Developer's Guide

Customizing the Browser Tier Widgets

5/6/2025

Customizing the Browser Tier Widgets

Genesys Web Engagement includes pre-integrated Browser Tier widgets that are used for engagements. These widgets are based on HTML, CSS, and JavaScript, and can be customized to suit the look and feel of your website.

Warning

If you make customizations to the widget HTML files, they will not be backward compatible with any new versions of Genesys Web Engagement.

Invitation Widget

Overview

The default invitation approach in Genesys Web Engagement is represented by **invite.html** (chat and callback invitation). This HTML file, by default, has all the required dependencies embedded to avoid extra requests to the server.

The **invite.html** file has three code sections:

- Initial HTML Section
- JavaScript Third-party Libraries (dependencies) Section
- JavaScript Invitation Business Logic Section

Customization

There are four main ways you can customize the invitation widget:

- HTML/CSS — You can edit the HTML/CSS of your page or the **invite.html** file.
- Business Logic — You can modify the business logic including in the **invite.html** file to work with second- or third-party integration.
- Notification Service — You can change the JavaScript configuration through the Notification Service REST API.
- You can build your own version of the invitation. The default invitation widget is an example of a custom-built widget.

HTML/CSS

Important

In the paragraphs below, Genesys assumes that you have basic knowledge of CSS and HTML technologies.

If you need to change the basic style of the invitation (color, company logo, size, and so on) Genesys recommends that you use the HTML/CSS approach.

By default, the invite widget also contains all of the CSS needed for invite rendering, which is automatically added to the beginning of the <head> section of the web page when the invite is initialized.

If you need to modify these default styles, but you don't want to make it difficult to upgrade to newer versions of the widget, you can can create custom override styles. Make sure that your overrides are scoped to the components that need additional styling, and structure them so that they don't conflict with or overwrite any stand CSS files.

When overriding styles, consider the following points:

1. Review how the classes are assigned in the invite widget markup to better understand how they're applied (and how they can be overridden).
2. Create an override stylesheet. The best way to safely fine-tune a widget's appearance is to write new style rules that override the invite widget's styles and append these "override rules" in a separate stylesheet. Override rules are written against widget CSS class names and must appear in the source order after your theme stylesheet; since styles are read in order, the last style rule always takes precedence. By maintaining override styles in a separate file, you can customize the widget styles as much or as little as you'd like and still preserve the ability to easily upgrade the widget files as needed and simply overwrite your existing theme stylesheet, knowing that your override rules remain intact. Override rules can be listed in a dedicated stylesheet for overriding default website styles, or if you prefer to limit the number of files linked to your pages (and therefore limit the number of requests to the server), append override rules to the master stylesheet for your entire project.

To see exactly what you can override, you can use the developer tools that are commonly found in most modern web browsers. Currently, the Web Engagement CSS selector is not documented and there is no guarantee for backward compatibility for future versions of the invite widget.

Customization Examples

Change subject, message, buttons caption

```
<div title="New Subject" class="gpe-helper-hidden gpe-dialog">
    <div class="gpe-branding-logo"></div>
    <div class="my-message-content">
        <span>New Message</span>
    </div>
</div>
```

Change colors (message, subject, background, and so on)

For example, if you want to change the dialog style to red colors, you can add these styles to your page:

```
<style>
    .gpe-dialog .gpe-dialog-titlebar {
        background-color: red;
    }
    .gpe-dialog .gpe-button-text {
        color: red;
    }
</style>
```

Or message color:

```
<style>
    .gpe-dialog .message-content {
        color: blue;
        background-color: red;
    }
</style>
```

Or inline customization:

```
<div title="Chat" class="gpe-helper-hidden gpe-dialog">
    <div class="gpe-branding-logo"></div>
    <div class="message-content" style="color: #ffcc00; background-color: #0066ff "></div>
</div>
```

Invite Size

To change the size (width and height) of the invite widget, you can use following snippet:

```
<style>
    .gpe-dialog {
        width: 300px !important;
        height: 200px !important;
    }
</style>
```

Branding Logo

To customize the branding logo, you can use the CSS class "gpe-branding-logo". By default, the **invite.html** file uses an embedded image resource with a Data URI Scheme (http://en.wikipedia.org/wiki/Data_URI_scheme) in base64 format:

```
<div class="gpe-branding-logo" style="
    background-image: url(data:image/png;base64,iVBORw0KGgoAAAAN  ...  AAASUVORK5CYII=);
">
```

To customize the logo, you can generate the same base64 data code for your own image with the generator (http://base64converter.com/).

Alternatively, you can just use CSS:

```
<div class="branding-content" style=" background-image:url('myLogo.png'); "></div>
```

Business Logic

The **invite.html** file includes functions that you can change or replace for second- or third-party media integration:

- `init()`
- `startChat()`
- `startCallback()`
- `sendInviteResult()`
- `onAccept()`

Generally, you will need to work inside the `startChat()` or `startCallback()` functions, but you can also make additional changes in other functions. For example, if you need to integrate another type of media besides chat or callback, you can use `onAccept()` to extend the number of medias the invite supports. You must also be sure to make any necessary changes in the Engagement Logic Strategy.

Customization Examples

- Integration with Second- and Third-Party Media - Examples

Notification Service

The Notification Service is used to pass data to the invitation from the server. By default, data is composed in the **engage.workflow** of the Engagement Logic Strategy (**apps/*application name*/_composer-project/WebEngagement_EngagementLogic/Workflows/engage.workflow**).

You can use predefined commands in the Notification Service REST API to show your own invitation — particularly, gpe.callFunction and gpe.appendContent.

Customization Examples

- Notification Service REST API - Using the API to Customize Widgets

Localization

You can localize the invite by using the Notification Service REST API. Use the **subject**, **message**, **acceptBtnCaption** and **cancelBtnCaption** options to set specific text for the invite widget.

Chat Widget

Overview

The chat widget provides the main chat functionality for Genesys Web Engagement. It's a versatile widget that can be customized through the Chat Service JS API and the Chat Widget JS API.

Customization

There are three different customization types available for modifying the chat widget UI: Template-based, CSS-based, and JavaScript-based. Using these customization types, you can do any of the following:

- modify the structure of the widget
- add content
- add css classes
- modify the style (including the logo and buttons)
- use JavaScript UI hooks to modify the widget

For details about the customization types and how you can use them, see Customizing the User Interface, part of the Chat Widget JS API.

You can also use the Chat Service JS API to build your own chat widget and control chat sessions. Before creating your own chat widget, be sure to review the default chat widget — it's highly customizable through the Chat Widget JS API, and it also provides access to the same Chat Service JS API.

Localization

You can use the `startChat` and `restoreChat` methods of the Chat Widget JS API to enable localization for the chat widget. For details and step-by-step instructions, see Localization.

Callback Widget

Overview

The callback widget is represented by the **callback.html** file. The callback widget can only be used only in separate window mode and is currently not supported in embedded mode (like chat). The HTML file, by default, has all the required dependencies embedded to avoid extra requests to the server.

The **callback.html** file has three code sections:

- Initial HTML Section
- JavaScript Third-party Libraries (dependencies) Section
- JavaScript Invitation Business Logic Section

Customization

There are three main ways you can customize the callback widget:

- HTML/CSS — You can edit the HTML/CSS of your page or the **callback.html** file.
- Notification Service — You can change the JavaScript configuration through the Notification Service REST API.
- You can build your own version of the callback widget. The default callback widget is an example of a custom-built widget.

HTML/CSS

Important

In the paragraphs below, Genesys assumes that you have basic knowledge of CSS and HTML technologies.

If you need to change the basic style of the callback widget (color, company logo, size) Genesys recommends that you use the HTML/CSS approach.

By default, the callback widget also contains all of the CSS needed for rendering. You can override any of the default styles by adding a `<link>` or `<style>` tag with your own CSS rules for the callback widget.

To check which CSS you can override, you can use developer tools that are commonly found in most modern web browsers. Currently, the Web Engagement CSS selector is not documented and there is no guarantee for backward compatibility for future versions of the callback widget.

Customization Examples

Change color scheme
For example, you change the color scheme by adding the following CSS style section to the **callback.html** file:

```
<style>
    .callback-content{
        color:green
        background-color:#E1E2E3
    }
</style>
```

Note that by default we use embedded resources for **calback.html**. All scripts, style sheets, images are embedded.

Branding Logo
To customize the branding logo, you can use the CSS class "branding-content". By default, the **callback.html** file uses an embedded image resource with a Data URI Scheme (http://en.wikipedia.org/wiki/Data_URI_scheme) in base64 format:

```
<div class="branding-content" style="
           background-image: url(data:image/png;base64,iVBORw0K ... GK5CYII=);
       "></div>
</div>
```

To customize the logo, you can generate the same base64 data code for your own image with the generator (http://base64converter.com/).

Alternatively, you can just use CSS:

```
<div class="branding-content" style=" background-image:url('myLogo.png'); "></div>
```

Notification Service

The Notification service is used to pass data to the invitation from the server. By default, data is composed in the **engage.workflow** of the Engagement Logic Strategy (**apps/*application name*/_composer-project/WebEngagement_EngagementLogic/Workflows/engage.workflow**).

You can use predefined commands in the Notification Service REST API to show your own callback widget — particularly, gpe.callFunction and gpe.appendContent

Customization Examples

Change the page size and position
You can use the Notification Service to set the callback size and position. For example, you could use the gpe.setVariable method to add a global variable, called **com.genesyslab.gpe.invite.data**, with a value of `callbackPage` which includes the modified page size and position:

```
var notification_message = [
    {
        'page': event.pageID,
        'channel': 'gpe.setVariable',
        'data': {
            'variable': 'com.genesyslab.gpe.invite.data',
            'value': {
                callbackPage: {
                    pageWidth: 320,
                    pageHeight: 380,
                    pageTop: 150,
                    pageLeft: 150
                }
            },
        }
    },
    {
        'page': event.pageID,
        'channel': 'gpe.appendContent',
        'data': {
            'url': '/frontend/resources/invite.html'
        }
    }
];
```

Other examples

- Notification Service REST API - Using the API to Customize Widgets

Localization

The callback widget localization uses the jQuery Localize approach. A localization file for each language is represented by the JSON file, which contains one or more key-value pairs. The keys correspond to the HTML template.

Enable Localization

1. Open the Web Engagement installation folder and navigate to the ***application name*_composer-project\WebEngagement_EngagementWidgets\locale** directory. This folder contains the localization resources:
   - **callback-*lang*.json** for the callback invite.
2. To add a new supported language *lang* for these widgets, where ***lang*** is the short locale name of the language (en, fr, ru, and so on) or the full locale name in IETF (en-US, fr-FR), follow these steps:
   - Create a copy of the ***name of the widget*-en.json** locale file.
   - Rename it to: ***name of the widget-lang*.json**.
   - Edit ***name of the widget-lang*.json** and replace all the text values with your translations.
   - Save.
3. To deploy these localization files:
   - Stop the Web Engagement Servers.
   - Build your application.

- Deploy your application.
- Start the Web Engagement Servers.

4. Change instrumentation to use the new language. Use the "languageCode" option in the Tracker Script.

Example of Localization File (**callback-en.json**)

The JSON file might contain any of the fields listed below. Fields that are not present are taken from the built-in default localization.

```
{
    "windowTitle"               : "Genesys Web Callback",
    "firstName"                 : "First name:",
    "lastName"                  : "Last name:",
    "phone"                     : "* Phone:",
    "required"                  : "is a required field.",
    "callMe"                    : "Call Me",
    "cancel"                    : "Cancel",
    "message"                   : "Our representative will call you in a few minutes.",
    "message1"                  : "Please enter your contact details and click Call Me
button.",
    "message2"                  : "Next available customer representative will call you
shortly.",
    "messageFailCallback"       : "Callback is not available now. Try again later.",
    "yourPhone"                 : "Your phone number is",
    "validationPhoneRequired"   : "Phone number is required!",
    "validationPhoneWrong"      : "Phone number format +1(234)567-8910",
    "validationNameWrong"       : "Need more than 2 characters"
}
```