



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Developer's Guide

Engagement

Engagement

The Engagement Agent provides the engagement mechanism — proactive/reactive chat communication or web callback initialization.

Select a tab below for details about the engagement method.

<tabber> Chat JS Application=

To implement chat, you simply include the Chat JS Application script in your web pages. This short piece of regular JavaScript activates chat functions by inserting the **GWC.min.js** script into the page. The JavaScript asynchronously loads the application into your pages, which means that Chat JS Application does not block other elements on your pages from loading.

Basic Configuration

The simplest way to get the Chat JS Application for your site is by using the Genesys Web Engagement Plug-in for Genesys Administrator Extension. All you have to do is select the "Chat" option in the "Script Generator" window to include chat in the generated script. See [Generating the Instrumentation Script](#) for details.

Advanced Configuration

The Chat JS Application script consists of two parts: **script loader** and **configuration**. The script loader part actually loads the **GWC.min.js** script, while the configuration part sets options that control things like window size and localization.

Script Loader

To load Chat JS Application, you just need to include a short piece of regular JavaScript, the script loader, in your HTML. That JavaScript will asynchronously load the application into your pages, which means that Chat JS Application will not block other elements on your web page from loading.

For example, your script loader code might look like this:

```
//Script loader
(function(v) {
  if (document.getElementById(v)) return;
  var s = document.createElement('script'); s.id = v;
  s.src = ( 'https:' == document.location.protocol ? 'https://'<Frontend Server
host>:<Frontend Server secure port>' :
  'http://<Frontend Server host>:<Frontend Server port>' ) + '/frontend/
resources/js/build/GWC.min.js';
  s.setAttribute('data-gwc-var', v);
  (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('<_gwc>');
```

Important

The above example uses `_gwc` as the configuration global variable — see the "Configuration" section below for details.

Configuration

By default the chat application uses the `_gwc` global variable (you can change this in the script loader) that is created before Chat JS Application script loader is actually added to the page. Some of the options you set in the configuration code can be overwritten in the Chat Widget JS API methods (`startChat(options)` and `restoreChat(options)`) for a particular chat session, if the parameter name matches the option name.

For example, your configuration code might look like this:

```
/* Configuration */
var _gwc = {widgetUrl: 'http://<Frontend Server host>:<Frontend Server port>/frontend/
resources/chatWidget.html',
           serverUrl: 'http://<Backend Server host>:<Backend Server port>/backend/cometd'};
```

Options

Option	Type	Default Value	Mandatory	Description
serverUrl	string	undefined	yes, when default "transport" is used	URL of the CometD chat server for default (built-in) CometD transport.
widgetUrl	string	undefined	yes, when "embedded" is set to false ("popup" mode)	URL of the chat widget HTML that is open in an external window when operating in "popup" mode. By default, the chat widget is stored under the Frontend Server and is available at the following URL: <code>http://{frontend_server}:{frontend_port}/resources/chatWidget.html</code> ; however, you can store the chatWidget.html file as a static resourced under any third-party server.
embedded	boolean	false	no	Sets chat mode of operation: "embedded" (chat

Option	Type	Default Value	Mandatory	Description
				<p>widget is rendered directly on a page) or "popup" (chat opens in a separate browser window).</p> <p>Default is "popup". Pass the value true to switch to "embedded" mode.</p>
localization	string	undefined	no	<p>URL of JSON file with chat localization. If omitted, default English localization will be used.</p> <p>See Localization for more on how to localize the chat widget.</p>
windowSize	object {width: <number>, height: <number>}	{ width: 400, height: 500 }	no	Size of external chat window when operating in "popup" mode.
windowName	string	genesysChatWindow	no	<p>A string name for the new window that will be passed to the window.open call when opening chat widget window. For details, see https://developer.mozilla.org/en-US/docs/Web/API/Window.open.</p>
windowOptions	object	The value of the windowSize option.	no	<p>An object containing window options that are passed to the window.open call when opening chat widget window. You can pass any window options, such as position (top, left), whether to show browser buttons (toolbar), location bar (location), and so on. For details about possible window options, see</p>

Option	Type	Default Value	Mandatory	Description
				<p>(https://developer.mozilla.org/en-US/docs/Web/API/Window.open). All options are converted to a string that is passed to the <code>window.open</code> call.</p>
debug	boolean	false	no	Set to true to enable chat debugging logs (by default standard <code>console.log</code> is used, see the "logger" option if you want to override that).
logger	function	<code>console.log</code>	no	<p>Pass a function that will be used for chat logging (if <code>debug</code> is set to true) instead of the default <code>console.log</code>. The function has to support the interface of the <code>console.log</code> — it must accept an arbitrary number of arguments and argument types. To use the custom logging function in a separate window, you have to pass it directly on the widget page to the <code>startChatInThisWindow</code> method.</p> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> <p>Important</p> <p>The "logger" function works only for the Chat Widget JS API context.</p> </div>
registration	boolean or function	true	no	By default chat starts with a built-in registration form (that you can customize using <code>ui.onBeforeRegistration</code>).

Option	Type	Default Value	Mandatory	Description
				Pass the value false to disable this default built-in registration form. See Custom registration in the Chat Widget JS API for details.
userData	object	undefined	no	Can be used to directly attach necessary UserData to a chat session.
createContact	boolean	true	no	Determines whether new contact should be created from registration data if it doesn't match any existing contact. Only effective if registration data is present (collected either by built-in or custom registration workflow). See createContact in the Chat Widget JS API for details.
maxOfflineDuration	number	5	no	Time (in seconds) during which state cookies are stored after page reload/navigation. If cookies expire, the chat is not restored. Basically, this option means "how long shall the chat session live after the user leaves my website?"
ui	boolean or object	true	no	Pass the value false to disable the chat widget UI completely. Or pass an object with "hook" functions that can modify the built-in UI.

Option	Type	Default Value	Mandatory	Description
				See ui in the Chat Widget JS API for details.
transport	object	undefined	no	Custom transport instance (for example, REST-based).
disableWebSockets	boolean	false	no	<p>By default, chat attempts to use WebSockets to connect to the server. When the WebSocket connection is unavailable (for example, if your load balancer doesn't support WebSockets), chat switches to other, HTTP-based, means of communication. This might take some time (a matter of seconds, usually), so if you want to speed up the process, you can disable WebSockets for chat by passing true to this option.</p> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> <p>Important This option is only effective with default (built-in) transport.</p> </div>
templates	string	undefined	no	The URL of the HTML files containing templates that are used to render the chat widget. The request is made via either JSONP or AJAX, following the same logic as for localization files (see Localization in the Chat Widget JS API). Default

Option	Type	Default Value	Mandatory	Description
				<p>templates are included into the JavaScript source, so by default no requests are made to load them. The template system is based on the popular lodash / underscore templates: http://lodash.com/docs#template, http://underscorejs.org/#template</p>
autoRestore	boolean	true	no	<p>On every page reload/navigation, chat automatically attempts to restore the chat widget using the <code>restoreChat</code> method in the Chat Widget JS API. You can use this option to disable this behavior if you want more control over chat widget restoration.</p>
onReady	array or function	undefined	no	<p>This field is a callback function fired when the application has initialized. The Chat Widget JS API object is provided as the first argument of the callback function.</p> <pre> _gwc.onReady.push(function(chat) { alert('Chat application ready!'); }); </pre> <p>If you use <code>_gwc.onReady.push</code>, make sure that <code>onReady</code> is defined as an array.</p> <pre> var _gwc = { ... onReady: [] </pre>

Option	Type	Default Value	Mandatory	Description
				};

Configuration Examples

Basic configuration for proactive engagement integration

```
/* Configuration */
var _gwc = {widgetUrl: 'http://<Frontend Server host>:<Frontend Server port>/frontend/
resources/chatWidget.html'};

// Script loader
(function(v) {
  if (document.getElementById(v)) return;
  var s = document.createElement('script'); s.id = v;
  s.src = ('https:' == document.location.protocol ? 'https://<Frontend Server
host>:<Frontend Server secure port>':
'http://<Frontend Server host>:<Frontend Server port>') + '/frontend/resources/js/
build/GWC.min.js';
  s.setAttribute('data-gwc-var', v);
  (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('<_gwc>');
```

Basic configuration for reactive chat

```
/* Configuration */
var _gwc = {widgetUrl: 'http://<Frontend Server host>:<Frontend Server port>/frontend/
resources/chatWidget.html',
  serverUrl: 'http://<Backend Server host>:<Backend Server port>/backend/cometd'};

// Script loader
(function(v) {
  if (document.getElementById(v)) return;
  var s = document.createElement('script'); s.id = v;
  s.src = ('https:' == document.location.protocol ? 'https://<Frontend Server
host>:<Frontend Server secure port>':
'http://<Frontend Server host>:<Frontend Server port>') + '/frontend/resources/js/
build/GWC.min.js';
  s.setAttribute('data-gwc-var', v);
  (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('<_gwc>');
```

Advanced configuration for chat application

```
/* Configuration */
var _gwc = {
  serverUrl: 'http://<Backend Server host>:<Backend Server port>/backend/cometd',
  widgetUrl: 'http://<Frontend Server host>:<Frontend Server port>/frontend/resources/
chatWidget.html',
  autoRestore: true,
  debug: false,
  embedded: true,
  createContact: true,
  localization: 'http://<Frontend Server host>:<Frontend Server port>/frontend/resources/
locale/chat-fr.json',
  windowSize: { width: 400, height: 500 },
  windowName: 'myWindowName',
}
```

```
        windowOptions: {
          left: 0,
          top: 0
        },
        /* Callbacks */
        onReady: [function (chatAPI) {
          var options = {
            registration: true
          };
          chatAPI.startChat(options);
        }]
      };
    // Script loader
    (function(v) {
      if (document.getElementById(v)) return;
      var s = document.createElement('script'); s.id = v;
      s.src = ('https:' == document.location.protocol ? 'https://' : '<Frontend Server
host>:<Frontend Server secure port>') + '/frontend/resources/js/
build/GWC.min.js';
      s.setAttribute('data-gwc-var', v);
      (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
    })('_gwc');
```

Tip

For more information about the start parameters, see the [Chat Widget JS API](#)

Chat JS Application API

The Chat JS Application API is provided by the [Chat Widget JS API](#) component. The API object provides two functions: [startChat\(options\)](#) and [restoreChat\(options\)](#). To access the API, use the `onReady` option in the Chat JS Application configuration.

Reactive Chat

The following example shows how you can start reactive chat on a button click using the `startChat` method.

```
$('#startChatButton1, #startChatButton2, #startChatButton3').click(function () {
  _gwc.onReady.push(function (chatAPI) {
    chatAPI.startChat();
  });
});
```

If you want to provide monitoring information to the chat session, you should attached the **visitID** and **pageID** from the Tracker Application to the chat interaction.

```
$('#startChatButton1, #startChatButton2, #startChatButton3').click(function () {
  _gwc.onReady.push(function (chatAPI) {
    _gt.push(['getIDs', function (IDs) {
      chatAPI.startChat({userData: {visitID: IDs.visitID, pageID: IDs.pageID}});
    }]);
  });
});
```

```
});  
});
```

How To

Auto-generate an e-mail address based on the visitID

Use the Tracker JS Application and the Chat JS Application together:

```
_gwc.onReady.push(function (chatAPI) {  
  _gt.push(['getIDs', function (IDs) {  
    /* Start chat with generated email */  
    chatAPI.startChat({ userData: {  
      visitID: IDs.visitID,  
      pageID: ID.pageID,  
      email: IDs.globalVisitID + '@anonymous.com'  
    }});  
  }]);  
});
```

| -| Callback widget=

The callback widget is represented by the **callback.html** file, which can only be used in separate window mode — it is not currently supported for embedded mode like chat.

By default, the **callback.html** file has all its dependencies embedded to avoid extra requests to the server. The file is located in the

**GWE installation directory/apps/application_name/_composer-project/
WebEngagement_EngagementWidgets/** folder by default.

Warning

If you modify this file, it will not be backward compatible with any new versions of Genesys Web Engagement.

Configuration

To configure the callback widget, you can pass the following URL parameters (they must be **URL Encoded**):

```
http://{server}:{port}/frontend/resources/  
callback.html?visitId={visitId}&pageId={pageId}&backendUrl={backendUrl}&locale={locale}&debug=  
{debug}
```

Parameters

Option	Type	Default Value	Mandatory	Description
visitId	string	undefined	yes	Unique identifier of the current visit. For instance, 58bd8e65-7390-4c56-8da9-79dd74bd73be. You could use the Monitoring JS API to get this value.
pageId	string	undefined	yes	Identifier of the current page. For instance, 662FE0368D654E9D80B0D1E1E29AE25F. You could use the Monitoring JS API to get this value.
backendUrl	string	undefined	yes	URL of the Backend Server; for instance, http://<Backend Server host>:<Backend Server port>/backend.
locale	string	'en'	no	Localization tag for language and region; for instance, en-US. For details, see Localization .
debug	boolean	false	no	Set to true to show callback widget debug information in the browser console.

Configuration Example

```
http://<Frontend Server host>:<Frontend Server port>/frontend/resources/
callback.html?visitId=58bd8e65-7390-4c56-8da9-79dd74bd73be&pageId=662FE0368D654E9D80B0D1E1E29A
E25F&backendUrl=http%3A%2F%2F<Backend Server host>%3A<Backend Server
port>%2Fbackend&locale=en-US&debug=true
```

Usage

To run the callback widget, simply open it in a separate window with the appropriate parameters:

```
var url = http://<Frontend Server host>:<Frontend Server port>/frontend/resources/
callback.html +
```

Engagement

```
'?visitId=' + encodeURIComponent('58bd8e65-7390-4c56-8da9-79dd74bd73be') +
'&pageId=' + encodeURIComponent('662FE0368D654E9D80B0D1E1E29AE25F') +
'&backendUrl=' + encodeURIComponent('http://<Backend Server host>:<Backend Server
port>/backend') +
'&locale=' + encodeURIComponent('en-US') +
'&debug=' + encodeURIComponent('true');

window.open(url,
            title,
            'toolbar=no,location=no,directories=no,status=no,menubar=no,scrollbars=no,resizable=no,copyhistory=
+
            ',width=' + 400 + ',height=' + 500 + ',top=' + 300 + ',left=' + 300);
```

Customization

For details about how to customize the callback widget, see [Customizing the Browser Tier Widgets](#).