code>displayName</code>	String	Name that describes the feature.	GET

description String Description of the feature. GET

## Supported Features

The following features are currently supported. If a feature is marked **Y** under **Assigned by default**, all contact centers will have this feature assigned.

### Important

- Currently, default features cannot be unassigned.

Name	Description	Assigned by default
api-provisioning-read	General provisioning read	Y
api-provisioning-write	General provisioning write	Y
api-voice	API for the voice channel	Y
ui-supervisor-provisioning-routing	Supervisor UI to provision routing	Y
ui-supervisor-provisioning-skill	UI to provision skills	Y
ui-supervisor-provisioning-user	Supervisor UI to display 'Agents' tab	Y



ui-supervisor-edit-  
user

Supervisor UI to allow editing of users

Y

---

## Services

This is part of the **General** section of the **Genesys Web Services REST API**.

---

## Overview

The **Services** resource provides a list of the services available in the system as well as their statuses and any information that is necessary to interact with the service (for example, a public SIP port for a "Voice" service). These services represent various aspects of Web Services that (in most cases) correspond to internal Genesys servers. For instance, each "Voice" service corresponds to a TServer,

and so on. A UI application can use this information to draw portions of the screen based on the status of a specific service. For instance, if the Provisioning service is "read only" the UI should disable all write operations but allow reading of provisioning data.

## Operations

The following operations are available for **/services**:

Operation	Description	Permissions
GET	Returns a list of service URIs or actual service resources if the <code>fields</code> parameter is specified. The list is populated based on the features currently enabled for the contact center.	<ul style="list-style-type: none"> <li>Agent</li> <li>Supervisor</li> <li>Contact Center Admin</li> </ul>

The following operations are available for **/services/{id}**:

Operation	Description	Permissions
GET	Returns information about the specified service.	<ul style="list-style-type: none"> <li>Agent</li> <li>Supervisor</li> <li>Contact Center Admin</li> </ul>

## Attributes

The following attributes are currently available for each service resource:

Attribute	Type	Description	Access	Applies To
id	String	A unique string identifying the service	GET	All Services
name	String	The service name. This will be equal to the name of the corresponding server's application object in Configuration Server. Note that in case there is a primary/backup pair, the primary server's	GET	All Services

application name will be used regardless of which instance is currently running. For the "Provisioning" service, the value will always be set to Provisioning.

One of the following:

type	String	Provisioning (CME + Cassandra)   Voice (T-Server)	GET	All Services
------	--------	---------------------------------------------------	-----	--------------

The service's current state. Possible values are:

state	String	Active   Inactive   ReadOnly (where applicable)	GET	All Services
-------	--------	-------------------------------------------------	-----	--------------

## Notifications

The client application can subscribe to the topic **/notifications/services** in order to receive service state change notifications. The following attributes will be present in a service state change notification:

Attribute	Value Type	Description
messageType	String	Will always be <code>ServiceStateChangeMessage</code>
service	Service Resource	The service resource for which the state has been changed

## Examples

```
GET /services?fields=*
{
  "statusCode": 0,
  "services": [{
    "id": "..",
    "name": "Provisioning",
    "type": "Provisioning",
    "state": "Active"
  },
  {
    "id": "..",
    "name": "SIPS1",
    "type": "Voice",
```

```
        "state": "Active"
    },
    {
        "id": "..",
        "name": "SIPS2",
        "type": "Voice",
        "state": "Active"
    },
}
```

GET /services/<service\_id>

```
{
    "statusCode": 0,
    "service": [{
        "id": <service_id>,
        "name": "Provisioning",
        "type": "Provisioning",
        "state": "Active" }
    ]
}
```

# Voice API

## RESTRICTED

### Important

- This is **restricted release** documentation, and therefore is subject to change and is not complete. Some features that are described in this section might not be fully implemented in the application.

This document describes the Voice API portion of the [Genesys Web Services REST API](#) and provides guidance for developers building voice-related client applications.

### Calls

---

Dial  
Answer  
Reject  
Hold  
Retrieve  
Hangup

### Calls (continued)

---

SendDTMF  
MuteCall  
UnMuteCall  
SetCallDisposition

### Call Forwarding

---

ForwardCallsOn  
ForwardCallsOff

### Conferences

---

SingleStepConference  
InitiateConference  
CompleteConference  
RemoveParticipantFromConference  
SwapCalls  
MergeWithOtherCall

## Transfers

---

SingleStepTransfer  
InitiateTransfer  
CompleteTransfer

## Call Data

---

AttachUserData  
UpdateUserData  
DeleteUserData  
DeleteUserDataPair

## Agent State

---

Ready  
NotReady  
AuxWork  
AfterCallWork  
Offline  
DoNotDisturbOn  
DoNotDisturbOff

## Supervisor

---

ListenIn  
Coach  
Bargeln  
CancelSupervisionMonitoring

## Supervisor (continued)

---

SwitchToBargeln  
SwitchToListenin  
MuteMonitoredUser  
UnmuteMonitoredUser

## Session Management

---

StartContactCenterSession  
EndContactCenterSession

---

## Calls

This is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Calls

---

Dial  
Answer  
Reject  
Hold  
Retrieve  
Hangup

### Calls (continued)

---

SendDTMF  
MuteCall  
UnMuteCall  
SetCallDisposition

## Dial

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Initiates a new outbound call to the specified destination.

---

<b>Request URL</b>	/api/v2/me/devices/{id}/calls
--------------------	-------------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

## Parameters

Parameter	Value
operationName	Dial
destination	A JSON object that includes the number to be dialed: <pre>{   "phoneNumber": "5551212" }</pre>
userData	An optional JSON object that includes key/value data to be included with the call: <pre>{   "Product": "Guitars" }</pre>

## Sample #1

### Request

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a/calls
```

```
{  
  "operationName": "Dial",  
  "destination": {  
    "phoneNumber": "5001"  
  }  
}
```



---

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec010",
      "state": "Dialing",
      "callUuid": "013V08JRL498H1OI04000VTAES00000G",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec010",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "SendDtmf",
        "UpdateUserData"
      ],
      "duration": "0",

```

---

```
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Sample #2

This sample includes a key/value pair with the `Dial` request:

### Request

```
POST api/v2/me/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a/calls
{
  "operationName": "Dial",
  "destination": {
    "phoneNumber": "5001"
  },
  "userData": {
    "CustomerSegment": "Gold"
  }
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",

```

```
"notificationType": "StatusChange",
"call": {
  "id": "0071023821aec015",
  "state": "Dialing",
  "callUuid": "013V08JRL498H1OI04000VTAES00000L",
  "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
  "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec015",
  "participants": [
    "5001"
  ],
  "participantsInfo": [
    {
      "digits": "5001",
      "e164Number": "5001",
      "formattedPhoneNumber": "5001"
    }
  ],
  "dnis": "5001",
  "callType": "Internal",
  "capabilities": [
    "DeleteUserDataPair",
    "AttachUserData",
    "Hangup",
    "DeleteUserData",
    "SendDtmf",
    "UpdateUserData"
  ],
  "userData": {
    "CustomerSegment": "Gold"
  },
  "duration": "0",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

---

## Answer

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Answers a ringing call.

**Note:** When a device is created with `telephonyNetwork = Public`, the `Answer` operation will not be included in capabilities for incoming calls on the device. When `telephonyNetwork` is set to `Public`, the assumption is that an answer request via CTI is not supported.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
--------------------	------------------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
-----------	-------

operationName	Answer
---------------	--------

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec011
{
  "operationName": "Answer"
}
```

#### Response

```
{
  "statusCode": 0
}
```

---

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec011",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000H",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec011",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "Hold",
        "SendDtmf",
        "InitiateConference",
        "InitiateTransfer",
        "SingleStepConference",
        "UpdateUserData",
        "SingleStepTransfer"
      ],
      "duration": "14",
      "mute": "Off",
      "supervisorListeningIn": false,

```

```
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Reject

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Rejects a ringing call.

When a call is rejected, it is redirected to the queue it was delivered from.

This operation is only available when a call is delivered via a queue. Reject is not possible on direct calls.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
-----------	-------

operationName	Reject
---------------	--------

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec012
{
  "operationName": "Reject"
}
```

---

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec012",
      "state": "Released",
      "callUuid": "013V08JRL498H1OI04000VTAES00000I",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/9c14cad7-17c4-48d0-8492-7cf0ff92c224",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec012",
      "participants": [
        "5005"
      ],
      "participantsInfo": [
        {
          "digits": "5005",
          "e164Number": "5005",
          "formattedPhoneNumber": "5005"
        }
      ],
      "dnis": "9000",
      "callType": "Internal",
      "capabilities": [],
      "duration": "5",
      "mute": "Off",
      "supervisorListeningIn": false,
      "monitoredUserMuted": false
    },
    "phoneNumber": "5001"
  },
}
```

---

```
"channel": "/v2/me/calls"  
}
```

## Hold

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Places a call on hold.

<b>Request URL</b>	/api/v2/me/calls/{id}
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice

### Parameters

Parameter	Value
operationName	Hold

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec013  
{  
  "operationName": "Hold"  
}
```

#### Response

```
{  
  "statusCode": 0  
}
```



---

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec013",
      "state": "Held",
      "callUuid": "013V08JRL498H1OI04000VTAES00000J",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec013",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "Retrieve",
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "InitiateConference",
        "InitiateTransfer",
        "SingleStepConference",
        "UpdateUserData",
        "SingleStepTransfer"
      ],
      "duration": "48",
      "mute": "Off",
      "supervisorListeningIn": false,
      "monitoredUserMuted": false
    }
  }
}
```

```
    },
    "phoneNumber": "5005"
  },
  "channel": "/v2/me/calls"
}
```

## Retrieve

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Retrieves a call from hold.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
-----------	-------

operationName	Retrieve
---------------	----------

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec013
{
  "operationName": "Retrieve"
}
```

---

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec013",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000J",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec013",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "Hold",
        "SendDtmf",
        "InitiateConference",
        "InitiateTransfer",

```

```
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "duration": "254",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Hangup

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Hangs up a call.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	Hangup

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec013
{
```

```
"operationName": "Hangup"
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec013",
      "state": "Released",
      "callUuid": "013V08JRL498H1OI04000VTAES00000J",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec013",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [],
      "duration": "599",
      "mute": "Off",
      "supervisorListeningIn": false,
      "monitoredUserMuted": false
    },
  },
}
```

```
    "phoneNumber": "5005"
  },
  "channel": "/v2/me/calls"
}
```

## SendDTMF

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Sends the provided DTMF digits. DTMF digits can be sent individually with multiple requests or together with multiple digits in one request.

**NOTE:** When using SIP Server, the `sip-dtmf-send-rtp` option may need to be set to true for this request to succeed. Consult SIP Server documentation for more details.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
--------------------	------------------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	SendDtmf
digits	A string consisting of one or more digits (0-9)

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec014
{
  "operationName": "SendDtmf",
  "digits": "7"
}
```

---

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "DtmfSent",
    "call": {
      "id": "0071023821aec014",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000K",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec014",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "Hold",
        "SendDtmf",
        "InitiateConference",
        "InitiateTransfer",

```

```
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "duration": "134",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## MuteCall

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Mutes the call.

While mute is on, the agent can hear the other participants on the call but other participants cannot hear the agent.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
-----------	-------

operationName	MuteCall
---------------	----------



---

## Sample

### Request

```
POST api/v2/me/calls/0071023821aec014
{
  "operationName": "MuteCall"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

## UnmuteCall

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Unmutes a previously muted call.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	UnmuteCall

---

## Sample

### Request

```
POST api/v2/me/calls/0071023821aec014
{
  "operationName": "UnmuteCall"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

## SetCallDisposition

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Sets the disposition for the specified call using the provided parameters.

If the call is active at the time of the request, it will be processed by updating the key in the user data.

If the call has already been released, then `EventUserEvent` will be sent to propagate the disposition to the reporting solution.

<b>Request URL</b>	/api/v2/me/calls/{id}
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice

---

## Parameters

Parameter	Value
operationName	SetCallDisposition
callUri	The URI of the call to disposition.
callUuid	The UUID of the call to disposition. This can be obtained from the call resource.
disposition	A string value to be used for the call disposition.
dispositionKey	An optional string value to be used for the as the <code>userdata</code> key for the call disposition. If not specified, the disposition key configured for the server will be used.

## Sample

### Request

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a
(SetCallDisposition)
{
  "operationName": "SetCallDisposition",
  "callUri": "http://localhost:8080/api/v2/calls/007102385535e00e",
  "callUuid": "011DJV5JI898NB2L04000VTAES00000I",
  "disposition": "IssueResolved",
  "dispositionKey": "DispositionCode"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

If this request is sent while the call is still active, a `CallStateChangeMessage` will be delivered. If the call has been released there will not be a notification.

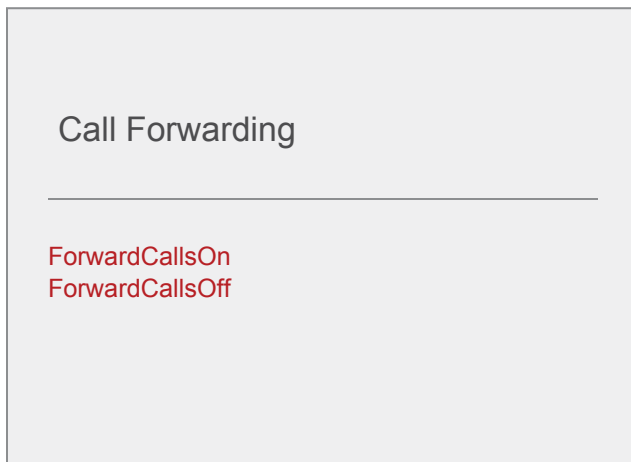
```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "AttachedDataChanged",
    "call": {
      "id": "007102385535e00e",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000I",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e00e",
      "participants": [
        "5000",
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        },
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Internal",
      "capabilities": [
        "AttachUserData",
        "InitiateConference",
        "UpdateUserData",
        "Hold",
        "SingleStepTransfer",
        "DeleteUserData",
        "SingleStepConference",
        "Hangup",
        "DeleteUserDataPair",
        "SendDtmf",
        "InitiateTransfer",
```

```
    "SwitchToListenIn",
    "RemoveParticipantFromConference"
  ],
  "userData": {
    "DispositionCode": "IssueResolved"
  },
  "duration": "1173",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

---

## Call Forwarding

This is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).



### ForwardCallsOn

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

---

## Overview

Sets call forwarding on the device to the specified destination.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

## Parameters

Parameter	Value
-----------	-------

operationName	ForwardCallsOn
---------------	----------------

destination	The number to forward calls to.
-------------	---------------------------------

## Sample

### Request

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a
{
  "operationName": "ForwardCallsOn",
  "destination": "5001"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
```

```

{
  "id": "efe1ab32-53f9-43ce-b65e-5768c61f7d4a",
  "deviceState": "Active",
  "userState": {
    "id": "9430250E-0A1B-421F-B372-F29E69366DED",
    "displayName": "Ready",
    "state": "Ready"
  },
  "phoneNumber": "5005",
  "e164Number": "5005",
  "telephonyNetwork": "Private",
  "doNotDisturb": "Off",
  "forwardTo": "5001",
  "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
  "capabilities": [
    "ForwardCallsOff",
    "DoNotDisturbOn",
    "ListenIn",
    "Coach",
    "BargeIn"
  ]
}
]
},
"channel": "/v2/me/devices"
}

```

## ForwardCallsOff

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Cancels call forwarding for a device.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

---

## Parameters

Parameter	Value
operationName	ForwardCallsOff

## Sample

### Request

```
POST api/v2/me/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a
{
  "operationName": "ForwardCallsOn",
  "destination": "5001"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "efelab32-53f9-43ce-b65e-5768c61f7d4a",
        "deviceState": "Active",
        "userState": {
          "id": "9430250E-0A1B-421F-B372-F29E69366DED",
          "displayName": "Ready",
          "state": "Ready"
        },
        "phoneNumber": "5005",
        "e164Number": "5005",
        "telephonyNetwork": "Private",
      }
    ]
  }
}
```



```
    "doNotDisturb": "Off",
    "forwardTo": "5001",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/
voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOff",
      "DoNotDisturbOn",
      "ListenIn",
      "Coach",
      "BargeIn"
    ]
  }
]
},
"channel": "/v2/me/devices"
}
```

---

## Conferences and Transfers

This is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

Conferences	Transfers
<ul style="list-style-type: none"><li>SingleStepConference</li><li>InitiateConference</li><li>CompleteConference</li><li>RemoveParticipantFromConference</li><li>SwapCalls</li><li>MergeWithOtherCall</li></ul>	<ul style="list-style-type: none"><li>SingleStepTransfer</li><li>InitiateTransfer</li><li>CompleteTransfer</li></ul>

### SingleStepConference

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

## Overview

Performs a single-step conference, adding the specified participant to the call.

**Request URL** /api/v2/me/calls/{id}

**HTTP Method** POST

**Required Features** api-voice

## Parameters

Parameter	Value
operationName	SingleStepConference
destination	A JSON object that includes the number to be dialed: <pre>{   "phoneNumber": "5551212" }</pre>
userData	An optional JSON object that includes key/value data to be included with the call: <pre>{   "Product": "Guitars" }</pre>

## Sample 1

### Request

```
POST api/v2/me/calls/0071023821aec014  
{  
  "operationName": "SingleStepConference",  
  "destination": {  
    "phoneNumber": "5001"  
  }  
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "ParticipantsUpdated",
    "call": {
      "id": "0071023821aec014",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000K",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec014",
      "participants": [
        "5000",
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        },
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "RemoveParticipantFromConference",

```

```
    "AttachUserData",
    "MuteCall",
    "Hangup",
    "DeleteUserData",
    "Hold",
    "SendDtmf",
    "InitiateConference",
    "InitiateTransfer",
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "userData": {
    "FirstConferencePartyDN": "5005"
  },
  "duration": "1568",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Sample 2

### Request

```
POST api/v2/me/calls/0071023821aec017
{
  "operationName": "SingleStepConference",
  "destination": {
    "phoneNumber": "5000"
  },
  "userData": {
    "AccountNumber": "12345"
  }
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "ParticipantsUpdated",
    "call": {
      "id": "0071023821aec017",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000N",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec017",
      "participants": [
        "5000",
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        },
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "RemoveParticipantFromConference",

```

```
    "AttachUserData",
    "MuteCall",
    "Hangup",
    "DeleteUserData",
    "Hold",
    "SendDtmf",
    "InitiateConference",
    "InitiateTransfer",
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "userData": {
    "AccountNumber": "12345",
    "FirstConferencePartyDN": "5005"
  },
  "duration": "43",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## InitiateConference

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Initiates a two-step conference to the specified destination.

This will place the existing call on hold and a new call will be created in the `dialing` state.

After initiating the conference, the [CompleteConference](#) operation can be used to complete the conference and bring all parties into the same call.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
--------------------	------------------------------------

---

**HTTP Method**    POST

**Required Features**    api-voice

## Parameters

Parameter	Value
operationName	InitiateConference
destination	A JSON object that includes the number to be dialed: <pre>{   "phoneNumber": "5551212" }</pre>
userData	An optional JSON object that includes key/value data to be included with the call: <pre>{   "Product": "Guitars" }</pre>

## Sample

### Request

```
POST api/v2/me/calls/007102385535e001
{
  "operationName": "InitiateConference",
  "destination": {
    "phoneNumber": "5000"
  }
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

The first notification will be that the initial call has been placed on hold:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e001",
      "state": "Held",
      "callUuid": "011DJV5JI898NB2L04000VTAES000001",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e001",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "UpdateUserData",
        "SingleStepConference",
        "DeleteUserData",
        "Hangup",
        "Retrieve",
        "AttachUserData",

```



```

        "SingleStepTransfer",
        "InitiateConference",
        "DeleteUserDataPair",
        "InitiateTransfer"
    ],
    "duration": "21",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

The second notification is a notification that the new consult call is dialing:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e002",
      "state": "Dialing",
      "callUuid": "011DJV5Ji898NB2L04000VTAES000002",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e002",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5000",
      "callType": "Consult",
      "capabilities": [

```

```
    "UpdateUserData",
    "DeleteUserData",
    "Hangup",
    "SwapCalls",
    "CompleteTransfer",
    "SendDtmf",
    "AttachUserData",
    "DeleteUserDataPair"
  ],
  "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e001",
  "duration": "0",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## CompleteConference

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Completes a previously initiated conference.

Once completed, the two separate calls are brought together and all three parties are participating in the same call.

<b>Request URL</b>	/api/v2/me/calls/{id}
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice

---

## Parameters

Parameter	Value
operationName	CompleteConference
consultCallUri	This optional parameter can be used to specify the consult call to be used to complete the conference. If not provided, Genesys Web Services will determine the related call.

## Sample

### Request

```
POST api/v2/me/calls/007102385535e002
{
  "operationName": "CompleteConference"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

The first notification will be that the consult call is released:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e002",
      "state": "Released",
      "callUuid": "011DJV5JI898NB2L04000VTAES000002",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
```

```

007102385535e002",
  "participants": [
    "5000"
  ],
  "participantsInfo": [
    {
      "digits": "5000",
      "e164Number": "5000",
      "formattedPhoneNumber": "5000"
    }
  ],
  "dnis": "5000",
  "callType": "Consult",
  "capabilities": [],
  "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e001",
  "duration": "9",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

The second notification will be that the participants for the original call have been updated to include the new party:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "ParticipantsUpdated",
    "call": {
      "id": "007102385535e001",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES000001",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e001",
      "participants": [

```

```
    "5000",
    "5001"
  ],
  "participantsInfo": [
    {
      "digits": "5000",
      "e164Number": "5000",
      "formattedPhoneNumber": "5000"
    },
    {
      "digits": "5001",
      "e164Number": "5001",
      "formattedPhoneNumber": "5001"
    }
  ],
  "dnis": "5005",
  "callType": "Internal",
  "capabilities": [
    "UpdateUserData",
    "SingleStepConference",
    "DeleteUserData",
    "Hangup",
    "RemoveParticipantFromConference",
    "SendDtmf",
    "Hold",
    "AttachUserData",
    "SingleStepTransfer",
    "InitiateConference",
    "DeleteUserDataPair",
    "MuteCall",
    "InitiateTransfer"
  ],
  "duration": "31",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## SingleStepTransfer

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Performs a single-step transfer to the specified destination.

**Request URL**     /api/v2/me/calls/{id}

**HTTP Method**    POST

**Required Features**  api-voice

### Parameters

Parameter	Value
operationName	SingleStepTransfer
destination	A JSON object that includes the number to be dialed: <pre>{   "phoneNumber": "5551212" }</pre>
userData	An optional JSON object that includes key/value data to be included with the call: <pre>{   "Product": "Guitars" }</pre>

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec014  
{  
  "operationName": "SingleStepTransfer",
```

```
"destination": {
  "phoneNumber": "5001"
}
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec014",
      "state": "Released",
      "callUuid": "013V08JRL498H1OI04000VTAES00000K",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec014",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [],
      "duration": "2745",
      "mute": "Off",
      "supervisorListeningIn": false,
    }
  }
}
```

```
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Sample 2

### Request

```
POST api/v2/me/calls/0071023821aec016
{
  "operationName": "SingleStepTransfer",
  "destination": {
    "phoneNumber": "5000"
  },
  "userData": {
    "TransferReason": "Escalation",
    "TransferAgent": "JSmith"
  }
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec016",
      "state": "Released",
      "callUuid": "013V08JRL498H1OI04000VTAES00000M",

```



```
    "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec016",
    "participants": [
      "5001"
    ],
    "participantsInfo": [
      {
        "digits": "5001",
        "e164Number": "5001",
        "formattedPhoneNumber": "5001"
      }
    ],
    "dnis": "5005",
    "callType": "Internal",
    "capabilities": [],
    "userData": {
      "TransferAgent": "JSmith",
      "TransferReason": "Escalation"
    },
    "duration": "68",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## InitiateTransfer

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Initiates a two-step transfer to the specified destination.

After initiating the transfer, the [CompleteTransfer](#) operation can be used to complete the transfer.

---

---

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

## Parameters

Parameter	Value
operationName	InitiateTransfer
destination	A JSON object that includes the number to be dialed: <pre>{   "phoneNumber": "5551212" }</pre>
userData	An optional JSON object that includes key/value data to be included with the call: <pre>{   "Product": "Guitars" }</pre>

## Sample

### Request

```
POST api/v2/me/calls/007102385535e005
{
  "operationName": "InitiateTransfer",
  "destination": {
    "phoneNumber": "5001"
  }
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

The first notification will be that the original call has been placed on hold:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Held",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "UpdateUserData",
        "SingleStepConference",
        "DeleteUserData",
        "Hangup",
        "Retrieve",
        "AttachUserData",

```

```

        "SingleStepTransfer",
        "InitiateConference",
        "DeleteUserDataPair",
        "InitiateTransfer"
    ],
    "duration": "36",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

The second notification will be of the new consult call dialing:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e006",
      "state": "Dialing",
      "callUuid": "011DJV5Ji898NB2L04000VTAES000006",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e006",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Consult",
      "capabilities": [

```

```

        "UpdateUserData",
        "DeleteUserData",
        "Hangup",
        "SwapCalls",
        "CompleteTransfer",
        "SendDtmf",
        "AttachUserData",
        "DeleteUserDataPair"
    ],
    "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
    "duration": "0",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

## Full Example Scenario

This section describes a full two-step transfer scenario from start to finish.

First, the agent receives a notification of the inbound call:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Ringing",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
      "participants": [
        "5000"
      ]
    }
  }
}

```

```

    ],
    "participantsInfo": [
      {
        "digits": "5000",
        "e164Number": "5000",
        "formattedPhoneNumber": "5000"
      }
    ],
    "dnis": "5005",
    "callType": "Internal",
    "capabilities": [
      "UpdateUserData",
      "DeleteUserData",
      "AttachUserData",
      "DeleteUserDataPair",
      "Answer"
    ],
    "duration": "0",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

The agent sends a request to answer the call:

```

POST api/v2/me/calls/007102385535e005
{
  "operationName": "Answer"
}

```

The agent receives a notification that the call has been established:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",

```

```

    "state": "Established",
    "callUuid": "011DJV5JI898NB2L04000VTAES000005",
    "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
    "participants": [
      "5000"
    ],
    "participantsInfo": [
      {
        "digits": "5000",
        "e164Number": "5000",
        "formattedPhoneNumber": "5000"
      }
    ],
    "dnis": "5005",
    "callType": "Internal",
    "capabilities": [
      "UpdateUserData",
      "SingleStepConference",
      "DeleteUserData",
      "Hangup",
      "SendDtmf",
      "Hold",
      "AttachUserData",
      "SingleStepTransfer",
      "InitiateConference",
      "DeleteUserDataPair",
      "InitiateTransfer"
    ],
    "duration": "10",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

The agent initiates the two-step transfer:

```
POST api/v2/me/calls/007102385535e005
{
  "operationName": "InitiateTransfer",
  "destination": {
    "phoneNumber": "5001"
  }
}
```

The agent receives notification that the first call has been held:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Held",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "UpdateUserData",
        "SingleStepConference",
        "DeleteUserData",
        "Hangup",
        "Retrieve",
        "AttachUserData",
        "SingleStepTransfer",
      ]
    }
  }
}
```



```

    "InitiateConference",
    "DeleteUserDataPair",
    "InitiateTransfer"
  ],
  "duration": "36",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

and that the consult call is now dialing:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e006",
      "state": "Dialing",
      "callUuid": "011DJV5JI898NB2L04000VTAES000006",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e006",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Consult",
      "capabilities": [
        "UpdateUserData",

```

```

        "DeleteUserData",
        "Hangup",
        "SwapCalls",
        "CompleteTransfer",
        "SendDtmf",
        "AttachUserData",
        "DeleteUserDataPair"
    ],
    "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
    "duration": "0",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

To complete, the agent requests completion of the transfer:

```

POST api/v2/me/calls/007102385535e006
{
  "operationName": "CompleteTransfer"
}

```

The agent then receives notification that original call has been released:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Released",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
      "participants": [

```

```

    "5000"
  ],
  "participantsInfo": [
    {
      "digits": "5000",
      "e164Number": "5000",
      "formattedPhoneNumber": "5000"
    }
  ],
  "dnis": "5005",
  "callType": "Internal",
  "capabilities": [],
  "duration": "48",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

and the consult call has been released:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e006",
      "state": "Released",
      "callUuid": "011DJV5JI898NB2L04000VTAES000006",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e006",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",

```

```

        "e164Number": "5001",
        "formattedPhoneNumber": "5001"
    }
],
    "dnis": "5001",
    "callType": "Consult",
    "capabilities": [],
    "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
    "duration": "12",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
},
    "phoneNumber": "5005"
},
    "channel": "/v2/me/calls"
}

```

## CompleteTransfer

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Completes a previously initiated two-step transfer.

**Request URL** /api/v2/me/calls/{id}

**HTTP Method** POST

**Required Features** api-voice

### Parameters

Parameter	Value
operationName	CompleteTransfer
consultCallUri	This optional parameter can be used to specify the consult call to be used to complete the transfer. If not provided, GWS will determine the related call itself.

## Sample

### Request

```
POST api/v2/me/calls/007102385535e006
{
  "operationName": "CompleteTransfer"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

After completing the transfer the agent will receive notification that the original call is released:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Released",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ]
    }
  }
}
```

```

    }
  ],
  "dnis": "5005",
  "callType": "Internal",
  "capabilities": [],
  "duration": "48",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

and that the consult call is released as well:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e006",
      "state": "Released",
      "callUuid": "011DJV5JI898NB2L04000VTAES000006",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e006",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Consult",
      "capabilities": [],

```

```
    "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
    "duration": "12",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Full Example Scenario

This section shows a full two-step transfer scenario from start to finish.

First, the agent receives a notification of the inbound call:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Ringing",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
    }
  }
}
```

```

    "callType": "Internal",
    "capabilities": [
      "UpdateUserData",
      "DeleteUserData",
      "AttachUserData",
      "DeleteUserDataPair",
      "Answer"
    ],
    "duration": "0",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

The agent sends a request to answer the call:

```

POST api/v2/me/calls/007102385535e005
{
  "operationName": "Answer"
}

```

The agent receives a notification that the call has been established:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
      "participants": [
        "5000"
      ],
    },
  },
}

```



```
    "participantsInfo": [
      {
        "digits": "5000",
        "e164Number": "5000",
        "formattedPhoneNumber": "5000"
      }
    ],
    "dnis": "5005",
    "callType": "Internal",
    "capabilities": [
      "UpdateUserData",
      "SingleStepConference",
      "DeleteUserData",
      "Hangup",
      "SendDtmf",
      "Hold",
      "AttachUserData",
      "SingleStepTransfer",
      "InitiateConference",
      "DeleteUserDataPair",
      "InitiateTransfer"
    ],
    "duration": "10",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

The agent initiates the two-step transfer:

```
POST api/v2/me/calls/007102385535e005
{
  "operationName": "InitiateTransfer",
  "destination": {
    "phoneNumber": "5001"
  }
}
```

The agent receives notification that the first call has been held:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Held",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "UpdateUserData",
        "SingleStepConference",
        "DeleteUserData",
        "Hangup",
        "Retrieve",
        "AttachUserData",
        "SingleStepTransfer",
        "InitiateConference",
        "DeleteUserDataPair",
        "InitiateTransfer"
      ],
      "duration": "36",
      "mute": "Off",
      "supervisorListeningIn": false,
      "monitoredUserMuted": false
    }
  }
}
```

```
    },
    "phoneNumber": "5005"
  },
  "channel": "/v2/me/calls"
}
```

and that the consult call is now dialing:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e006",
      "state": "Dialing",
      "callUuid": "011DJV5JI898NB2L04000VTAES000006",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e006",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Consult",
      "capabilities": [
        "UpdateUserData",
        "DeleteUserData",
        "Hangup",
        "SwapCalls",
        "CompleteTransfer",
        "SendDtmf",
        "AttachUserData",
        "DeleteUserDataPair"
      ]
    },
  },
}
```

```

    "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
    "duration": "0",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"pre>
}

```

To complete, the agent requests completion of the transfer:

```

POST api/v2/me/calls/007102385535e006
{
  "operationName": "CompleteTransfer"
}

```

The agent then receives notification that original call has been released:

```

<{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e005",
      "state": "Released",
      "callUuid": "011DJV5JI898NB2L04000VTAES000005",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e005",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ]
    }
  }
}

```

```

    }
  ],
  "dnis": "5005",
  "callType": "Internal",
  "capabilities": [],
  "duration": "48",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

and the consult call has been released:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e006",
      "state": "Released",
      "callUuid": "011DJV5JI898NB2L04000VTAES000006",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e006",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Consult",
      "capabilities": [],

```

```
    "parentCallUri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e005",
    "duration": "12",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## RemoveParticipantFromConference

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Removes the specified participant from the call.

This can only be performed by the owner of the conference call or a supervisor during monitoring.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	RemoveParticipantFromConference
participant	The participant to be removed from the call.

## Sample

### Request

```
POST api/v2/me/calls/0071023821aec014
{
  "operationName": "RemoveParticipantFromConference",
  "participant": "5001"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "ParticipantsUpdated",
    "call": {
      "id": "0071023821aec014",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000K",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec014",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
    ],
  }
}
```

---

```
"dnis": "5005",
"callType": "Internal",
"capabilities": [
  "DeleteUserDataPair",
  "AttachUserData",
  "Hangup",
  "DeleteUserData",
  "Hold",
  "SendDtmf",
  "InitiateConference",
  "InitiateTransfer",
  "SingleStepConference",
  "UpdateUserData",
  "SingleStepTransfer"
],
"duration": "2511",
"mute": "Off",
"supervisorListeningIn": false,
"monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## SwapCalls

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Swaps between two calls when one call is held and the other is established.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------



## Parameters

Parameter	Value
operationName	SwapCalls
	The uri of the call to swap with.
otherCallUri	Example: <code>http://localhost:8080/api/v2/me/calls/{id}</code>

## Sample

### Request

```
POST api/v2/me/calls/0071023821aec01c
{
  "operationName": "SwapCalls",
  "otherCallUri": "http://localhost:8080/api/v2/me/calls/0071023821aec01b"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notifications

The SwapCalls operations results in two notifications. One for the call that is placed on hold, and a second for the held call is retrieved.

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
```

```
    "id": "0071023821aec01b",
    "state": "Established",
    "callUuid": "013V08JRL498H1OI04000VTAES00000R",
    "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01b",
    "participants": [
      "5000"
    ],
    "participantsInfo": [
      {
        "digits": "5000",
        "e164Number": "5000",
        "formattedPhoneNumber": "5000"
      }
    ],
    "dnis": "5005",
    "callType": "Internal",
    "capabilities": [
      "DeleteUserDataPair",
      "AttachUserData",
      "Hangup",
      "DeleteUserData",
      "Hold",
      "SendDtmf",
      "InitiateConference",
      "SwapCalls",
      "InitiateTransfer",
      "SingleStepConference",
      "UpdateUserData",
      "SingleStepTransfer",
      "CompleteTransfer",
      "CompleteConference"
    ],
    "duration": "60",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
```

---

```
"channel": "/v2/me/calls"
}

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01c",
      "state": "Held",
      "callUuid": "013V08JRL498H1OI04000VTAES00000S",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01c",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Internal",
      "capabilities": [
        "Retrieve",
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "InitiateConference",
        "SwapCalls",
        "InitiateTransfer",
        "SingleStepConference",
        "UpdateUserData",
        "SingleStepTransfer",
        "CompleteTransfer",
        "CompleteConference"
      ],
    },
  },
}
```

---

```
    "duration": "41",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Full Example Scenario

The following outlines the full set of requests and events that demonstrate using the SwapCalls operation.

The agent receives a notification of an inbound call:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01b",
      "state": "Ringing",
      "callUuid": "013V08JRL498H1OI04000VTAES00000R",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01b",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",

```

```

    "capabilities": [
      "Answer",
      "DeleteUserDataPair",
      "AttachUserData",
      "DeleteUserData",
      "UpdateUserData"
    ],
    "duration": "0",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

The agent answers the inbound call and receives notification of the call state change:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01b",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000R",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec01b",
      "participants": [
        "5000"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        }
      ],
      "dnis": "5005",

```

```
    "callType": "Internal",
    "capabilities": [
      "DeleteUserDataPair",
      "AttachUserData",
      "Hangup",
      "DeleteUserData",
      "Hold",
      "SendDtmf",
      "InitiateConference",
      "InitiateTransfer",
      "SingleStepConference",
      "UpdateUserData",
      "SingleStepTransfer"
    ],
    "duration": "5",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

The agent requests the inbound call be placed on hold:

```
POST api/v2/me/calls/0071023821aec01b
{
  "operationName": "Hold"
}
```

Notification is received of the call being held:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01b",
      "state": "Held",
      "callUuid": "013V08JRL498H10I04000VTAES00000R",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/"
    }
  }
}
```

---

```
efe1ab32-53f9-43ce-b65e-5768c61f7d4a",
  "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01b",
  "participants": [
    "5000"
  ],
  "participantsInfo": [
    {
      "digits": "5000",
      "e164Number": "5000",
      "formattedPhoneNumber": "5000"
    }
  ],
  "dnis": "5005",
  "callType": "Internal",
  "capabilities": [
    "Retrieve",
    "DeleteUserDataPair",
    "AttachUserData",
    "Hangup",
    "DeleteUserData",
    "InitiateConference",
    "InitiateTransfer",
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "duration": "10",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

The agent requests the second call be made:

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a/calls
{
  "operationName": "Dial",
```

```
"destination": {
  "phoneNumber": "5001"
}
}
```

The agent receives notification that the second call is dialing:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01c",
      "state": "Dialing",
      "callUuid": "013V08JRL498H1OI04000VTAES00000S",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01c",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "SendDtmf",
        "SwapCalls",
        "UpdateUserData",
        "CompleteTransfer"
      ],
      "duration": "0",
```



```

    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

and then notification that the second call is answered:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01c",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000S",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01c",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "Hold",
        "SendDtmf",

```

```
    "InitiateConference",
    "SwapCalls",
    "InitiateTransfer",
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer",
    "CompleteTransfer",
    "CompleteConference"
  ],
  "duration": "10",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

The agent then sends the `SwapCalls` request:

```
POST api/v2/me/calls/0071023821aec01c
{
  "operationName": "SwapCalls",
  "otherCallUri": "http://localhost:8080/api/v2/me/calls/
0071023821aec01b"
}
```

The agent then receives notification that the second call is now held:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01c",
      "state": "Held",
      "callUuid": "013V08JRL498H1OI04000VTAES00000S",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01c",
```

```

    "participants": [
      "5001"
    ],
    "participantsInfo": [
      {
        "digits": "5001",
        "e164Number": "5001",
        "formattedPhoneNumber": "5001"
      }
    ],
    "dnis": "5001",
    "callType": "Internal",
    "capabilities": [
      "Retrieve",
      "DeleteUserDataPair",
      "AttachUserData",
      "Hangup",
      "DeleteUserData",
      "InitiateConference",
      "InitiateTransfer",
      "SingleStepConference",
      "UpdateUserData",
      "SingleStepTransfer"
    ],
    "duration": "41",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

and to complete the operation, a notification that the first call is now be retrieved from hold:

```

{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "0071023821aec01b",

```

---

```
    "state": "Established",
    "callUuid": "013V08JRL498H1OI04000VTAES00000R",
    "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01b",
    "participants": [
      "5000"
    ],
    "participantsInfo": [
      {
        "digits": "5000",
        "e164Number": "5000",
        "formattedPhoneNumber": "5000"
      }
    ],
    "dnis": "5005",
    "callType": "Internal",
    "capabilities": [
      "DeleteUserDataPair",
      "AttachUserData",
      "Hangup",
      "DeleteUserData",
      "Hold",
      "SendDtmf",
      "InitiateConference",
      "SwapCalls",
      "InitiateTransfer",
      "SingleStepConference",
      "UpdateUserData",
      "SingleStepTransfer",
      "CompleteTransfer",
      "CompleteConference"
    ],
    "duration": "60",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
```

---

```
"channel": "/v2/me/calls"  
}
```

## MergeWithOtherCall

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Merges the call in the URL with the one specified in the `otherCallUri` parameter.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
--------------------	------------------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	MergeWithOtherCall
otherCallUri	The uri of the call to merge with. Example: <code>http://localhost:8080/api/v2/me/calls/{id}</code>

### Sample

#### Request

#### Response

```
{  
  "statusCode": 0  
}
```

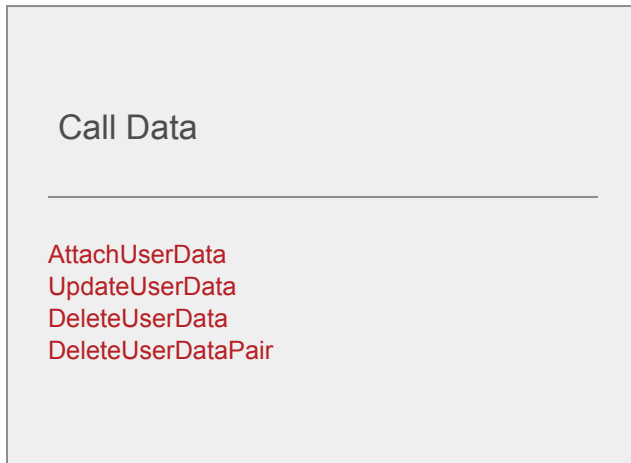
---

## Notification

---

# Call Data

This is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).



## AttachUserData

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Attaches the provided data to the call.

This operation will attach the data regardless of whether or not the key(s) already exist.

#### Important

If `AttachUserData` provides keys that already exist on the call, duplicates of the same key will exist in the call data. This is discouraged. Unless you have a specific requirement to allow duplicate keys in the call data, use `UpdateUserData` instead.

---

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

## Parameters

Parameter	Value
-----------	-------

operationName	AttachUserData
---------------	----------------

## Sample

### Request

```
POST api/v2/me/calls/007102385535e00b
{
  "operationName": "AttachUserData",
  "userData": {
    "Colors": "Blue"
  }
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "AttachedDataChanged",
    "call": {
      "id": "007102385535e00b",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000C",
    }
  }
}
```

---

```
    "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
74152ed8-858f-4a33-9e96-36213a678d30",
    "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e00b",
    "participants": [
      "5001"
    ],
    "participantsInfo": [
      {
        "digits": "5001",
        "e164Number": "5001",
        "formattedPhoneNumber": "5001"
      }
    ],
    "dnis": "5000",
    "callType": "Internal",
    "capabilities": [
      "AttachUserData",
      "InitiateConference",
      "UpdateUserData",
      "Hold",
      "SingleStepTransfer",
      "DeleteUserData",
      "SingleStepConference",
      "Hangup",
      "DeleteUserDataPair",
      "SendDtmf",
      "InitiateTransfer"
    ],
    "userData": {
      "Colors": "Blue"
    },
    "duration": "290",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5000"
},
"channel": "/v2/me/calls"
}
```



---

## UpdateUserData

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Adds the provided key/value pairs to the call.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	UpdateUserData
	A JSON object that includes the key/value data to be added to the call:
userData	<pre>{   "CustomerNumber": "54321" }</pre>

### Sample - Update Customer Name

#### Request

```
POST api/v2/me/calls/0071023821aec01d
{
  "operationName": "UpdateUserData",
  "userData": {
    "CustomerName": "Willard"
  }
}
```

---

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "AttachedDataChanged",
    "call": {
      "id": "0071023821aec01d",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000T",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01d",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "Hold",
        "SendDtmf",
        "InitiateConference",
        "InitiateTransfer",

```

---

```
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "userData": {
    "CustomerName": "Willard"
  },
  "duration": "28",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## Sample 2 - Adding multiple key/value pairs with one request:

### Request

```
POST api/v2/me/calls/0071023821aec01e
{
  "operationName": "UpdateUserData",
  "userData": {
    "CustomerNumber": "12345",
    "CustomerSegment": "Platinum",
    "CustomerName": "Willard"
  }
}
```

### Response

```
{
  "statusCode": 0
}
```

---

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "AttachedDataChanged",
    "call": {
      "id": "0071023821aec01e",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000U",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01e",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",
        "Hangup",
        "DeleteUserData",
        "Hold",
        "SendDtmf",
        "InitiateConference",
        "InitiateTransfer",
        "SingleStepConference",
        "UpdateUserData",
        "SingleStepTransfer"
      ],
      "userData": {
        "CustomerSegment": "Platinum",
        "CustomerNumber": "12345",

```

```
    "CustomerName": "Willard"
  },
  "duration": "63",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

## DeleteUserData

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Deletes all user data from the call.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
-----------	-------

operationName	DeleteUserData
---------------	----------------

### Sample

#### Request

```
POST api/v2/me/calls/0071023821aec01e
{
  "operationName": "DeleteUserData"
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

### Important

- Note the absence of the `userData` property in the update notification below:

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "AttachedDataChanged",
    "call": {
      "id": "0071023821aec01e",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000U",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/0071023821aec01e",
      "participants": [
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5005",
      "callType": "Internal",
      "capabilities": [
        "DeleteUserDataPair",
        "AttachUserData",

```

```

    "Hangup",
    "DeleteUserData",
    "Hold",
    "SendDtmf",
    "InitiateConference",
    "InitiateTransfer",
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "duration": "620",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

## DeleteUserDataPair

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Deletes the specified key from the call data.

<b>Request URL</b>	/api/v2/me/calls/{id}
--------------------	-----------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	DeleteUserDataPair
key	The key to be removed from the call data.

## Sample

Before sending this request, the test call has the following `userData` property:

```
"userData": {
  "CustomerSegment": "Platinum",
  "CustomerNumber": "12345",
  "CustomerName": "Willard"
}
```

## Request

```
POST api/v2/me/calls/0071023821aec01e
{
  "operationName": "DeleteUserDataPair",
  "key": "CustomerSegment"
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "AttachedDataChanged",
    "call": {
      "id": "0071023821aec01e",
      "state": "Established",
      "callUuid": "013V08JRL498H1OI04000VTAES00000U",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efe1ab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
0071023821aec01e",
      "participants": [
        "5001"
      ]
    }
  }
}
```

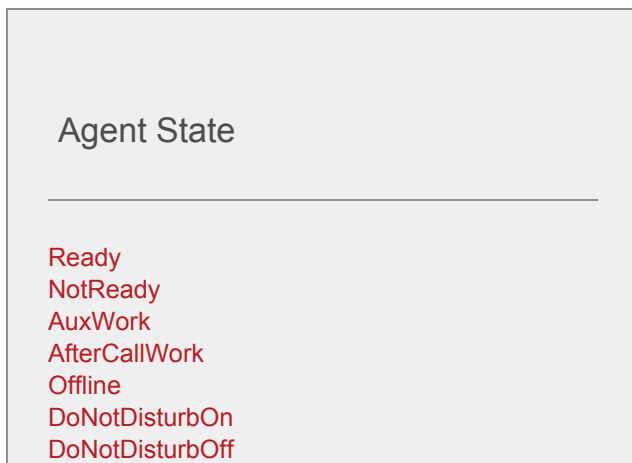


```
],
  "participantsInfo": [
    {
      "digits": "5001",
      "e164Number": "5001",
      "formattedPhoneNumber": "5001"
    }
  ],
  "dnis": "5005",
  "callType": "Internal",
  "capabilities": [
    "DeleteUserDataPair",
    "AttachUserData",
    "Hangup",
    "DeleteUserData",
    "Hold",
    "SendDtmf",
    "InitiateConference",
    "InitiateTransfer",
    "SingleStepConference",
    "UpdateUserData",
    "SingleStepTransfer"
  ],
  "userData": {
    "CustomerNumber": "12345",
    "CustomerName": "Willard"
  },
  "duration": "181",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
  "phoneNumber": "5005"
},
  "channel": "/v2/me/calls"
}
```

---

## Agent State

This is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).



## Ready

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Sets the current user to the `Ready` state.

`Ready` is a system defined agent state operation and is always available for use.

**Request URL** `/api/v2/me/calls/{id}`

**HTTP Method** `POST`

**Required Features** `api-voice`

### Parameters

Parameter	Value
<code>operationName</code>	<code>Ready</code>

---

## Sample

### Request

```
POST api/v2/me/channels/voice
{
  "operationName": "Ready"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "74152ed8-858f-4a33-9e96-36213a678d30",
        "deviceState": "Active",
        "userState": {
          "id": "9430250E-0A1B-421F-B372-F29E69366DED",
          "displayName": "Ready",
          "state": "Ready"
        },
        "phoneNumber": "5000",
        "e164Number": "5000",
        "telephonyNetwork": "Private",
        "doNotDisturb": "Off",
        "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
        "capabilities": [
          "ForwardCallsOn"
        ]
      }
    ]
  }
}
```

```
  },  
  "channel": "/v2/me/devices"  
}
```

## Not Ready

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Sets the current user to the `NotReady` state.

`NotReady` is a system defined agent state operation and is always available for use.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
--------------------	------------------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
-----------	-------

operationName	NotReady
---------------	----------

### Sample

#### Request

```
POST api/v2/me/channels/voice  
{  
  "operationName": "NotReady"  
}
```

---

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "74152ed8-858f-4a33-9e96-36213a678d30",
        "deviceState": "Active",
        "userState": {
          "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
          "displayName": "Not Ready",
          "state": "NotReady"
        },
        "phoneNumber": "5000",
        "e164Number": "5000",
        "telephonyNetwork": "Private",
        "doNotDisturb": "Off",
        "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
        "capabilities": [
          "ForwardCallsOn"
        ]
      }
    ]
  },
  "channel": "/v2/me/devices"
}
```

## Aux Work

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

---

## Overview

Sets the current user to the `NotReady` state with a workmode of `AuxWork`.

`AuxWork` is a system defined agent state operation and is always available for use.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice

## Parameters

Parameter	Value
<code>operationName</code>	<code>AuxWork</code>

## Sample

### Request

```
POST api/v2/me/channels/voice
{
  "operationName": "AuxWork"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
```

```
{
  "id": "74152ed8-858f-4a33-9e96-36213a678d30",
  "deviceState": "Active",
  "userState": {
    "id": "2B36138D-C564-4562-A8CB-3C32D564F296",
    "displayName": "AuxWork",
    "state": "NotReady",
    "workMode": "AuxWork"
  },
  "phoneNumber": "5000",
  "e164Number": "5000",
  "telephonyNetwork": "Private",
  "doNotDisturb": "Off",
  "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
  "capabilities": [
    "ForwardCallsOn"
  ]
}
],
},
"channel": "/v2/me/devices"
}
```

## After Call Work

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Sets the current user to the `NotReady` state with a workmode of `AfterCallWork`.

`AfterCallWork` is a system defined agent state operation and is always available for use.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
--------------------	------------------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

---

## Parameters

Parameter	Value
operationName	AfterCallWork

## Sample

### Request

```
POST api/v2/me/channels/voice
{
  "operationName": "AfterCallWork"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "74152ed8-858f-4a33-9e96-36213a678d30",
        "deviceState": "Active",
        "userState": {
          "id": "D3663509-3D82-4DD3-A82E-2EA8EFA02AEF",
          "displayName": "AfterCallWork",
          "state": "NotReady",
          "workMode": "AfterCallWork"
        },
        "phoneNumber": "5000",
        "e164Number": "5000",
        "telephonyNetwork": "Private",
      }
    ]
  }
}
```



```
    "doNotDisturb": "Off",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/
voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn",
      "DoNotDisturbOn"
    ]
  }
}
},
"channel": "/v2/me/devices"
}
```

## Offline

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Sets the current user to `Offline` (logged out).

`Offline` is a system defined agent state operation and is always available for use.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
--------------------	------------------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	Offline

---

## Sample

### Request

```
POST api/v2/me/channels/voice
{
  "operationName": "Offline"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "74152ed8-858f-4a33-9e96-36213a678d30",
        "deviceState": "Active",
        "userState": {
          "id": "0F7F5003-EF26-4D13-A6Ef-D0C7EC819BEB",
          "displayName": "Offline",
          "state": "LoggedOut"
        },
        "phoneNumber": "5000",
        "e164Number": "5000",
        "telephonyNetwork": "Private",
        "doNotDisturb": "Off",
        "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
        "capabilities": [
          "ForwardCallsOn",
          "DoNotDisturbOn"
        ]
      }
    ]
  }
}
```

```
    ]  
  },  
  "channel": "/v2/me/devices"  
}
```

## DoNotDisturbOn

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Turns `do-not-disturb` on for the device.

<b>Request URL</b>	<code>/api/v2/me/devices/{id}</code>
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice

### Parameters

Parameter	Value
<code>operationName</code>	<code>DoNotDisturbOn</code>

### Sample

#### Request

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a  
{  
  "operationName": "DoNotDisturbOn"  
}
```

---

## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "efelab32-53f9-43ce-b65e-5768c61f7d4a",
        "deviceState": "Active",
        "userState": {
          "id": "9430250E-0A1B-421F-B372-F29E69366DED",
          "displayName": "Ready",
          "state": "Ready"
        },
        "phoneNumber": "5005",
        "e164Number": "5005",
        "telephonyNetwork": "Private",
        "doNotDisturb": "On",
        "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
        "capabilities": [
          "DoNotDisturbOff",
          "ForwardCallsOn",
          "ListenIn",
          "Coach",
          "BargeIn"
        ]
      }
    ]
  },
  "channel": "/v2/me/devices"
}
```

---

## DoNotDisturbOff

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Turns `do-not-disturb` off for the device.

<b>Request URL</b>	<code>/api/v2/me/devices/{id}</code>
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice

### Parameters

Parameter	Value
<code>operationName</code>	DoNotDisturbOff

### Sample

#### Request

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a
{
  "operationName": "DoNotDisturbOff"
}
```

#### Response

```
{
  "statusCode": 0
}
```

#### Notification

```
{
  "data": {
```

```
"messageType": "DeviceStateChangeMessage",
"devices": [
  {
    "id": "efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "deviceState": "Active",
    "userState": {
      "id": "9430250E-0A1B-421F-B372-F29E69366DED",
      "displayName": "Ready",
      "state": "Ready"
    },
    "phoneNumber": "5005",
    "e164Number": "5005",
    "telephonyNetwork": "Private",
    "doNotDisturb": "Off",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn",
      "DoNotDisturbOn",
      "ListenIn",
      "Coach",
      "BargeIn"
    ]
  }
]
},
"channel": "/v2/me/devices"
}
```

---

## Supervisor

This is part of the Voice API section of the [Genesys Web Services REST API](#).

This section describes general concepts for supervisor functionality.

<p>Supervisor</p> <hr/> <p>ListenIn Coach Bargeln CancelSupervisionMonitoring</p>	<p>Supervisor (continued)</p> <hr/> <p>SwitchToBargeln SwitchToListenin MuteMonitoredUser UnmuteMonitoredUser</p>
-----------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------

## ListenIn

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Configures silent monitoring of the target agent device.

After sending this request, a notification will be delivered confirming the change to the supervisor monitoring state.

When the agent being monitored receives a call, that supervisor will also be delivered the call and can listen in silently.

**Request URL** /api/v2/me/devices/{id}

**HTTP Method** POST

**Required Features** api-voice, api-supervisor-monitoring

### Parameters

Parameter	Value
operationName	ListenIn
targetDeviceUri	The URI of the device to monitor.

**Example:**

```
http://localhost:8080/api/v2/devices/
9c14cad7-17c4-48d0-8492-7cf0ff92c224
```

- supervisorMonitoringScope**
- This optional parameter specifies the scope of monitoring:
- **Agent** - Only activity for the target agent will be monitored. If the monitored agent transfers the call to another agent, the supervisor will be released from the call with the agent.
  - **Call** - The supervisor will monitor the entire call, regardless of the path it takes. If the monitored agent transfers the call, the supervisor will continue to monitor as the customer is serviced by the new agent.
  - If not specified, **Call** is the default.

- supervisorMonitoringMode**
- This optional parameter specifies the monitoring mode:
- **NextCall** - Monitor only the next call the agent receives, then stop monitoring.
  - **AllCalls** - Monitoring all agent calls until monitoring is cancelled.
  - If not specified, **AllCalls** is the default.

## Sample

### Request

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a
{
  "operationName": "ListenIn",
  "targetDeviceUri": "http://localhost:8080/api/v2/devices/
9c14cad7-17c4-48d0-8492-7cf0ff92c224"
}
```



## Response

```
{
  "statusCode": 0
}
```

## Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "efelab32-53f9-43ce-b65e-5768c61f7d4a",
        "deviceState": "Active",
        "userState": {
          "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
          "displayName": "Not Ready",
          "state": "NotReady"
        },
        "phoneNumber": "5005",
        "e164Number": "5005",
        "telephonyNetwork": "Private",
        "doNotDisturb": "On",
        "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4df67e011",
        "capabilities": [
          "ForwardCallsOn",
          "DoNotDisturbOff",
          "ListenIn",
          "Coach",
          "BargeIn",
          "CancelSupervisorMonitoring"
        ],
        "supervisorMonitoringState": {
          "state": "ListenIn",
          "mode": "AllCalls",
          "scope": "Call",
          "targetDeviceUri": "http://127.0.0.1:8080/api/v2/devices/9c14cad7-17c4-48d0-8492-7cf0ff92c224"
        }
      }
    ]
  }
}
```

```

    }
  ]
},
"channel": "/v2/me/devices"
}

```

## Coach

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Configures coaching of the target agent device.

When coaching is configured and the agent receives a call, the supervisor is brought into the call.

Only the agent can hear the supervisor.

**Request URL** /api/v2/me/devices/{id}

**HTTP Method** POST

**Required Features** api-voice, api-supervisor-monitoring

### Parameters

Parameter	Value
operationName	Coach
targetDeviceUri	<p>The URI of the device to monitor.</p> <p>Example:</p> <pre>http://localhost:8080/api/v2/devices/9c14cad7-17c4-48d0-8492-7cf0ff92c224</pre>
supervisorMonitoringScope	<p>This optional parameter specifies the scope of monitoring:</p> <ul style="list-style-type: none"> <li>Agent - Only activity for the target agent will be monitored. If the monitored agent transfers the call</li> </ul>

to another agent, the supervisor will be released from the call with the agent.

- Call - The supervisor will monitor the entire call, regardless of the path it takes. If the monitored agent transfers the call, the supervisor will continue to monitor as the customer is serviced by the new agent.
- If not specified, Call is the default.

supervisorMonitoringMode

This optional parameter specifies the monitoring mode:

- NextCall - Monitor only the next call the agent receives, then stop monitoring.
- AllCalls - Monitoring all agent calls until monitoring is cancelled.
- If not specified, AllCalls is the default.

## Sample

### Request

```
POST api/v2/me/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a
{
  "operationName": "Coach",
  "targetDeviceUri": "http://localhost:8080/api/v2/devices/9c14cad7-17c4-48d0-8492-7cf0ff92c224"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
```

```

    "id": "efelab32-53f9-43ce-b65e-5768c61f7d4a",
    "deviceState": "Active",
    "userState": {
      "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
      "displayName": "Not Ready",
      "state": "NotReady"
    },
    "phoneNumber": "5005",
    "e164Number": "5005",
    "telephonyNetwork": "Private",
    "doNotDisturb": "On",
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/
voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn",
      "DoNotDisturbOff",
      "ListenIn",
      "Coach",
      "BargeIn",
      "CancelSupervisorMonitoring"
    ],
    "supervisorMonitoringState": {
      "state": "Coach",
      "mode": "AllCalls",
      "scope": "Call",
      "targetDeviceUri": "http://127.0.0.1:8080/api/v2/devices/
9c14cad7-17c4-48d0-8492-7cf0ff92c224"
    }
  }
]
},
"channel": "/v2/me/devices"
}

```

## BargeIn

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Configures barge to the target device.

If the agent is currently on a call and the TServer is configured to allow barge, the supervisor will be immediately added to the call with both the monitored agent and the customer being able to hear the supervisor.

If the target agent is not on a call at the time of the request, the supervisor will be brought into the call when the agent receives a new call.

Both the agent and customer will be able to hear and speak with the supervisor.

**Request URL** /api/v2/me/devices/{id}

**HTTP Method** POST

**Required Features** api-voice, api-supervisor-monitoring

## Parameters

Parameter	Value
operationName	Bargeln
targetDeviceUri	<p>The URI of the device to monitor.</p> <p>Example:</p> <pre>http://localhost:8080/api/v2/devices/9c14cad7-17c4-48d0-8492-7cf0ff92c224</pre>
supervisorMonitoringScope	<p>This optional parameter specifies the scope of monitoring:</p> <ul style="list-style-type: none"> <li>• Agent - Only activity for the target agent will be monitored. If the monitored agent transfers the call to another agent, the supervisor will be released from the call with the agent.</li> <li>• Call - The supervisor will monitor the entire call, regardless of the path it takes. If the monitored agent transfers the call, the supervisor will continue to monitor as the customer is serviced by the new agent.</li> <li>• If not specified, Call is the default.</li> </ul>
supervisorMonitoringMode	<p>This optional parameter specifies the monitoring mode:</p> <ul style="list-style-type: none"> <li>• NextCall - Monitor only the next call the agent receives, then stop monitoring.</li> </ul>

- AllCalls - Monitoring all agent calls until monitoring is cancelled.
- If not specified, AllCalls is the default.

## Sample

### Request

```
POST api/v2/me/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a
{
  "operationName": "BargeIn",
  "targetDeviceUri": "http://localhost:8080/api/v2/devices/
9c14cad7-17c4-48d0-8492-7cf0ff92c224"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "efelab32-53f9-43ce-b65e-5768c61f7d4a",
        "deviceState": "Active",
        "userState": {
          "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
          "displayName": "Not Ready",
          "state": "NotReady"
        },
        "phoneNumber": "5005",
        "e164Number": "5005",
        "telephonyNetwork": "Private",
        "doNotDisturb": "On",
      }
    ]
  }
}
```

```
    "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
    "capabilities": [
      "ForwardCallsOn",
      "DoNotDisturbOff",
      "ListenIn",
      "Coach",
      "BargeIn",
      "CancelSupervisorMonitoring"
    ],
    "supervisorMonitoringState": {
      "state": "BargeIn",
      "mode": "AllCalls",
      "scope": "Call",
      "targetDeviceUri": "http://127.0.0.1:8080/api/v2/devices/9c14cad7-17c4-48d0-8492-7cf0ff92c224"
    }
  }
}
},
"channel": "/v2/me/devices"
}
```

## CancelSupervisionMonitoring

### Overview

This request cancels supervisor monitoring. This request should only be sent (and will only appear in device capabilities) when supervisor monitoring is active.

<b>Request URL</b>	/api/v2/me/devices/{id}
--------------------	-------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice, api-supervisor-monitoring
--------------------------	--------------------------------------

## Parameters

Parameter	Value
operationName	CancelSupervisorMonitoring

## Sample

### Request

```
POST api/v2/me/devices/efe1ab32-53f9-43ce-b65e-5768c61f7d4a
{
  "operationName": "CancelSupervisorMonitoring"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

#### Important

- Note the absence of the `supervisorMonitoringState` property.

```
{
  "data": {
    "messageType": "DeviceStateChangeMessage",
    "devices": [
      {
        "id": "efe1ab32-53f9-43ce-b65e-5768c61f7d4a",
        "deviceState": "Active",
        "userState": {
          "id": "900D55CC-2BB0-431F-8BF9-D3525B383BE6",
          "displayName": "Not Ready",

```



```

        "state": "NotReady"
      },
      "phoneNumber": "5005",
      "e164Number": "5005",
      "telephonyNetwork": "Private",
      "doNotDisturb": "On",
      "voiceEnvironmentUri": "http://127.0.0.1:8080/api/v2/
voice-environments/370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
      "capabilities": [
        "ForwardCallsOn",
        "DoNotDisturbOff",
        "ListenIn",
        "Coach",
        "BargeIn"
      ]
    }
  ]
},
"channel": "/v2/me/devices"
}

```

## SwitchToBargeIn

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Switches from listening in to barge for the current call.

The `SwitchToBarge` operation will be available if the current monitoring mode is either `ListenIn` or `Coach`.

If switching to `BargeIn` from `Coach`, the supervisor cannot return to `Coach` and will only be able to move to `ListenIn`.

<b>Request URL</b>	<code>/api/v2/me/calls/{id}</code>
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice, api-supervisor-monitoring

---

## Parameters

Parameter	Value
operationName	SwitchToBargeIn

## Sample

### Request

```
POST api/v2/me/calls/007102385535e00e
{
  "operationName": "SwitchToBargeIn"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

#### Important

Note that `mute now` has a value of `Off`.

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e00e",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000I",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efe1ab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
```

---

```
007102385535e00e",
  "participants": [
    "5000",
    "5001"
  ],
  "participantsInfo": [
    {
      "digits": "5000",
      "e164Number": "5000",
      "formattedPhoneNumber": "5000"
    },
    {
      "digits": "5001",
      "e164Number": "5001",
      "formattedPhoneNumber": "5001"
    }
  ],
  "dnis": "5001",
  "callType": "Internal",
  "capabilities": [
    "AttachUserData",
    "InitiateConference",
    "UpdateUserData",
    "Hold",
    "SingleStepTransfer",
    "DeleteUserData",
    "SingleStepConference",
    "Hangup",
    "DeleteUserDataPair",
    "SendDtmf",
    "InitiateTransfer",
    "SwitchToListenIn",
    "RemoveParticipantFromConference"
  ],
  "duration": "21",
  "mute": "Off",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
```

```
"channel": "/v2/me/calls"  
}
```

## SwitchToListenIn

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Switches to ListenIn for the current call.

If the supervisor is switching from `Coach` to `ListenIn`, they will not be able to switch back to `Coach`.

Once in `ListenIn` they can only switch to [Bargain](#).

<b>Request URL</b>	/api/v2/me/calls/{id}
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice, api-supervisor-monitoring

### Parameters

Parameter	Value
operationName	SwitchToListenIn

### Sample

#### Request

```
POST api/v2/me/calls/007102385535e00e  
{  
  "operationName": "SwitchToListenIn"  
}
```

---

## Response

```
{
  "statusCode": 0
}
```

## Notification

Note that `mute` now has a value of `On`.

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "StatusChange",
    "call": {
      "id": "007102385535e00e",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000I",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/
efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/
007102385535e00e",
      "participants": [
        "5000",
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        },
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ],
      "dnis": "5001",
      "callType": "Internal",
      "capabilities": [
```

```

    "AttachUserData",
    "InitiateConference",
    "UpdateUserData",
    "Hold",
    "SingleStepTransfer",
    "DeleteUserData",
    "SingleStepConference",
    "Hangup",
    "DeleteUserDataPair",
    "SendDtmf",
    "InitiateTransfer",
    "SwitchToBargeIn",
    "RemoveParticipantFromConference"
  ],
  "duration": "75",
  "mute": "On",
  "supervisorListeningIn": false,
  "monitoredUserMuted": false
},
"phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}

```

## MuteMonitoredUser

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Mutes the monitored agent.

This is intended for use when a supervisor monitoring an agent needs to intervene and take over a call.

<b>Request URL</b>	/api/v2/me/calls/{id}
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice, api-supervisor-monitoring

---

## Parameters

Parameter	Value
operationName	MuteMonitoredUser

## Sample

### Request

```
POST api/v2/me/calls/007102385535e00e
{
  "operationName": "MuteMonitoredUser"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

#### Important

- Note that the `monitoredUserMuted` property is `true`.

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "MonitoredUserMutedStateChange",
    "call": {
      "id": "007102385535e00e",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000I",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
    }
  }
}
```

---

```
    "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e00e",
    "participants": [
      "5000",
      "5001"
    ],
    "participantsInfo": [
      {
        "digits": "5000",
        "e164Number": "5000",
        "formattedPhoneNumber": "5000"
      },
      {
        "digits": "5001",
        "e164Number": "5001",
        "formattedPhoneNumber": "5001"
      }
    ],
    "dnis": "5001",
    "callType": "Internal",
    "capabilities": [
      "AttachUserData",
      "InitiateConference",
      "UpdateUserData",
      "Hold",
      "SingleStepTransfer",
      "DeleteUserData",
      "SingleStepConference",
      "Hangup",
      "DeleteUserDataPair",
      "SendDtmf",
      "InitiateTransfer",
      "SwitchToListenIn",
      "RemoveParticipantFromConference"
    ],
    "duration": "348",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": true
  },
  "phoneNumber": "5005"
},
```

---



---

```
"channel": "/v2/me/calls"  
}
```

## UnmuteMonitoredUser

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Unmutes a muted monitored agent.

This is used when a supervisor monitoring an agent has taken over a call, has muted an agent, and now wants to unmute the agent.

<b>Request URL</b>	/api/v2/me/calls/{id}
<b>HTTP Method</b>	POST
<b>Required Features</b>	api-voice

### Parameters

Parameter	Value
operationName	UnmuteMonitoredUser

### Sample

#### Request

```
POST api/v2/me/calls/007102385535e00e  
{  
  "operationName": "UnmuteMonitoredUser"  
}
```

## Response

```
{
  "statusCode": 0
}
```

## Notification

### Important

- **Note that the `monitoredUserMuted` property is `false`.**

```
{
  "data": {
    "messageType": "CallStateChangeMessage",
    "notificationType": "MonitoredUserMutedStateChange",
    "call": {
      "id": "007102385535e00e",
      "state": "Established",
      "callUuid": "011DJV5JI898NB2L04000VTAES00000I",
      "deviceUri": "http://127.0.0.1:8080/api/v2/devices/efelab32-53f9-43ce-b65e-5768c61f7d4a",
      "uri": "http://127.0.0.1:8080/api/v2/me/calls/007102385535e00e",
      "participants": [
        "5000",
        "5001"
      ],
      "participantsInfo": [
        {
          "digits": "5000",
          "e164Number": "5000",
          "formattedPhoneNumber": "5000"
        },
        {
          "digits": "5001",
          "e164Number": "5001",
          "formattedPhoneNumber": "5001"
        }
      ]
    }
  }
}
```

---

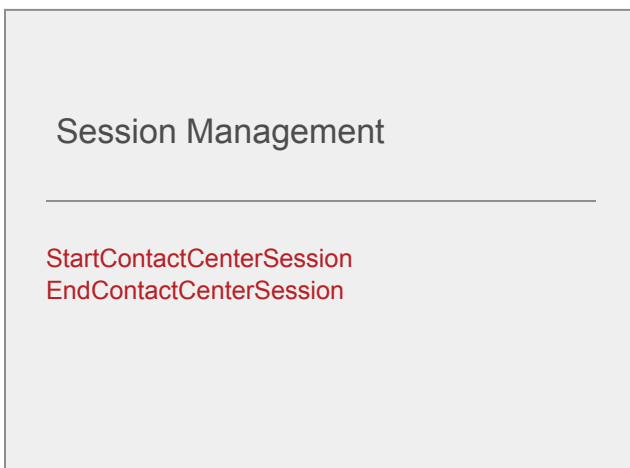
```
    ],
    "dnis": "5001",
    "callType": "Internal",
    "capabilities": [
      "AttachUserData",
      "InitiateConference",
      "UpdateUserData",
      "Hold",
      "SingleStepTransfer",
      "DeleteUserData",
      "SingleStepConference",
      "Hangup",
      "DeleteUserDataPair",
      "SendDtmf",
      "InitiateTransfer",
      "SwitchToListenIn",
      "RemoveParticipantFromConference"
    ],
    "duration": "383",
    "mute": "Off",
    "supervisorListeningIn": false,
    "monitoredUserMuted": false
  },
  "phoneNumber": "5005"
},
"channel": "/v2/me/calls"
}
```

---

## Session Management

This is part of the Voice API section of the [Genesys Web Services REST API](#).

This section describes session management.



## StartContactCenterSession

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Initializes a session for the user with the provided parameters.

Almost all client applications that provide a full set of voice features should send this operation at `login` and call [EndContactCenterSession](#) when the application exits.

**Request URL** `/api/v2/me/`

**HTTP Method** `POST`

**Required Features** `api-voice`

### Parameters

Parameter	Value
<code>operationName</code>	<code>StartContactCenterSession</code>
<code>channels</code>	An array of channels the agent will use this session. Valid channel names are <code>voice</code>
<code>place</code>	This optional parameter specifies the place that should be used for the session.  The place should have a single DN associated with it. The device corresponding to this DN will be assigned to the user for the duration of the session. When the client ends the session the default device assignment is restored. If not specified, the default place for the agent is used.
<code>loginCode</code>	This optional parameter specifies the switch login code that should be used for login.  If not specified, Genesys Web Services will look for an appropriate switch <code>login</code> that is assigned to the user for the TServer.

---

**queue** This optional parameter specifies a queue to be included in the login request.

## Sample

### Request

```
POST api/v2/me
{
  "operationName": "StartContactCenterSession",
  "channels": [
    "voice"
  ]
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

This operation may not result in a notification. If the agent is logged in to a device as a result of starting the session, a `DeviceStateChangeMessage` will be delivered via `cometd`. If the agent is already logged in, the login request is skipped and there will not be a notification.

## Sample #2

### Request

```
POST api/v2/me
{
  "operationName": "StartContactCenterSession",
  "place": "SIP_5000",
  "loginCode": "8000",
  "queue": "9000",
}
```

```
"channels": [
  "voice"
]
```

## Response

```
{
  "statusCode": 0
}
```

## EndContactCenterSession

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Ends a previously started session for the agent.

For voice, this will `logout` the agent and restore the agent's default device assignment.

This request also invalidates the client HTTP session and should be performed as the last step for session cleanup.

<b>Request URL</b>	<code>/api/v2/me/</code>
--------------------	--------------------------

<b>HTTP Method</b>	POST
--------------------	------

<b>Required Features</b>	api-voice
--------------------------	-----------

### Parameters

Parameter	Value
operationName	EndContactCenterSession

## Sample

### Request

```
POST api/v2/me
{
  "operationName": "EndContactCenterSession"
}
```

### Response

```
{
  "statusCode": 0
}
```

### Notification

There are no notifications associated with this operation.

## Mobile Push Notifications

This operation is part of the [Voice API](#) section of the [Genesys Web Services REST API](#).

### Overview

Genesys Web Services provides the capability to optionally delivery notification messages to a mobile device using Genesys Mobile Services. This allows mobile applications to be notified of state changes when they do not have an active [CometD](#) connection to Genesys Web Services.

By including key parameters in the `StartContactCenterSession` request, Genesys Web Services will have the information required to deliver notifications. When Genesys Web Services detects that the [CometD](#) connection for the application has been closed, it will start to deliver messages to the mobile application through Genesys Mobile Services using the push notification service. When the mobile application re-establishes a [CometD](#) session with Genesys Web Services, notifications will no longer be sent through Genesys Mobile Services and will continue as normal through [CometD](#).



The mobile application must also have the ability to filter messages it receives through push notifications as part of its session configuration.

The use of this functionality must be controllable on a per-contact-center basis.

### StartContactCenterSession

A new object parameter `mobileSession` has been added to the existing `StartContactCenterSession` operation. This new object contains the following properties:

Property	Description	Possible Values	Required
<code>mobileDeviceId</code>	The unique identifier of the mobile device. Y	IOS, Android (case sensitive)	Y
<code>mobilePushType</code>	The type of push notification that should be used for the device.		
<code>filter</code>	An optional filter string that can be used to control which notifications are delivered to the device. To receive all notifications, the filter <code>gws.api.v2.users.&lt;id&gt;.*</code> would be used. To receive only call related notifications - <code>gws.api.v2.users.&lt;id&gt;.calls.*</code> , and only device related notifications - <code>gws.api.v2.users.&lt;id&gt;.devices.*</code> If not specified, all notifications will be sent.	A valid Genesys Mobile Services tag filter	N
<code>timeout</code>	The maximum time the mobile device can stay disconnected from Genesys Web Services before the user's session is cleaned up. If not specified the default is <code>ThirtyMinutes</code> . <code>ThirtyMinutes</code> , <code>OneHour</code>		
<code>debug</code>	A flag that Genesys Web Services passes to Genesys Mobile Services indicating whether the client wishes to use debug mode or not. In debug mode, Genesys Mobile Services will send notifications through the developer sandbox. When not in debug mode, the	true/false	N

notifications are delivered through the production infrastructure. When not specified, this property defaults to false.

A minimal `StartContactCenterSession` request to enable mobile push notifications in the session would look like this:

```
POST /api/v2/me
{
  "operationName": "StartContactCenterSession",
  "channels": ["voice"],
  "mobileSession": {
    "mobileDeviceId": "72ED12518F714800AA900370B697A777",
    "mobilePushType": "IOS"
  }
}
```

When Genesys Web Services processes this request, it will save the parameters provided in the `mobileSession` property to the `user_sessions` column family. The row key in this cf is the user id and in addition to the provided parameters, Genesys Web Services also stores the node id (as defined in `server-settings.yaml`) and a flag indicating whether delivery of push notifications is currently active. Genesys Web Services will also send a request to Genesys Mobile Services to create a subscription at this point. The subscription is deleted when `EndContactCenterSession` is executed (either via the API, or via automated cleanup)

### Important

- An internal expiration of 3600 is currently being used when sending the subscription request. This is something that will need to be made configurable or Genesys Web Services will need to handle re-creating the subscription if it expires.

### CometD Session Notifications

Once the session information has been stored by the `StartContactCenterSession` operation, Genesys Web Services can activate push notifications for the user when it detects that their last CometD session has disconnected. When this notification is delivered,

the active flag in the user session is updated. Once updated, event processing code will start sending push notification requests to GMS.

When a new CometD session is created, if the user has data stored for their session in the `user_sessions` column family, the active flag will be updated to false to stop notifications. The node id will also be updated to help ensure that only one Genesys Web Services node takes responsibility for sending push notifications.

### Push notification format

Each of the push notification services (Apple or Android) impose limits on the size of the notification that are quite small (256 bytes for iOS). With this in mind, only a minimal set of information is extracted from the normal CometD messages and sent in the push notification. Mobile applications are expected to use this notification to wake up and then make additional GET requests to get the updated state of calls or devices.

For call related events, `CallStateChangeMessage` is translated to include the following properties:

Property	Value
<code>callId</code>	The id of the call resource that has changed. Sending a GET to <code>api/v2/me/calls/&lt;id&gt;</code> will retrieve the updated state.
<code>messageType</code>	<code>CallStateChangeMessage</code>
<code>notificationType</code>	<code>DtmfSent</code> , <code>StatusChange</code> , <code>ParticipantsUpdated</code> , <code>AttachedDataChanged</code> , or <code>MonitoredUserMutedStateChange</code>

Example:

```
{
  "callId": "0071023529204008",
  "messageType": "CallStateChangeMessage",
  "notificationType": "StatusChange"
}
```

Likewise, `DeviceStateChangeMessage` includes the following properties:

Property	Value
<code>deviceId</code>	The id of the device resource that has changed.
<code>messageType</code>	<code>DeviceStateChangeMessage</code>

**Example:**

```
{
  "deviceId": "7ac00e37-e19c-4a42-b091-b0d440f0471a",
  "messageType": "DeviceStateChangeMessage"
}
```

**Important**

- DeviceUserEventMessage, DeviceStateErrorMessage and CallStateErrorMessage are not delivered as push notifications.

**Important**

- Multiple devices assigned to a single user are not support by mobile push notifications. If more than one device is assigned to the user, notifications will be delivered using the first device only.

**Event processing**

When voice events are processed, the task looks for an entry for the user in `user_sessions`. If a session is found, the active flag is checked and a comparison is made between the stored node id and the node id of the processing Genesys Web Services node. If the flag is false or the node ids don't match, then no request is sent to GMS. When the active flag is updated via the CometD disconnect handling routine, the next event to be processed will qualify for delivery via mobile push.

**Important**

- Any errors in sending events via mobile push are logged as warnings and no exceptions are raised.

---

## User monitor and cleanup

When the CometD session from the mobile application is disconnected, our existing user monitor handles the notification. If there is no mobile session data associated with the user, the user becomes eligible for logout and cleanup after the agent logout timeout expires (either Contact Center defined or default of 60 seconds).

If a mobile sessions exists for the user, the timeout provided with `StartContactCenterSession` will be used instead. This gives the user either 30 minutes or one hour of inactivity where mobile push events will be delivered and the session will not be cleaned up. After this time expires, `EndContactCenterSession` will be run by the node as usual and the mobile session parameters are discarded.

## EndContactCenterSession

When the `EndContactCenterSession` operation is run, either explicitly by a request through the API, or due to session timeout in the user monitor, the data stored in `user_sessions` for the user is deleted.

## Implementation Plan

The initial implementation will be based on our existing CometD infrastructure and notifications. It will support the core scenario but will rely on the mobile application being the only active CometD session for a given user. It will also have potential for misbehavior in more complex node failover scenarios, depending on the specifics.

## Deployment Instructions

### Cassandra Schema

This feature uses a new column family `user_sessions` that was introduced in Genesys Web Services release 8.5.100.08. Please ensure that the `cf-schema-8.5.100.08.txt` update script has been applied to the database.

### server-settings.yaml - nodeld

Each HTCC node must define a `nodeld` in `server-settings.yaml` in order for this feature to work.

## Connecting Genesys Mobile Services with Genesys Web Services

To use mobile push, the Genesys environment must include a connection to either a specific Genesys Mobile Services node (for single node or lab deployments) or a Genesys Mobile Services cluster application. Genesys Web Services uses the `server/external_url_base` option to determine the appropriate URI to POST events to. Note that since Genesys Mobile Services uses the Generic Server or Application Cluster application types, Genesys Web Services also uses the existence of the `server/external_url_base` option to identify a connection as Genesys Mobile Services. If this option is not included, the Genesys environment will not be updated.

In premise mode, add a connection to the Genesys Mobile Services or cluster application to the connections tab of the Genesys Web Services cluster application. The same procedure can be used in cloud deployments and is likely the most convenient approach. Including details of a Genesys Mobile Services connection when creating a Genesys environment through the ops API is also supported.

### Important

- For Genesys Mobile Services, no backup information is required in the server descriptor nor are ADDP related settings.

Below is an example Genesys environment structure that includes Genesys Mobile Services:

```
{ "genesysEnvironment" : { "configServers" : [ { "backupAddress" :
"gen_backup_host",
  "backupDbid" : 105,
  "backupPort" : "8888",
  "backupServerName" : "confserv_b",
  "connectionProtocol" : "",
  "id" : "b41cc4e7-0170-41aa-93a3-9cd826f4a195",
  "localTimeout" : 0,
  "primaryAddress" : "192.168.109.142",
  "primaryDbid" : 99,
  "primaryPort" : "8888",
  "primaryServerName" : "confserv",
  "remoteTimeout" : 0,
  "tenant" : "Environment",
```

```
        "traceMode" : "CFGTMNoTraceMode"
    } ],
    "contactServers" : [ ],
    "genesysMobileServices" : [ { "backupAddress" : "",
        "backupDbid" : 0,
        "backupPort" : "",
        "backupServerName" : "",
        "connectionProtocol" : "",
        "externalBaseUrl" : "http://localhost:5009/",
        "id" : "c7e1837c-26a6-484b-9f32-a28e19cf15a0",
        "localTimeout" : 0,
        "primaryAddress" : "192.168.109.142",
        "primaryDbid" : 149,
        "primaryPort" : "5009",
        "primaryServerName" : "GMS_811",
        "remoteTimeout" : 0,
        "traceMode" : "CFGTMNoTraceMode"
    } ],
    "id" : "52417423-019c-4f77-86fb-57220a9da0f8",
    "ixnServers" : [ ],
    "statServers" : [ { "backupAddress" : "gen_backup_host",
        "backupDbid" : 129,
        "backupPort" : "2060",
        "backupServerName" : "SS_B",
        "connectionProtocol" : "",
        "id" : "61aea219-8cbd-4878-baa7-a8ed847458d2",
        "localTimeout" : 0,
        "primaryAddress" : "192.168.109.142",
        "primaryDbid" : 128,
        "primaryPort" : "2060",
        "primaryServerName" : "SS",
        "remoteTimeout" : 0,
        "traceMode" : "CFGTMNone"
    } ],
    "voiceEnvironments" : [ { "backupAddress" : "",
        "backupDbid" : 0,
        "backupPort" : "",
        "backupServerName" : "",
        "backupSipPort" : "",
        "connectionProtocol" : "",
        "id" : "370ef5e6-9e3c-4d91-9588-7f4dfe67e011",
        "localTimeout" : 0,
```

---

```
        "primaryAddress" : "192.168.109.142",
        "primaryDbid" : 113,
        "primaryPort" : "5001",
        "primaryServerName" : "SIPS",
        "primarySipPort" : "5060",
        "remoteTimeout" : 0,
        "switchDbid" : 101,
        "switchName" : "SIP_Switch",
        "switchType" : "CFG_SIP_Switch",
        "traceMode" : "CFG_TM_None"
    } ]
},
"statusCode" : 0
}
```

Please see the [GMS Deployment Guide](#) for additional details on configuring Genesys Mobile Services.

### Enabling the required feature for a contact center

The `api-mobile-push-notifications` feature must be assigned to a contact center in order for mobile push notifications to be requested for a given agent.



# Hierarchical Dispositions API

This document describes the Hierarchical Dispositions API portion of the [Genesys Web Services REST API](#).

## RESTRICTED

### Important

- This is **restricted release** documentation, and therefore is subject to change and is not complete. Some features that are described in this section might not be fully implemented in the application.

## Overview

As it is becoming necessary to support a hierarchical structure for disposition codes, the new grouping API will be used to provide this mechanism.

## Configuration

The hierarchical disposition structure in Configuration Manager will be represented as described here.

Genesys Web Services will maintain two-way synchronization between Configuration Manager configuration and Cassandra.

## API Design

The Grouping API will be used to provide access to dispositions. The following additions will be made to this API as follows:

### Disposition Group

Each disposition group will have a `dispositions` sub-resource with disposition descriptors and will contain a list of sub-groups in the list.

The group attributes for disposition groups will be updated as follows:

- For each root disposition group, the group "name" is the name of the corresponding Business Attribute. For sub-groups, the group `name` is the name of the corresponding business attribute folder.
- The value of `contains` will be `dispositions`.
- A new, optional boolean attribute `defaultGroup` will specify whether this is the default dispositions group.
- `parentUri` must be set if the group has a parent.
- The `type` attribute for this group will be `dispositions`.

## Dispositions resource

Each disposition descriptor will have the following attributes:

Field	Description	Mandatory
<code>name</code>	Name of disposition code. Used for two purposes: <ol style="list-style-type: none"> <li>1. in the settings group of disposition codes it is used as the key; and</li> <li>2. when used in <code>SetDispositionCode</code> T-Server scenario it should be used as disposition value.</li> </ol> <p>This field is immutable and can not be updated.</p>	Yes
<code>displayName</code>	Value to be displayed on UI, this field can be updated using the <code>PUT</code> operation.	Yes

## Parameters

In addition to the filters and request parameters specified in the grouping API, the following filters should be supported for disposition groups:

filter	Description
<code>subresources=*</code>	<code>subresources=*</code> returns only the first level of subresources.
<code>subresources=**</code>	<code>subresources=**</code> will return the full tree (note that <code>subresources=*</code> should be supported as well)

## Annex Configuration

All Business Attributes that contain dispositions except for the default must have an annex section in Configuration Manager

---

htcc

with one key/value pair:

```
contains:dispositions
```

This is used to determine which business attributes must be imported as disposition groups.

The `default` group is defined by updating the `interaction.disposition.value-business-attribute`. When the `interaction.disposition.value-business-attribute` option is updated in Configuration Server, the disposition group corresponding to that Business Attribute will be treated as `default`.

---

## Retrieve all configured dispositions in a specific disposition group

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

### Overview

Retrieves all dispositions in a specific disposition group.

### Example

```
GET /groups/{id}/dispositions
{
  "dispositions": [
    {
      "name": "Disp1",
      "displayName": "Dispo One"
    },
    {
      "name": "Disp2",
      "displayName": "Dispo Two"
    },
    {
      "name": "Disp3",
```

```
        "displayName":"Dispo Three"
      }
    ]
  }
}
```

---

## Retrieve full contents of a specific disposition category

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

### Overview

Retrieve the dispositions and sub-categories of any disposition group.

### Example

```
GET /groups/{id}?subresources=*
{
  "id":<id>,
  "uri":"<uri>"
  "name":"DispGroup",
  "contains":"dispositions"
  "groups":[{"name":"Sub-group", "contains":"dispositions",
"parentUri":"/groups/{parent_id}", "uri":"/groups/{id}",
"id":"{id}"}]
  "dispositions":[
    {
      "name":"Disp1",
      "displayName":"Dispo One"
    },
    {
      "name":"Disp2",
      "displayName":"Dispo Two"
    },
    {
      "name":"Disp3",
      "displayName":"Dispo Three"
    }
  ]
}
```

```
    }
  ]
}
```

---

## Retrieve full disposition and category tree

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

### Overview

Retrieve all descendants of a node in the group tree.

### Example

```
GET /groups/{id}?subresources=**
{
  "name": "Dispositions",
  "groups": [
    {
      "name": "Sub-group",
      "contains": "dispositions",
      "parentUri": "/groups/{parent_id}",
      "uri": "/groups/{id}",
      "id": "{id}",
      "groups": [{ "dispositions": [..etc..], etc..}, etc]
      "dispositions": [
        {
          "name": "Disp4",
          "displayName": "Dispo 4"
        },
        {
          "name": "Disp5",
          "displayName": "Dispo 5"
        },
        {
          "name": "Disp6",
          "displayName": "Dispo 6"
        }
      ]
    }
  ]
}
```

```
    ]
  "dispositions":[
    {
      "name":"Disp1",
      "displayName":"Dispo One"
    },
    {
      "name":"Disp2",
      "displayName":"Dispo Two"
    },
    {
      "name":"Disp3",
      "displayName":"Dispo Three"
    }
  ]
}
```

---

## Create a new disposition

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

### Overview

Create a new disposition in the appropriate group.

Each disposition must include a name and a display name.

The "name" is what the client application will send using the `SetCallDisposition` telephony operation.

### Example

```
POST /groups/{id}/dispositions
{
  "name":"MyDisposition",
  "displayName":"Display Name"
}
```

## In Configuration Manager

The corresponding business attribute value must be created in the folder that corresponds to `groups/{id}` or at root level for the corresponding business attribute if `groups/{id}` represents the root of the structure.

---

## Remove a disposition

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

### Overview

Remove a specific disposition.

A Configuration Manager Administrator must be able to remove a Business Attribute Value in any Business Attribute that is known to contain dispositions.

The removed value is immediately unavailable via the API.

### Example

Note that dispositions only live at the group level. There is no root level disposition resource:

```
DELETE /groups/{id}/dispositions/{id}
```

## In Configuration Manager

The business attribute value associated with the disposition must be removed.

---

## Create sub-category of dispositions

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

---

## Overview

Create a sub-category within another disposition group.

## Example

```
POST /groups/{id}/groups
{
  "name": "MySubGroup",
  "contains": "dispositions"
}
```

## In Configuration Manager

A folder is created within the folder represented by /groups/{id}.

---

# Create top-level disposition group

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

## Overview

Add a top-level disposition group.

## Example

```
POST /groups/{id}/groups
{
  "name": "MyGroup",
  "contains": "dispositions"
}
```

## In Configuration Manager

The associated Business Attribute must be created.

---



---

## Remove sub-category of dispositions

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

### Overview

Remove a disposition sub-category.

This should remove all child objects of the sub-category (for example: including dispositions and sub-groups).

### Example

```
DELETE /groups/{id}
```

### In Configuration Manager

The corresponding folder and all of its contents must be removed.

---

## Remove root-level disposition group

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

### Overview

Remove a root level disposition group. This should remove all child objects of the group.

### Example

```
DELETE /groups/{id}
```

### In Configuration Manager

The associated business attribute must be removed.

---

# Configuration Manager Scenarios

This operation is part of the [Hierarchical Dispositions API](#) portion of the [Genesys Web Services REST API](#).

## Overview

This describes various scenarios for which Hierarchical Dispositions can be managed with Configuration Manager.

## Create a disposition in CME

A Configuration Manager Administrator must be able to create a new Business Attribute Value in any Business Attribute that is known to contain dispositions.

This Business Attribute Value must be interpreted as a disposition in the appropriate category based on Business Attribute Value folder structure.

The new disposition should become immediately available via the API.

## Modify a disposition

A Configuration Manager Administrator must modify a Business Attribute Value in any Business Attribute that is known to contain dispositions.

The new value is reflected immediately via the API.

## Remove a disposition

A Configuration Manager Administrator must be able to remove a Business Attribute Value in any Business Attribute that is known to contain dispositions.

The removed value is immediately unavailable via the API.

## Remove a Business Attribute Value folder

A Configuration Manager Administrator must remove a Business Attribute Value folder in any Business Attribute that is known to contain dispositions.

The removed folder (for example: group in API terms) is immediately unavailable via the API.

In addition, all of its children must be removed in Cassandra as well.

---

## Remove a Business attribute representing a disposition container

A Configuration Manager Administrator must remove a business attribute which represents dispositions and the corresponding group should be removed in Cassandra as well.

Note that the default disposition container may be removed as well in which case we may have a scenario where there is no default disposition container.

## Create a new Business Attribute and mark it as a disposition container

A Configuration Manager Administrator can create a new business attribute and mark it as a disposition container by **setting the annex**.

Genesys Web Services imports this business attribute and provide access to its elements via the "groups" structure.

## Set a disposition container as "default"

When the `interaction.disposition.value-business-attribute` option is updated in Configuration Manager, Genesys Web Services must set the disposition group which corresponds to that business attribute as `default`.

### Important

- If this option is not set, the `default` business attribute name is `DispositionCode`.