Genesys Multicloud CX Web-based API Reference

Introduction to Engage Cloud APIs

5/15/2026

Contents

The Engage Cloud APIs are a collection of web APIs offered in Engage Cloud that send and receive data over HTTP in JSON (JavaScript Object Notation). You can use these APIs to create your own custom applications that integrate with Genesys.

Client libraries

One of the first things you'll need to decide is whether to use the APIs directly or use the client libraries. The libraries are designed to simplify how you interact with the API and they take care of a lot of the supporting code needed to make HTTP requests, handle HTTP responses, and enable CometD.

The libraries are available on GitHub in Java and Node.js, which means others can make bug fixes or enhancements that would then be reviewed and approved by Genesys before becoming generally available.

Genesys recommends using the client libraries if possible, but either way you should review the reference documentation for the Authentication API and any other Engage Cloud APIs you plan to use to develop your application.

Authentication

All Engage Cloud APIs implement and adhere to the OAuth 2 standard for secure authentication. See the Authentication Overview for details.

CometD

Many requests in the Engage Cloud APIs are asynchronous. When you send an asynchronous request, the API returns an HTTP response with a status code, but this only means the request was processed and sent to a backend Genesys server. When the server finishes processing the request and notifies the API of any changes in state or errors, the API then sends the updated state or error details to the client application as an unsolicited notification.

The Engage Cloud APIs uses CometD to deliver these 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. For more information about CometD, or for details about where to obtain client-side CometD libraries for various platforms, see the official CometD site.

These APIs require CometD to deliver notifications:

- [[PEC-Developer/Current/WebAPIs/Provisioning_API|]]
- Statistics Overview
- [[PEC-Developer/Current/WebAPIs/Workspace_API|]]

Click the links above and review the “CometD” section on the API’s landing page for information about which channels you can subscribe to and the notification you’ll receive.

When an API requires CometD, you’ll see a “Notifications” category in its API reference information (see the Statistics API Notifications page for an example). You can use these endpoints to implement CometD if you aren’t using a client-side CometD library.

Requests

The Engage Cloud APIs use the following standard HTTP verbs to perform actions on resources:

Verb	Description
GET	Retrieve resources.
POST	Create resources.
PUT	Update resources.
DELETE	Delete resources.

Requests support typical parameter types like body, header, path, and query, but there is also a special body parameter called **data**. This is a JSON object containing any information needed to execute the action. For example, if you POST to `/workspace/v3/voice/make-call`, you must include a data object with the call destination:

```
{
    "data":{
       "destination":"5002"
    }
 }
```

Responses and errors

All Engage Cloud APIs return a **status** JSON object with a required **code** field. You might also see the **message** and **detail** fields, which provide more information about the response. For example:

```
{
    "status":{
       "code":0
    }
 }
```

For successful responses, you can expect to see the following codes:

Code	Description
0	The synchronous operation was successful.
1	The asynchronous operation was sent successfully. You'll find out about the state of the operation through CometD notifications. For example, if you POST to `/workspace/v3/initialize-workspace`, the Workspace API immediately returns a response with a status code of 1 and later follows up with the WorkspaceInitializationComplete event, delivered via CometD.
2	The synchronous operation was partially successful. This is returned if at least one action succeeds in a request that performs a bulk operation. In this case, the response includes the **status** object with a **code** of 2, as well as **data** and **errors**. The **data** object includes any resources that were successfully retrieved as part of the bulk operation. The **errors** object is an array of **status** objects for each of the operations that failed in the request.

All other status codes indicate an error and include a **message**, and possibly a **detail** field, with more information about the problem. For example:

```
{
    "status":{
       "code":500,
       "message":"Resource not found",
       "detail":{
          "ConfCode":12345
       }
    }
 }
```

For asynchronous requests, you might also see errors delivered via CometD notifications. For example, the Workspace API returns an EventError message that includes the related DN, along with the error code and message, and the call connection ID and call UUID, if available.