API Reference

Chat API

5/3/2025

Contents

- 1 Chat API
  - 1.1 Advanced Usage of the Chat API
  - 1.2 Accessing the Chat Service API of the Built-In Chat Widget

Chat API

Important

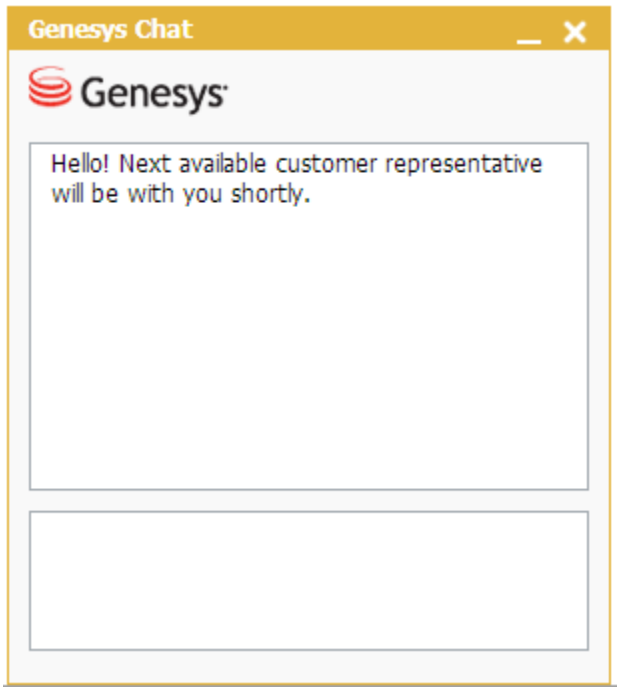
Starting with the 8.5.100.11 release of Genesys Co-browse, Genesys is deprecating the Built-in Chat Widget and its APIs in preparation for discontinuing support in the upcoming 9.0 release.

This functionality is now available through a single set of consumer-facing digital channel APIs that are part of Genesys Mobile Services (GMS), and through Genesys Widgets, a set of productized widgets that are optimized for use with desktop and mobile web clients, and which are based on the GMS APIs.

Genesys Widgets provide for an easy integration with Co-browse, allowing you to proactively serve these widgets to your web-based customers.

Although the deprecated APIs and Built-in Chat Widget will be supported for the life of the 8.5 release of Co-browse, Genesys recommends that you move as soon as you can to the new APIs and to Genesys Widgets to ensure that your functionality is not affected when you migrate to the 9.0 release.

Co-browse is shipped with a built-in chat widget. Out-of-the-box, the chat widget looks like this:



To configure the chat widget, see Configuration API.

To get access to the Chat Widget API, see Accessing the Co-browse and Chat APIs. You generally will not need to access the Chat Widget API as configuration can be done in instrumentation. The Chat Widget API can be used to get access to the lower lever Chat API. See Advanced Usage below for more details.

For a full Chat Widget API reference, see Chat Widget JS API.

Advanced Usage of the Chat API

The Chat Widget API is built on top of the Chat Service JS API. The Chat Service API can be used to send chat commands to the server, for example, *send a message* or *leave session*. The Chat Service API also lets you subscribe to session events such as `agentConnected` and `messageReceived`.

Getting Access to the Chat Service API

There are two ways to get access to the Chat Service API:

- Accessing the Chat Service API of the Built-In Chat Widget
- Use the JavaScript Bundle

Accessing the Chat Service API of the Built-In Chat Widget

The following code example shows how you can access the Chat Service API. Note that this example is a bit simplistic in that it starts chat unconditionally on every page load and does not handle errors.

```
var _genesys = {
    chat: {
        // 1. Tell Co-browse JS not to call restoreChat(),
        //    because you will call it manually.
        autoRestore: false,
        // 2. Subscribe to chat widget's "ready" event
        //    to get access to widget API.
        onReady: function(chat) {
            // 3. Use chat widget API to get access to service API.
            chat.onSession(function(session) {
                // Use chat service API here. For example,
                // session.sendMessage('Hello World!');
            })
        }
    }
}
```

Accessing the Chat Service API using the JavaScript Bundle

You can use the JavaScript bundle shipped with Co-browse to access the Chat Service JS API. This file is available at the following URL:

`http(s)://<COBROWSE_SERVER>/cobrowse/js/chatAPI.min.js`

When loaded in a browser, this file exports the Chat Service JS API as a global `chat` variable. The size of this file is 113 KB (~35 KB gzipped) and it does not require any dependencies.

Another version of this file is available at `http(s)://<COBROWSE_SERVER>/cobrowse/js/chatAPI-noDeps.min.js`. The size of this file is 23 KB (~8 KB gzipped), but it requires that you have the following libraries globally available:

- `$` for jQuery (v. 1.8.1 or higher)
- `_` for underscore (v. 1.5.0 or higher) or lodash (v. 2.0.0 or higher)
- `org.cometd` for Cometd (v. 2.8.0)

Important

If you choose to implement your own chat widget using the Chat Service JS API in the form of a seperate JS file, your chat widget will *not* be automatically integrated with Co-browse. Integration consists of two features:

- Co-browse automatically determines if the user is on chat when the user starts a Co-browse session.
- The Co-browse session token is automatically passed to an agent.

To support these integration features, you will also have to implement the External Media Adapter API for your chat widget and pass the implementation object to the Configuration API `primaryMedia` option.