Widgets Developer's Guide

12/5/2024

Table of Contents

Contents	
Genesys Widgets deployment guide	5
Supported browsers	13
Cookies	15
Configure widgets and services	21
Localize widgets and services	25
Customize appearance	30
Rich Media	43
Accessibility	46

Contents

- 1 Related resources

The Genesys Widgets Developer's Guide covers how to get started with Widgets.

Related documentation:

-

This manual includes the following topics:

- Configure widgets and services
- Customize appearance
- Supported browsers
- Cookies
- Localize widgets and services
- Rich Media
- Accessibility

Related resources

- How Genesys Widgets works
- Widgets API Reference

Genesys Widgets deployment guide

Contents

- 1 Audience
- 2 How can I deploy Genesys Widgets?
- 3 Deploying Genesys Widgets (lazy loading)
  - 3.1 Files used
  - 3.2 On-demand lazy loading
  - 3.3 Preloading plugins
- 4 Deploying Genesys Widgets (all-in-one)
  - 4.1 Files Used
- 5 Alternative deployment script
- 6 Releases hosted on Content Delivery Network (CDN)
  - 6.1
  - 6.2
  - 6.3 Choose region
  - 6.4 Using Genesys Widgets CDN with versions
  - 6.5 Versioning examples with scenarios
  - 6.6 Deployment methods
- 7 Checking Widgets version
- 8 Genesys Web Fonts

- Developer

This guide provides the steps required to instrument your website with Genesys Widgets.

Related documentation:

-

Audience

This document is for website developers who are in charge of website code. You must have knowledge of HTML, JavaScript, and CSS.

How can I deploy Genesys Widgets?

There are two deployment methods available:

- Lazy loading— The recommended method, lazy loading breaks the JavaScript bundle apart into individual plugin files and loads them into the page only as you need them. This is the preferred method of deploying Genesys Widgets.
- All-in-one (deprecated) —The all-in-one method is the "classic" method of deploying Genesys Widgets. In this method, you have one JavaScript file and one CSS file that contain all plugins and resources.

Deploying Genesys Widgets (lazy loading)

> **Important**
> Lazy loading is the recommended method for Widgets. The (all-in-one) method is deprecated.

Files used

- **widgets/cxbus.min.js**
- **widgets/plugins/widgets-core.min.js**
- **widgets/plugins/*.***

A good starting point is the following script:

This script does the following:

1. Loads **cxbus.min.js**. This makes the global CXBus instance available.
2. Configures CXBus to turn on debug logging and set the path to the Widgets plugin folder.
3. Loads your configuration file, **widgets.config.js**. (This is an imaginary file. You must create it).
4. Loads **widgets-core**, the core Genesys Widgets library.

Use this script as a starting point and customize it as needed.

Remember that your configuration can be defined inline on the page or loaded in as a separate file (as shown in this script).

> **Important**
>
> Whichever method you choose, you must ensure your configuration is in the page before you load **widgets-core**. Otherwise, **widgets-core** cannot read the configuration.

> **Important**
>
> Refer to Configuring Genesys Widgets for help.

On-demand lazy loading

Genesys Widgets is designed to load plugins into the page on-demand as you use the product. For example, if you call the command **WebChat.open**, CXBus fetches the **webchat.min.js** plugin from the **plugins/** folder and loads it into the page. Any WebChat command triggers it to load. Likewise, WebChat calls WebChatService commands, thus CXBus loads **webchatservice.min.js** into the page as well.

Preloading plugins

In some cases, you might not want to load plugins on-demand, or the demand is to load them at startup. A good example is SideBar. You probably want this plugin to appear on the screen immediately so the customer can use it. To make this possible, you can specify which plugins you want to preload at startup in your configuration.

```
_genesys.widgets.main.preload = [

    "sidebar"
];
```

You may specify as many plugins as you want in the preload list. The plugins load in order after you

load Widgets Core.

All plugin names are lower-case. Please refer to the file names in the **plugins/** folder. For example, to preload **webchat.min.js**, specify `webchat`, the first part of the file name.

You may find other plugins or features of plugins that necessitate preloading.

Deploying Genesys Widgets (all-in-one)

> ### Important
> Lazy loading is the recommended method for Widgets. The (all-in-one) method is deprecated.

Files Used

- **widgets/widgets.min.css**
- **widgets/widgets.min.js**

A good starting point is the following script:

First, you must define your configuration for Genesys Widgets. You can do this inline on the page by using a *script* tag, or you can store it in a separate file and load it in before **widgets.min.js**. In the script example above, we assume your configuration is stored in another file. You must create the **widgets.config.js** file for this script to function properly.

> ### Important
> Whichever method you choose, you must ensure your configuration is in the page before you load **widgets.min.js**. Otherwise, **widgets.min.js** cannot read the configuration.

> ### Important
> Refer to Configuring Genesys Widgets for help.

Alternative deployment script

To simplify the deployment process while using tools like Google Tag Manager, you can use the below

script to embed Widgets.

Releases hosted on Content Delivery Network (CDN)

Genesys Widgets is now available over CDN, providing optimized load times and instant access to new releases.

```
https://apps.mypurecloud.com/widgets//
```

Note that and are placeholders.

This value varies based on the deployment method you choose.

> **Tip**
> In the case where a CDN URL that you are trying to access is **not found**, it means that either the release or the file you are looking for is not yet available.

version can take one of the following 3 values.

- 9.0 - (Major) - A version that is company-wide or
- 9.0.xxx - (Major).(Minor) - Minor version is product specific and is tied to each widget's iteration or
- 9.0.xxx.xx - (Major).(Minor).(Release candidate) - Specific release version

For all the available released versions, refer to the Widgets Release Notes.

Choose region

Widgets is available in a number of regions worldwide, as shown below. Choose the nearest or appropriate region URL based on where you are located.

Region	URL
North America (East)	https://apps.mypurecloud.com/widgets/{version}/{path/to/file}
North America (West)	https://apps.usw2.pure.cloud/widgets/{version}/{path/to/file}
North America (Canada)	https://apps.cac1.pure.cloud/widgets/{version}/{path/to/file}
Australia or New Zealand	https://apps.mypurecloud.com.au/widgets/{version}/{path/to/file}
EU (Ireland)	https://apps.mypurecloud.ie/widgets/{version}/{path/to/file}
EU (Frankfurt)	https://apps.mypurecloud.de/widgets/{version}/{path/to/file}

Region	URL
UK (London)	https://apps.euw2.pure.cloud/widgets/{version}/{path/to/file}
Japan	https://apps.mypurecloud.jp/widgets/{version}/{path/to/file}
Mumbai	https://apps.aps1.pure.cloud/widgets/{version}/{path/to/file}
Seoul	https://apps.apne2.pure.cloud/widgets/{version}/{path/to/file}

Using Genesys Widgets CDN with versions

Starting in the 9.0.006.02 release, all the released versions are accessible from the Genesys CDN URL. The sections below explain how to access the latest available released version or a specific released version using Genesys CDN.

To get the latest released version under the 9.0 family: https://apps.mypurecloud.com/widgets/9.0/widgets.min.js

To get the last available released version under a specific (Major).(Minor) version (this also includes any hot fixes for that release): https://apps.mypurecloud.com/widgets/**9.0.xxx**/widgets.min.js

Example: https://apps.mypurecloud.com/widgets/**9.0.006**/widgets.min.js

To get a specific release/hot-fix version: https://apps.mypurecloud.com/widgets/**9.0.xxx.xx**/widgets.min.js

Example: https://apps.mypurecloud.com/widgets/**9.0.006.02**/widgets.min.js

> ### Important
> Note that all older versions of Genesys Widgets may not be available in the CDN. All the released versions are available only starting with the 9.0.006.02 version.

Versioning examples with scenarios

When a new release version comes out, it is available under all the 3 different CDN URLs below. In this example, if 9.0.006.01 is the first ever release announced, then it is available under the following CDN URLs.

- /9.0/
- /9.0.006/
- /9.0.006.01/

When 9.0.007.04 is released, it is available under /9.0/ but **not** under /9.0.006/ or /9.0.006.01/. Instead, /9.0.007/ and /9.0.007.04/ CDN URLs are created and this release is available under them:

- /9.0/
- /9.0.007/
- /9.0.007.04/

If a hot fix (such as 9.0.006.02) is released after 9.0.007.04 is released, then the hot fix is available under the following CDN URLs:

- /9.0.006/
- /9.0.006.02/

If a hot fix (such as 9.0.007.05) is released before announcing any new release, then it is available under the following CDN URLs:

- /9.0/
- /9.0.007/
- /9.0.007.05/

Deployment methods

Lazy loading

Recommended approach: When using the lazy loading method, the base Genesys CDN URL must be prefixed in the lazy loading deployment script. The value is not needed in this scenario because they are auto-loaded from the base CDN configured. Here is what the deployment script looks like when using 9.0.006.02 release:

All-in-one

Legacy approach (deprecated): When using the all-in-one deployment method, the values are the files mentioned in the all-in-one section. For example, if you would like to use widgets.min.js and widgets.min.css under 9.0.006.02 release, CDN URLs will look like this:

```
https://apps.mypurecloud.com/widgets/9.0.006.02/widgets.min.js
https://apps.mypurecloud.com/widgets/9.0.006.02/widgets.min.css
```

Checking Widgets version

CXBus.command("App.info");

Prints out the debug header with version information.

window._genesys.widgets.common.data("version");

Returns the version number directly, as a string.

Genesys Web Fonts

Google Fonts are now hosted in Genesys Infrastructure. Please Choose the nearest or appropriate region URL based on where you are located and configure it through the googleFontUrl option.

> **Important**
>
> By default, Genesys web fonts are loaded from the North America (East) region.

Region	URL
North America (East)	https://apps.mypurecloud.com/webfonts/roboto.css
North America (West)	https://apps.usw2.pure.cloud/webfonts/roboto.css
North America (Canada)	https://apps.cac1.pure.cloud/webfonts/roboto.css
Australia or New Zealand	https://apps.mypurecloud.com.au/webfonts/roboto.css
EU (Ireland)	https://apps.mypurecloud.ie/webfonts/roboto.css
EU (Frankfurt)	https://apps.mypurecloud.de/webfonts/roboto.css
UK (London)	https://apps.euw2.pure.cloud/webfonts/roboto.css
Japan	https://apps.mypurecloud.jp/webfonts/roboto.css
Mumbai	https://apps.aps1.pure.cloud/webfonts/roboto.css
Seoul	https://apps.apne2.pure.cloud/webfonts/roboto.css

Supported browsers

Contents

- 1 Desktop browsers
  - 1.1 Windows
  - 1.2 Mac OS
- 2 Mobile Browsers

- Administrator
- Developer

Genesys has tested the following desktop and mobile browsers.

Related documentation:

-

> **Important**
> Support for the device/OS/browser combinations listed below will only be available for as long as Genesys can properly reproduce the issue. Please report any issues you encounter with any of our tested browsers.

Desktop browsers

Windows

- Google Chrome — Current release or one version previous
- Microsoft Edge — Current release or one version previous
- Mozilla Firefox — Current release or one version previous

Mac OS

- Google Chrome — Current release or one version previous
- Microsoft Edge — Current release or one version previous
- Mozilla Firefox — Current release or one version previous
- Safari — Current release or one version previous

Mobile Browsers

- Google Chrome — Current release or one version previous
- Safari — Current release or one version previous

Cookies

Contents

- 1 Overview
  - 1.1 Purpose
  - 1.2 Cookie creation
  - 1.3 Duration
  - 1.4 Sub-domains
  - 1.5 Cookie support in test environments
- 2 App
- 3 Console
- 4 WebChat
- 5 Local storage

- Administrator
- Developer

Learn which session cookies are used by Genesys Widgets to restore chat sessions, track the state of the UI, store a customer's decisions, and more.

Related documentation:

-

Overview

Purpose

Genesys Widgets uses cookies to store non-sensitive data in the browser. The end-user's browser must allow cookies for Genesys Widgets to operate properly. Each cookie is required, and without the ability to read and write these cookies, Genesys Widgets features will not function properly.

Cookie creation

All cookies start with the prefix "**_genesys.widgets**" to easily identify them. By default, Genesys Widgets cookies are created in a way that allows the cookies to be read across sub-domains by setting the "domain" attribute in the cookie options. We derive the proper domain value by parsing the host site's domain and extracting it.

> **Important**
> Genesys Widgets never stores Personally Identifiable Information (PII) in its cookies.

Duration

All cookies used by Genesys Widgets are created as session cookies and will be deleted when the user's browser is fully closed.

Sub-domains

Normally, cookies cannot be transferred between sub-domains of a website unless they are configured to do so. Genesys Widgets automatically detects the domain of the host site and configures all cookies to be transferable between sub-domains. For example, you could start a chat on **www.testsite.com** and restore that chat session on **store.testsite.com**,

support.testsite.com, or **portal.testsite.com**.

Cookie support in test environments

Genesys Widgets uses special cookies that persist across sub-domains. This is a critical feature for plugins like WebChat that need to restore an active chat session while navigating around a website. The side effect of using this type of cookie is they won't work when using test environment domain names such as "localhost" or an IP address. You must use a fully-qualified domain name (FQDN) such as "localhost.com" or any other variant that can be identified as a domain name. Cookies will also fail to work if you run the test site as an HTML file path directly in the browser.

One workaround is to update your system's **hosts** file to create an FQDN alias for "localhost", your test environment's name, or an IP address.

Example

```
127.0.0.1       localhost
127.0.0.1       localhost.com
```

A fully-qualified domain name (FQDN) such as "localhost.com" or any other variant that can be identified as a domain name is not mandatory, but it is recommended. This way, the cookies will also work when using test environment domain names such as "localhost" or an IP address.

The following is a list of cookies used by Genesys Widgets.

App

Cookie Name	Purpose
_genesys.widgets.app.autoLoadList	Contains a list of active plugin names that are updated based on the usage of widgets during the lazy loading deployment method. This is to ensure that a widget is auto-loaded during a page refresh or page navigation when there is an active session associated with it.

Console

Cookie Name	Purpose
_genesys.widgets.console.session	Contains the active Console plugin open/close state.
_genesys.widgets.console.commandPlugin	Contains the selected plugin name from the Commands section.

Cookie Name	Purpose
_genesys.widgets.console.command	Contains the selected command to run from the Commands section.
_genesys.widgets.console.eventPlugin	Contains the selected plugin from the Events section to listen for events.
_genesys.widgets.console.event	Contains the selected event type to listen against, from the Events section.
_genesys.widgets.console.optionsArea	Contains the command options to send when executing a command.
_genesys.widgets.console.activeSubscriptions	Contains the list of all active event subscriptions listening via the Console plugin.
_genesys.widgets.console.windowPosition	Contains the position of the Console plugin on the screen.

WebChat

Cookie Name	Purpose
_genesys.widgets.webchat.state.open	Contains the WebChat Widget open or close state for internal tracking purposes.
_genesys.widgets.webchat.state.keys	Can contain encrypted keys related to the current active chat session.
_genesys.widgets.webchat.state.ping	Contains the time at which the last successful request was made to the server.
_genesys.widgets.webchat.metaData	Contains all the Metadata details related to the current active chat session.
_genesys.widgets.webchat.state.index	Contains the last unique Message ID for internal tracking purposes.
_genesys.widgets.webchat.state.filters	Contains any prefilters that were added using WebChatService plugin commands addPrefilter or sendFilteredMessage.
_genesys.widgets.webchat.state.session	Contains the unique Session ID related to the current active chat session. It is used to restore the active chat session during scenarios like page refresh or page navigation.
_genesys.widgets.webchat.state.minimized	Contains the WebChat Widget minimized or maximized state for internal tracking purposes.
_genesys.widgets.webchat.autoInvite.disabled	Contains a value that disables or enables the WebChat autoInvite feature. It is dynamically updated based on the user's response to the initial WebChat invite.
_genesys.widgets.webchat.state.unreadMessages	Tracks the number of unread messages during an active chat session, when WebChat is minimized. It is cleared whenever the WebChat Widget is

Cookie Name	Purpose
	maximized by the user to read the new messages.
_genesys.widgets.webchat.state.lastMessageCountRead	Contains the number of messages that are read during an active chat session that calculates the number of unread messages when WebChat is minimized. It is automatically cleared whenever the WebChat Widget is maximized or closed/ended.
_genesys.widgets.webchat.state.asyncUnreadMessageCount	Keeps track of the number of unread messages related to an Async Chat, when WebChat is minimized. It is cleared whenever the WebChat Widget is maximized by the user to read the new messages.
_genesys.widgets.webchat.state.pureengage-v3-rest.session	Used only with Genesys Engage V3 API. It contains the Session ID related to the current active chat session. It is used to restore the active chat session during scenarios like page refresh or page navigation.
_genesys.widgets.webchat.state.pureengage-v3-rest.keys	Used only with Genesys Engage V3 API, containing the encrypted keys related to the current active chat session.
_genesys.widgets.webchat.state.pureengage-v3-rest.index	Used only with Genesys Engage V3 API, containing the last unique message ID for internal tracking purposes.
_genesys.widgets.webchat.state.pureengage-v3-rest.open	Used only with Genesys Engage V3 API, containing the WebChat Widget open or close state for internal tracking purposes.
_genesys.widgets.webchat.state.purecloud-v2-sockets.JWtoken	Used only with Genesys Cloud CX V2 API, containing the JWT token related to the current active chat session.
_genesys.widgets.webchat.state.purecloud-v2-sockets.ConversationID	Used only with Genesys Cloud CX V2 API, containing the active conversation ID related to the current chat session.
_genesys.widgets.webchat.state.purecloud-v2-sockets.MemberID	Used only with Genesys Cloud CX V2 API, containing the user ID of the WebChat Widget related to the current active chat session.
_genesys.widgets.webchat.state.purecloud-v2-sockets.WS_URL	Used only with Genesys Cloud CX V2 API, containing the WebSocket event stream URI for listening to new incoming messages.
_genesys.widgets.webchat.state.purecloud-v2-sockets.LastMsgId	Used only with Genesys Cloud CX V2 API, containing the last unique ID of the message sent in the WebChat Widget.

Local storage

Genesys Widgets uses local storage to store non-sensitive data in the the browser with no expiration date.

Key name	Purpose
_genesys.widgets.inFocus	A globally unique identifier (GUID) to identify the current active chat session browser tab/window, when the WebChat Widget opens in multiple browser tabs/windows.

Configure widgets and services

Contents

- 1 Main Configuration
- 2 Widget Configuration Options
- 3 Launcher
  - 3.1 How to use Launcher

- Developer

Learn how to configure widgets and services.

Related documentation:

-

Depending on your product, you can use Genesys Widgets for functionality such as WebChat or Callback. You can configure all widgets and services in the same configuration object. When you add new Genesys products and services, you must update your Genesys Widgets configuration to enable those widgets.

After you deploy Genesys Widgets on your website, configure the CX Widget by defining the **global window._genesys** Javascript object.

To include the JavaScript script, you can choose one of the following options:

- Place the script inline on your website
- Place it in a separate JavaScript file, and include the file on your page

The following example is a basic view of the global Genesys Widgets configuration object:

```
// include widgets.min.js after defining your configuration options
```

The following example is a populated Widget configuration that includes configuration options for WebChat.

> **Important**
>
> You must define your configuration options on the page **before** widgets.min.js is loaded.

Main Configuration

Genesys Widgets is a hub for multiple Genesys products and services. Some configuration options are set globally and therefore apply to all products and services running on the CX Widget platform. In the main application configuration you can configure options such as visual theme, language, and mobile support.

For detailed information on configuration options, see app configuration options.

Widget Configuration Options

- WebChat configuation
- WebChatService configuration
- ChannelSelector configuration

For a complete catalogue of Widgets configuration options, see Genesys Widgets API Reference.

Launcher

Launcher is a sample page that shows how Genesys Widgets are displayed on any host website. Use this page to:

- View Genesys Widgets with your own configuration.
- Copy the Configuration Script; for example, using the details you entered on the form, the configuration script is generated in the **Need Configuration Script** section. You can copy this script and use it in your website to launch Widgets.
- Starting in the 9.0.008.03 version, use the Launcher tool to test and configure between the different API services available in Genesys, namely under Genesys Multicloud CX, GenesysEngage-cloud and Genesys Cloud CX.

How to use Launcher

Sidebar
To enable the Sidebar plugin, select this check box in Launcher. By default, Sidebar will be shown on the right side of the screen on Widgets startup. You can configure SideBar by using the options shown under this section. Provide the sidebar channel configuration in the corresponding text area according to the Sidebar documentation or use the provided sample configuration links to pre-fill the sample data in the text area. Ensure that the channels defined here are enabled and configured as well.

Enable Live Assist (EWT)
Select the check box next to **Enable Live Assist (EWT)** to enable the ChannelSelector plugin. Enter the Stats URL followed by the virtual queue names to fetch the Estimated Wait Time details. Refer to the Channel Selector documentation for more information. To show Chat, SendMessage and CallUs channels in this plugin, please make sure that you select these plugins in this Launcher page.

WebChat
Select the **WebChat** check box to enable WebChat. You must enter a URL. Other values are optional and self-explanatory.

> Important

> Starting in the 9.0.008.03 version, WebChat supports Genesys Multicloud CX v3 API via the transports configuration section. You can test this in the Launcher tool using the WebChat [with Transport only] section under the GenesysEngage-cloud tab.

CallUs
Select the check box next to **Call Us** and provide the configuration data. Note that you can select the **Edit/Use Sample Config** option to use sample configuration data, which you can edit it with your own detail data. Ideally, Call Us is shown in the Live Assist widget. It can also be launched with the `CallUs.open` bus command.

Callback
Select the check box next to **Callback** to include the Callback and Calendar plugins. Enter the callback service provider URL field and other details as required. Ensure **Enable Sidebar with Live Assist** is selected.

Lazy Loading
Select the check box next to **Enable Lazy loading** to launch Widgets in the lazy-load mode. Otherwise they will be launched on startup. At the minimum, Sidebar plugin must be enabled and configured with the required channels to load it on Widgets startup.

Once you have entered all of the necessary configuration details, click the **Launch** button to launch Widgets.

Localize widgets and services

Contents

- 1 Master localization file
- 2 Multiple translated language packs
  - 2.1 Example
- 3 Configuration options
- 4 Language pack JSON format
- 5 Localization namespaces
- 6 Language codes
- 7 Plugin localization options

- Developer

Localize your Genesys Widgets user messages and prompts by creating and hosting a Language Pack that Genesys Widgets can access.

Related documentation:

-

The Language Pack is a special file written in JSON format.

You also have to specify your Language Pack file in the **window_genesys.widgets.main** section of your Genesys Widgets configuration options, as shown in this example:

Master localization file

The widgets-en.i18n.json file provides the latest i18n localization content containing all the language codes and strings of all Widgets. This acts as a centralized master file that you can use as a reference to create your own modified localization file and host it. In this way, you can use this to override the language content.

> **Important**
> The English language pack file provided in the above URL is just for reference. Do not load this file into Widgets because it is already built into Widgets by default.

Multiple translated language packs

Multiple i18n language pack files are available as individual JSON files in the /i18n folder. You can select the desired language pack file and then set the `i18n` and `lang` properties in the **window._genesys.widgets.main** configuration option. Each language pack file is named using the language code to identify easily. The same language code is also used inside the language pack file to construct the i18n JSON. This language code must be specified in the **main.lang** configuration option.

Example

The French language pack file is available as **widgets-fr.i18n.json**. To use this language pack file, follow this example:

```
window._genesys.widgets = {
```

```
    main: {

        lang: "fr",
        i18n: "/relative/path/to/i18n/widgets-fr.i18n.json"

        // OR using the CDN URL
        i18n: "https://apps.mypurecloud.com/widgets//i18n/widgets-fr.i18n.json"
    }
};
```

Language code mapping examples:

Language	Code
Brazilian Portuguese	pt-BR
Chinese Simplified	zh-CN
Chinese Traditional	zh-TW
Danish	da
Dutch	nl
English	en
Finnish	fi
French	fr
German	de
Italian	it
Japanese	ja
Korean	ko
Norwegian	no
Polish	pl
Spanish	es
Swedish	sv
Thai	th
Turkish	tr

> **Important**
> You may use any language code you wish. The above table is for reference only.

Configuration options

main.lang
Type: string
Default: "en"

Requirement: Optional
Description: A language code to specify which language to display in the Widgets. Language codes are set by the customer.

main.i18n (external file)
Type: string
Default: built-in English words and phrases
Requirement: Required when using main.lang option.
Description: A URL that the Widgets use to fetch the Language Pack file upon startup.Can be partial or complete. Unspecified strings will use default values.

main.i18n (inline object)
Type: object
Default: built-in English words and phrases
Requirement: Required when using main.lang option.
Description: An inline JSON object. Can be partial or complete. Unspecified strings will use default values.

Language pack JSON format

The language pack is written in JSON format.

```
// Root
{
    // Language Code
    "en": {

        // Widget name
        "webchat": {

            // Localized strings
            "ChatStarted": "Chat Started",
            "ChatEnded": "Chat Ended",
            "ChatFailed": "There was a problem starting the chat session. Please Retry.",

            // Customer Defined Strings - Match & Replace messages received from chat server
            "SYS0001": "An Agent will be with you shortly"
        },

        "sendmessage": {

              // Localized strings
             "SendMessageButton": "Send Message",
             "EmailFormFirstname": "First Name",
             "EmailFormLastname": "Last Name",

             //Errors
             "ErrorServerNotAvailable": "Unable to reach server. Please try again.",
             "ErrorAttachfileSizeMax": "Total size of attachments exceeds limit: "
        }
    }
}
```

Localization namespaces

Plugin	Namespace
Calendar	calendar
CallBack	callback
CallUs	callus
ChannelSelector	channelselector
Offers	offers
WebChat	webchat

Language codes

To allow flexibility in the way your website handles multiple languages and language codes, there are no rules for language codes other than that they must be strings. This means that you can use any language code system.

However, the language code that you set in **window._genesys.widgets.main.lang** must correspond to a language code in the Language Pack File.

> ### Important
> When using one of the available pre-translated language packs, ensure the language code maps with the one included in the language pack file.

Plugin localization options

- ChannelSelector
- CallUs
- Calendar
- Callback
- SideBar
- WebChat

Customize appearance

Contents

- 1 Set the active theme
- 2 Create a custom theme
  - 2.1 Theme templates
  - 2.2 Name a theme
  - 2.3 Customization guidelines
- 3 Register a theme with Genesys Widgets
- 4 Change the appearance of a specific widget
- 5 Change the layout and structure of a widget
- 6 Change fonts
  - 6.1 Disable Roboto font download
  - 6.2 Change font size
- 7 Icons
  - 7.1 How to use icons
  - 7.2 Multi-tone icon set
  - 7.3 Outline icon set

- Developer

Use *themes* to change the appearance of Genesys Widgets. Themes allow you to apply colors and fonts to all of your widgets in a single operation.

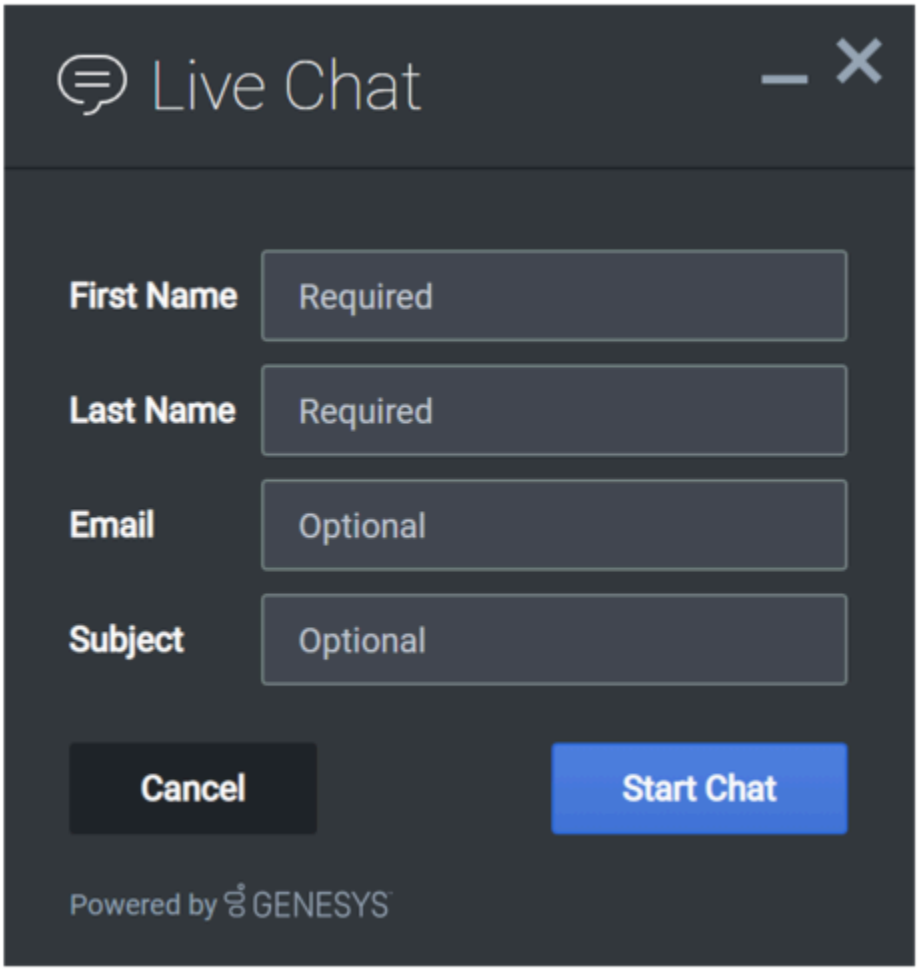
Related documentation:

-

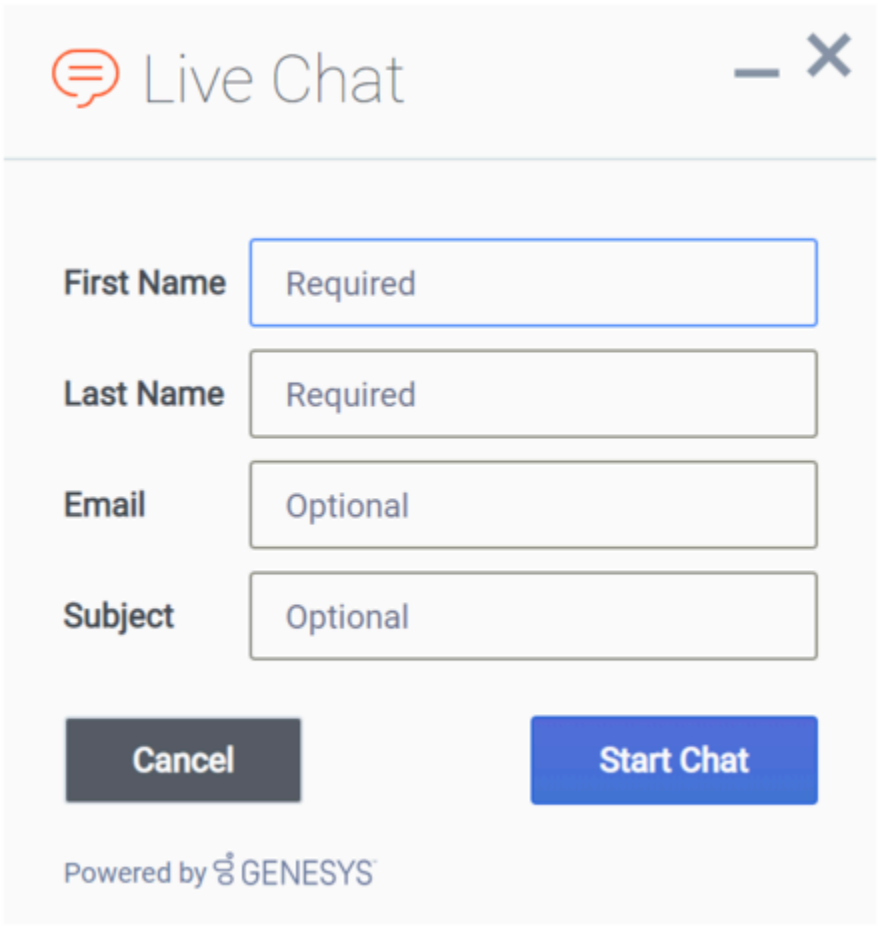
Genesys Widgets includes two built-in themes: **dark** and **light**.

Note: The dark theme is active by default.

Dark theme



Light theme



Set the active theme

There are two ways to set the active theme:

Configuration

```
window._genesys.widgets.main.theme = "light"; // or "dark"
```

Widget Bus Command

```
window._genesys.widgets.bus.command("App.setTheme", {theme: "light"}); // or "dark"
```

Create a custom theme

Theme templates

Genesys Widgets uses special LESS files called *theme templates* to define themes. Use a theme template to create a new color palette and add custom styles. Everything is laid out clearly in the template file.

LESS syntax defines local variables that allow you to create a clear color palette consisting of no less than 28 separate color variables, which are grouped by their usage:

- Background colors
- Text colors
- Icon colors
- Border colors
- Outline colors

At a bare minimum, you can create a new style by simply changing the color values in the color palette. You can also add or remove colors from this palette.

Color palette example

```
/* Color Palette */

@bg_color_1:                #33383D; // Main Background Color
@bg_color_2:                #444A52; // Form Inputs
@bg_color_3:                #222529; // Button default
@bg_color_4:               #5081E1; // Button primary gradient 1
@bg_color_5:               #4375D6; // Button primary gradient 2
@bg_color_6:               #CCCCCC; // Button disabled / scrollbar color
@bg_color_7:               #212529; // Native scrollbar track color
@bg_color_8:               #A3A8AE; // Scrollbar color

@txt_color_1:                #FDFDFD; // Main text color
@txt_color_2:                #98A7B8; // footer text
@txt_color_3:                #FDFDFD; // Button default & primary / autocomplete text hover
color
@txt_color_4:                #FDFDFD; // Hyperlink color
@txt_color_5:               #C5CCD6; // Placeholder color
@txt_color_6:               #F53131; // Alert/error color

@icon_color_1:               #FDFDFD; // Base icon color
@icon_color_2:               #8C8C8C; // Secondary icon color (multitone only)
@icon_color_3:               #000000; // Icon shadow color (multitone only)
@icon_color_4:               #000000; // Icon secondary shadow color (multitone only)
@icon_color_5:               #98A7B8; // Window control icon color
@icon_color_6:               #98A7B8; // Form input icon overlay color (e.g. "clear" icon)
@icon_color_7:                #5081E1; // Interactive icon color 1 (attach files, delete
file, etc)
@icon_color_8:                #4AC764; // Positive Color (confirmation, availability,
usually green)
@icon_color_9:                #F53131; // Negative Color (error, exception, usually red)
@icon_color_10:         #F8A740; // Warning Color (warning, pending, offline, usually yellow
or orange)
@icon_color_11:         #FDFDFD; // Icon color for primary buttons

@border_color_1:         #222529; // Main border color
@border_color_2:        #2E69DB; // Button primary
@border_color_3:        transparent; // Button default
```

```
@border_color_4:        transparent; // Button disabled
@border_color_5:        #F53131; // Alert/error color
@border_color_6:         #758384; // Form controls default state

@outline_color_1:         #75A8FF; // Form input focus outline / autocomplete hover
background color
@outline_color_2:         #DAE6FC; // Outline color for primary buttons
```

Sample theme template files

Click the following links to automatically download the sample template files:

theme-template-dark.less

theme-template-light.less

> **Important**
>
> Theme templates are LESS files, which must be converted to CSS before being used on a website. Use a website or tool to convert them when you're ready to test and implement them on your site.

> **Important**
>
> By default, theme templates override the styles of all of your Genesys Widgets—but you can also make changes that only affect a specific widget, as described below.

Name a theme

In the dark theme template file, the first class selector is defined as:

.cx-widget.cx-theme-dark

.cx-widget is the base class for the entire Genesys Widgets UI. The outermost container of every widget or standalone UI element has this class and is used to identify UI elements that belong to Genesys Widgets.

.cx-theme-dark is the class name created for the dark theme. Themes are applied by searching for all elements with the **.cx-widget** class and appending the theme's classname to it. Thus, the combined class selector indicates styles that will be applied only when your custom theme is active in the configuration object.

Name your theme's classname anything you wish. There are no restrictions or limitations.

In a later step, you will register this theme classname in your configuration.

Customization guidelines

When you create your own themes, you can *only* use the following CSS properties:

- color
- background
- font-family
- font-style
- border-color
- border-style
- and other non-structural properties

> **Warning**
> Widgets primarily relies on classnames for CSS selectors, rather than fixed node path selectors. Using classnames allows for the HTML structure to be changed without breaking selectors. For example, the selector ".cx-webchat .cx-message" is all that is needed to target message bubbles inside WebChat. Using a fixed node path equivalent, like "div.cx-webchat > div.cx-body > div.cx-transcript > div.cx-message-group > div.cx-message" creates a dependency on the HTML node type and structure. If any changes are made to WebChat's HTML structure, this CSS selector will break. Use the smallest necessary specificity in your selectors and try to use classnames only.
>
> Be careful not to modify the structure and functionality of the CSS when you make changes. Otherwise, it won't work properly. In particular, avoid setting the following CSS properties: height, width, thickness, size, and visibility, or any other properties that change the structure of widgets. These properties are not supported, and changing them can break widget stability and usability.

> **Important**
> By default, the Widgets CSS uses the Roboto font, available at https://fonts.google.com.

Register a theme with Genesys Widgets

The following example shows how to register themes in the Genesys Widgets configuration.

```
window._genesys.widgets.main.themes = {
    "blue": "cx-theme-blue"
```

```
};
```

The name:value pair used here consists of a key ("blue") and the theme's CSS classname ("cx-theme-blue"). You can add as many themes to this list as you need.

Use a theme's key to make it the active theme:

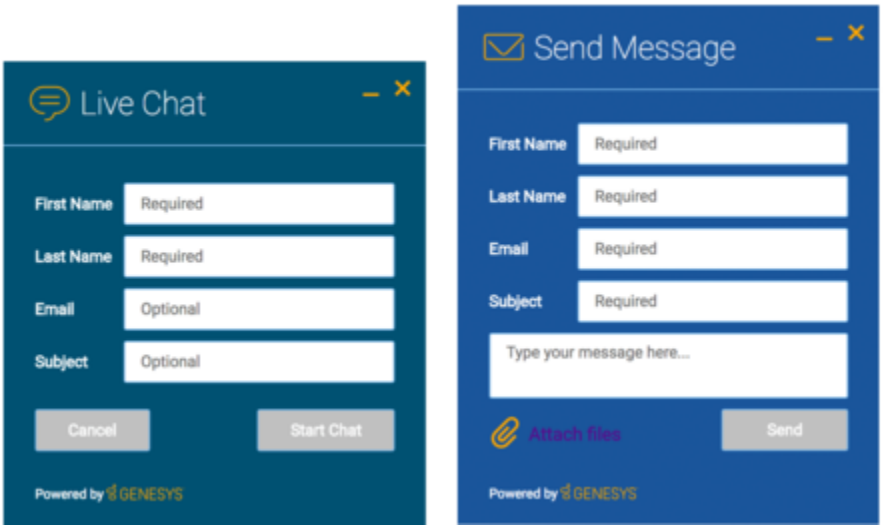
```
window._genesys.widgets.main.theme = "blue";
// OR
window._genesys.widgets.bus.command("App.setTheme", {theme: "blue"});
```

Change the appearance of a specific widget

You can specify specific widgets—and even specific elements within a widget—by appending the widget's CSS classname to the theme classname.

The following example shows how to extend the **cx-theme-blue** class with a widget-specific entry that makes the WebChat widget's background color a darker shade.

```
.cx-widget.cx-theme-blue, .cx-widget .cx-container{

    color: #FDFDFD;
    background: #1e5799;
}

.cx-widget.cx-theme-blue *{

    border-color: #7DB9E8;
}

.cx-widget.cx-theme-blue.cx-webchat, .cx-widget.cx-theme-blue .cx-webchat{

    background: #225897;
}
```

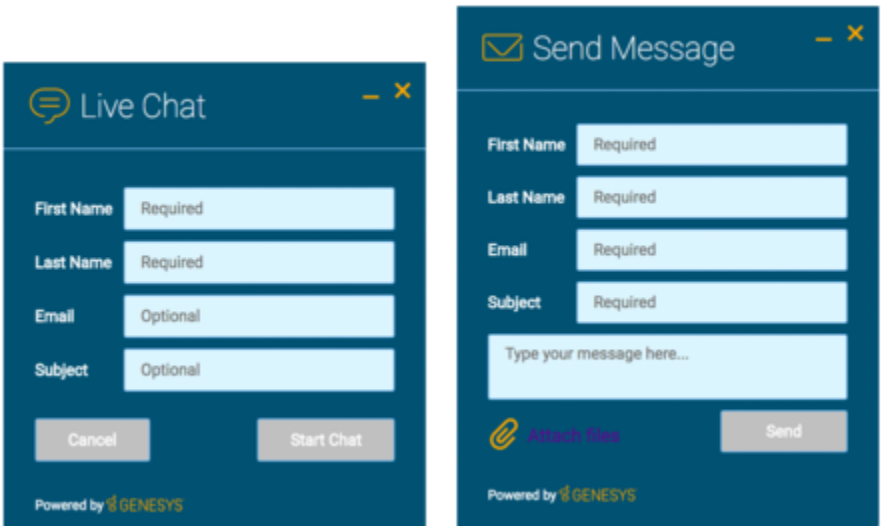


> **Important**
>
> Notice the dual CSS selector used when specifying the widget. This is required to make sure your styles always apply properly.

Widget-Specific and Element-Specific

The next example shows how to extend the "cx-theme-blue" class with a widget- and element-specific entry that changes the background color of the input fields within the WebChat widget to a light shade of blue.

```
.cx-widget.cx-theme-blue, .cx-widget .cx-container{

    color: #FDFDFD;
    background: #1e5799;
}

.cx-widget.cx-theme-blue *{

    border-color: #7DB9E8;
}

.cx-widget.cx-theme-blue.cx-webchat, .cx-widget.cx-theme-blue .cx-webchat{

    background: #225897; // Darker Shade
}

.cx-widget.cx-theme-blue.cx-webchat .form input, .cx-widget.cx-theme-blue .cx-webchat .form
input{

    background: #DCF5FF; // Lighter Shade
}
```



Change the layout and structure of a widget

You can only use themes to customize a limited set of styles for your version of Genesys Widgets. To create an alternate layout of your own design, disable the widget you want to customize and use the provided service plugins to build your own replacement.

Choosing Which Plugins to Load

Refer to the **plugins** configuration option here: app configuration

Service Plugins

Service plugins provide a high-level API for quickly integrating a UI with backend services. Each widget is matched with a corresponding service plugin. This separation allows for advanced integrations.

- WebChatService
- CallbackService

> **Warning**
> Genesys does not support changes to the layout of the official Genesys Widgets, as your changes can be overwritten when you upgrade to a newer version of Genesys Widgets.

Change fonts

By default, Genesys Widgets downloads and uses Google's Roboto font. Use the following CSS to specify a different font:

```
.cx-widget{ font-family: name-of-font-here; }
```

The font you choose here will be applied to all of the Genesys Widgets.

Disable Roboto font download

To prevent Google's Roboto font file from being downloaded at startup, set the **main.downloadGoogleFont** configuration option to **false**:

```
_genesys.widgets.main.downloadGoogleFont = false;
```

If this option is set to **true**, Google's Roboto font will be downloaded. The default value is *true*.

> **Important**
> Use this configuration option if you have security concerns about including fonts from third-party sources, to optimize your page load time, or if you already include Roboto on your website.

Change font size

By default, the font size in Genesys Widgets content is in *em* units. This is to support accessibility guidelines allowing font size to scale as needed when zoomed in or out based on the screen size. For normal text, the font size value is **0.75em** and can vary for other text contents.

> **Important**
> Since these are relative units, the actual value is derived from the font size of the parent page body. A base font size can be defined on the .cx-widget class in em units to change font size, which allows Widgets to calculate internal font size using this value.

Icons

Genesys Widgets are provided in SVG format, which means you can apply color fills and other SVG CSS properties when you use them. SVG also supports the highest possible rendering quality on all devices, regardless of the zoom level or resolution. You can scale the icons to fit any container; use them either inline or as blocks; and animate their orientation, colors, and other styling.

You can also use the Genesys icons in your own custom extensions, which allows them to match the look and feel of the default Genesys widgets. Here are some of the things you can do with them:

- Create a custom launcher button for chat, using the chat icon
- Create a custom widget with your choice of icon in the title bar
- Mix icons right in with your text, so you can refer to your widgets graphically

Genesys Widgets includes two sets of icons:

- The Multi-tone icon set uses several layers and colors per icon
- The Outline icon set takes a minimalist approach to both design and color

You can use these icons in any way that works for you, but please note that you can't customize or

replace the icons.

How to use icons

Automatic HTML injection

Specify which icons you want and where you want them by applying the CSS **cx-icon** class and **data-icon** attribute to the appropriate elements:

```
    ...SVG icon will be inserted here
```

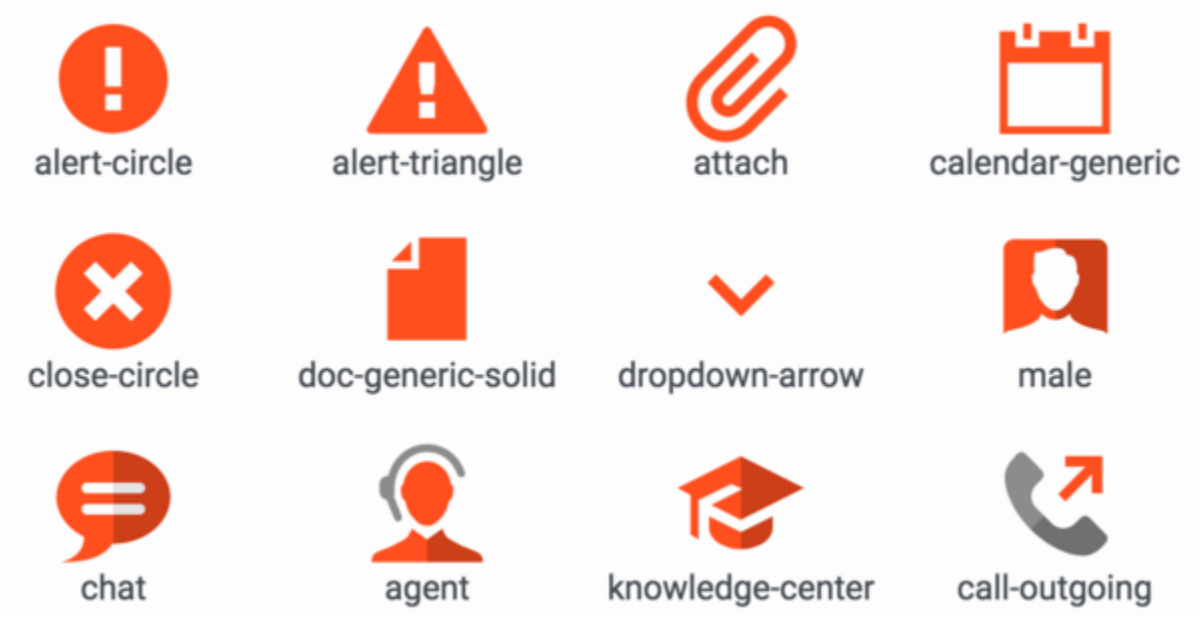
When you pass the element into the **CXCommon.populateAllPlaceholders($("#your-element"))** function—as a jQuery-wrapped set or an HTML string—Genesys Widgets inserts the appropriate SVG icons and returns the HTML to you.

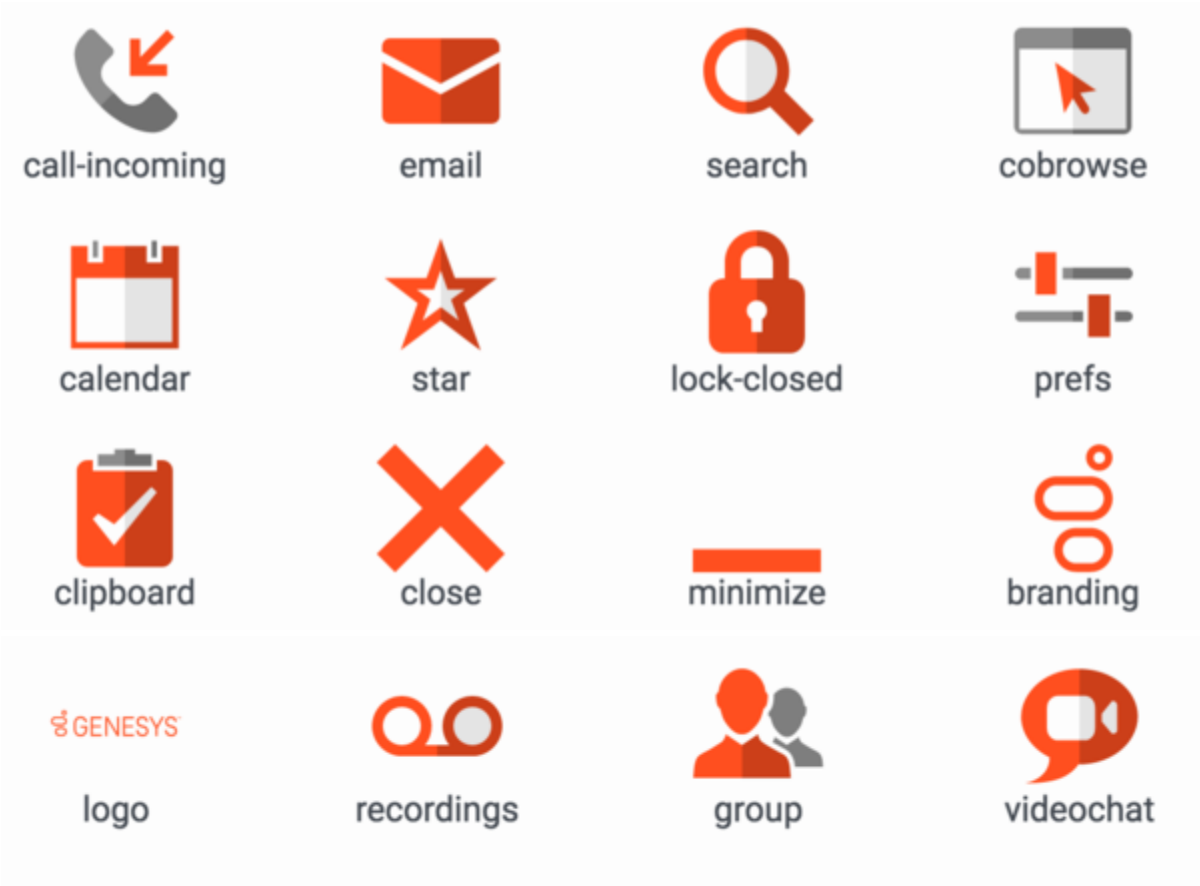
Fetching SVG icon markup

You can also fetch the markup for each SVG icon manually:

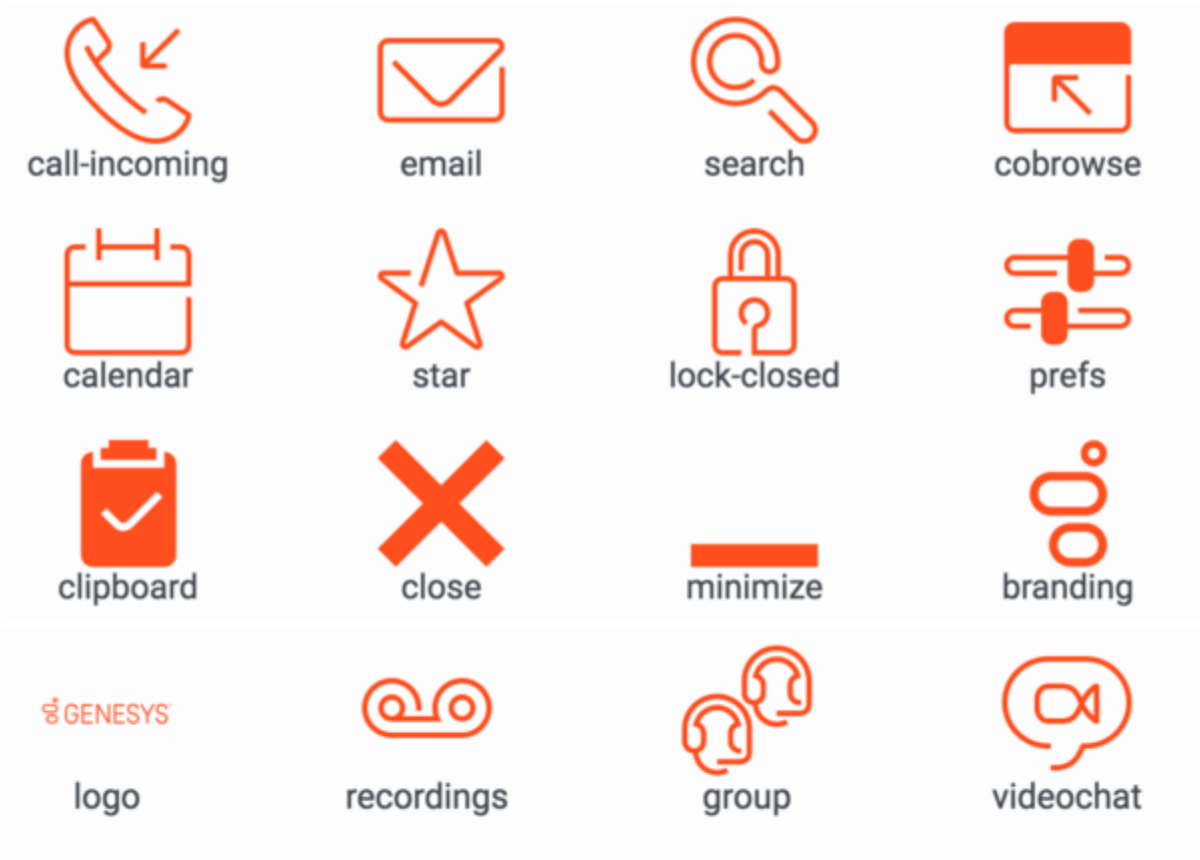
```
$("#your-element").append( CXCommon.Generate.Icon("chat") );
```

Multi-tone icon set





Outline icon set


call-incoming
email
search
cobrowse
calendar
star
lock-closed
prefs
clipboard
close
minimize
branding
GENESYS
logo
recordings
group
videochat

Rich Media

Contents

- 1 Quick Replies

- Developer

WebChat can display rich messages, which enable a more interactive digital experience with your customers. Quick replies help your customers quickly respond without having to type.

Related documentation:

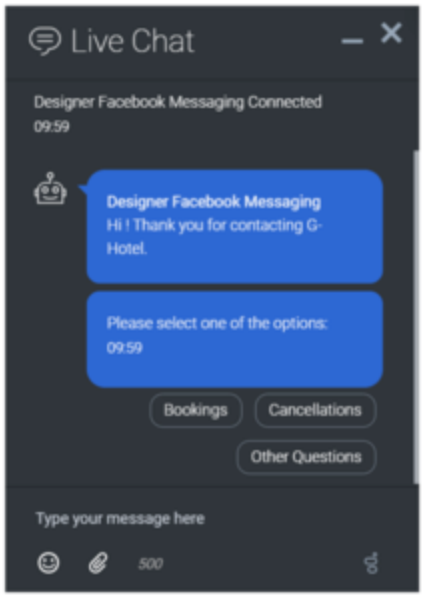
-

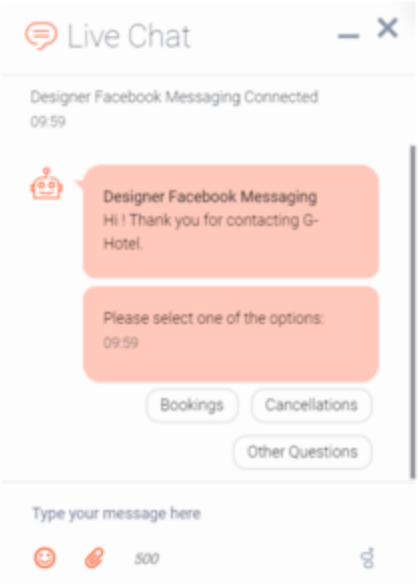
Quick Replies

> **Important**
> Quick Replies in Genesys Multicloud CX are only supported in bots. Agents cannot currently use them.

The WebChat Widget displays the **DTMF Key** prompts as *quick reply* buttons. Quick Replies offer the customer a choice of responses to the last chatbot message in the transcript. Tapping or clicking one of these Quick Replies posts that reply to the bot as a text message, which saves the customer from having to type a response manually.

Quick Replies are flexible. A chatbot can provide context-sensitive replies that aid in making a selection. Examples include polite responses (such as *OK*, *No*, *thank you,* or *Booking* or *Cancel)*, numeric responses, or the ability to choose a set of preset time slots. For more information, refer to Menu Block in the Designer User's Guide.


Live Chat
Designer Facebook Messaging Connected
09:59
Designer Facebook Messaging
Hi ! Thank you for contacting G-Hotel.
Please select one of the options:
09:59
Bookings
Cancellations
Other Questions
Type your message here
500


Live Chat
Designer Facebook Messaging Connected
09:59
Designer Facebook Messaging
Hi ! Thank you for contacting G-Hotel.
Please select one of the options:
09:59
Bookings
Cancellations
Other Questions
Type your message here
500

Accessibility

Contents

- 1 Overview
- 2 What is WCAG?
- 3 Support
  - 3.1 Web Content Accessibility Guidelines (WCAG) 2.1, Level AA Plugin Support by Platform
- 4 Screen reader support
- 5 Keyboard accessibility
- 6 Focus trap
- 7 Color contrast
- 8 Browser zoom and text resizing
- 9 Customization
  - 9.1 Localization
  - 9.2 Configuration options
- 10 Resources and tools used
  - 10.1 Online
  - 10.2 Screen readers

- Administrator
- Developer

Learn how Widgets aligns with the Web Content Accessibility Guidelines (WCAG) 2.1, Level AA.

Related documentation:

-

Overview

Genesys provides a Voluntary Product Accessibility Template® - VPAT® report from ITI, to document conformance of Widgets to WCAG 2.1 specification. The VPAT® report is a standardized template for documenting conformance to various accessibility specifications. The VPAT® report provided by Genesys follows the W3C/WAI's WCAG 2.1 specification, as this is an international standard adopted and recognized by our customers worldwide. The Genesys VPAT® can be downloaded here: Genesys Widgets WCAG 2.1 AA VPAT®.

What is WCAG?

Web Content Accessibility Guidelines (WCAG) 2.1 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content more accessible to a wider range of people with disabilities and will also often make Web content more usable to users in general. WCAG relies on four guiding principles for building accessible UIs:

1. Perceivable: Information and user interface components must be presentable to users in ways they can perceive.
2. Operable: User interface components and navigation must be operable.
3. Understandable: Information and the operation of user interface must be understandable.
4. Robust: Content must be robust enough that it can be interpreted by a wide variety of user agents, including assistive technologies.

Support

Widgets provides support for WCAG 2.1 Level AA for desktop and mobile web browsers. However, not all Widgets meet these guidelines. The table below lists the Widgets that address and meet the standard accessibility requirements to help assist users with vision, hearing, or mobility impairments in gaining greater access to the customer support.

Web Content Accessibility Guidelines (WCAG) 2.1, Level AA Plugin Support by Platform

Plugin	Genesys Cloud CX
WebChat	Level AA
CallUs	Level AA
ChannelSelector	Level AA
Engage	Level AA
SideBar	Level AA
Callback	Level AA

Widgets supports all the WCAG 2.1 Level AA accessibility guidelines for both mobile and desktop. Some of the high-level features are listed below.

Screen reader support

Supported widgets are accessible via screen readers, which announce the following: all the textual and non-textual content on the Widgets window elements, new chat messages sent by the agent to the user, outgoing messages sent by the user to the agent, and error messages. To achieve a consistent reading behavior of live data across all the screen readers and the browsers, recommended ARIA live regions have been implemented in WebChat for reading new messages.

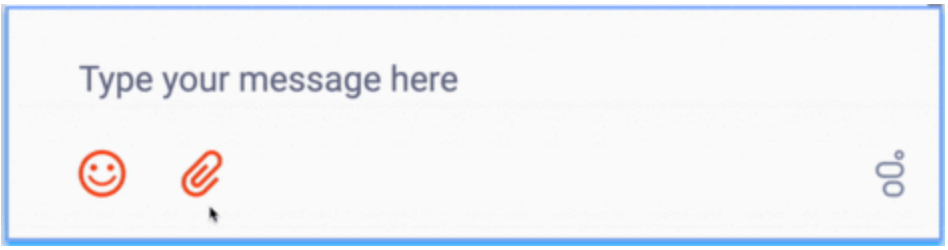
Genesys Widgets is built and maintained following WCAG A/AA accessibility standards. These standards are supported by popular screen readers, such as JAWS, VoiceOver (MacOS, iOS), TalkBack, and others.

Genesys supports compatibility with most commonly used screen readers in the following cases:

- **PC Windows OS:** JAWS with Chrome and Internet Explorer 11 or Edge
- **Mac OS**: VoiceOver with Chrome and Safari
- **iOS**: VoiceOver with Safari
- **Android**: TalkBack with Chrome

> ### Important
> Not all screen readers may read all the textual and non-textual window functionality. There are known issues around Firefox and Internet Explorer with some screen readers. The content is read as long as the screen reader model is supported on that particular browser.

Keyboard accessibility



Supported Widgets are accessible via the keyboard. Users may navigate to and within any widget using the **tab** key or **shift+tab** key combo. For dropdowns and the date picker, the user can highlight a selection using the **arrow** keys. The **enter** or **space** key can then be used to make a selection, send a message, or activate a button.

- **tab** - step forward to the next element
- **shift+tab** - step backward to the previous element
- **arrow keys** - move between options within a dropdown or date picker
- **enter** - make a selection or submit
- **space** - make a selection or activate a button

> **Important**
> In macOS, Safari Browser's accessibility settings must be enabled to allow for proper keyboard navigation in Widgets.

Focus trap

In desktop browsers, when the Engage Offer Widget is rendered in an overlay modal dialog with the background disabled, the focus is trapped within the content until it is closed. In mobile devices, all the widget layouts are expanded to full screen modal dialog. These mobile layouts contain the aria-modal property as recommended in the W3C ARIA Dialog modal best practices.

> **Important**
> Widgets does not add the "aria-hidden" attribute on the customer page html elements. Due to this limitation, when using screen reader gestures in some Android devices, the focus may not be trapped within the widget. To trap the focus, a custom event handling script needs to be added subscribing to the widget opened event. Also, add the "aria-hidden" attribute on the host page html elements, and remove them from subscribing to the corresponding closed/minimized events.

Color contrast

Text and background colors and buttons now meet WCAG 2.1 Level AA accessibility contrast guidelines. This allows text to be read clearly. There are changes in the default Widgets themes to increase color contrast in our Dark and Light themes. Changes include border, button, link, text, and background color adjustments to meet the contrast requirements while maintaining the same look and feel. In addition, there is an outline to indicate which element or section of each widget is in focus.

The following table details some examples of the changes included as part of WCAG implementation. The changes apply to both the light and dark themes, and the light theme is used in the table examples.

Description	Before	After
As per the “1.4.11 Non-text contrast” success criterion, icon color has been modified to meet the contrast requirement of at least 3:1 ratio against the adjacent/background color.		
As per the “1.4.3 Contrast (minimum)” success criterion, background color of the primary button has been modified to ensure that the contrast ratio of at least 4.5:1 exists between text and background.		
As per the “1.4.3 Contrast (minimum)” success criterion, placeholder text color has been modified to ensure that the contrast ratio of at least 4.5:1 exists between text and background.		
As per the “2.4.7 Focus visible & 1.4.11 Non-text contrast” success criterion, borders with 3:1 contrast ratio have been added to highlight the focused state of the menu items.		

Browser zoom and text resizing

Genesys Widgets supports zooming in and out, or resizing text using the browser's built-in controls. This makes it easier for some viewers to read text on the screen.

> **Important**
> The SideBar Widget can only support the zoom feature properly if it contains six or fewer rows.

Customization

Localization

Aria labels are used throughout Genesys Widgets to supply callouts and context for screen readers. These labels have been added to the standard localization language pack definition, allowing you to customize these labels yourself. All aria label strings are prefixed with "aria" to make them easy to identify. Review each widget's localization reference page to find these new aria labels. Example: WebChat Widget Localization reference.

Configuration options

Widget	Option name	Description
WebChat	ariaIdleAlertIntervals	An array containing the intervals as a percentage at which the screen reader will announce the remaining idle time. By default, it is enabled with the following time intervals, and it is customizable according to the user's needs. Configuring a value of `false` will let the screen reader call out idle time for every change.
WebChat	ariaCharRemainingIntervals	An array containing the intervals as a percentage at which the screen reader will announce the remaining characters when the user inputs text into the message area. By default, it is enabled with the following intervals, and it is customizable according to user needs. Configuring a value of `false` will let the screen reader call out remaining characters for every change.
WebChat	emojiList	emojiList must be configured with display names to support the screen reader calling out the emoji name. These emoji names are applied as **aria-label** attributes on the non-text emoji markup.

Resources and tools used

Online

- webaim.org
- deque.com
- contrastchecker.com

Screen readers

- JAWS
- NVDA
- VoiceOver