Widgets Developer Resources

7/22/2024

Table of Contents

Search the table of all articles in this guide, listed in alphabetical order, to find the article you need.

Related documentation:

-

Related documentation:

-

App

Contents

Learn how to control your widgets.

Related documentation:

-

Overview

App is the main controller for Genesys Widgets and has no UI. It controls all startup routines, global configurations, and extensions, and it executes the **onReady** event and distributes changes to theme, language, mobile mode, and other application-wide effects.

Usage

App's main interface is its configuration. You set all global defaults using the **window._genesys.widgets.main** property. App also has a few commands you can use to change the language and theme.

Customization

App itself cannot be customized, but its configuration options affect all widgets.

Mobile support

App has built-in mobile detection and can automatically notify all widgets to switch to mobile mode. You can also control this manually.

Configuration

Description

App uses the configuration property '_genesys.widgets.main'. App controls the Genesys Widgets product as a whole, handling themes, languages, and mobile devices.

Example

```
window._genesys.widgets = {

        main: {
                theme: 'dark',
                themes: {
```

```
                        dark: 'cx-theme-dark',
                        light: 'cx-theme-light',
                        blue: 'cx-theme-blue',
                        red: 'cx-theme-red'
                },
                lang: 'en',
                i18n: 'i18n.json',
                mobileMode: 'auto',
                mobileModeBreakpoint: 600,
                debug: true,
                header: {'Authorization': 'value'},
                cookieOptions: {
                        secure: true,
                        domain: 'genesys.com',
                        path: '/',
                        sameSite: 'Strict'
                }
        },
        onReady: function(){

                // Do something on Widgets ready
        }
}
```

Options

Name	Type	Description	Default	Required	Introduced/ updated
main.themes	object	An object list containing the CSS classname for each theme. The property names are used to select the theme in the 'theme' property, for example {dark:cx-theme-dark, light:cx-theme-light, red:cx-theme-red, blue:cx-theme-blue}. Where dark and light are the built-in themes provided in Genesys Widgets, red and blue are example custom theme	{dark: cx-theme-dark, light: cx-theme-light}	n/a	

Name	Type	Description	Default	Required	Introduced/ updated
		names you may create on your own. **Important** It is not necessary to define the dark and light theme as shown in this example. It is included to help show how the formatting works. Whatever you put in this object will be merged with the default themes object internally.			
main.theme	string	Selects the theme to apply to Genesys Widgets from the **themes** object. Uses the property name of the theme. For example using the example from themes above, possible values for this could be **dark, light, red, blue**.	dark	n/a	
main.lang	string	Select the language to use from the 'i18n' language pack. Language codes are selected by the customer. Any language code format can be used as long as this property matches one of the language codes in your i18n language pack. For more	en	n/a	

Name	Type	Description	Default	Required	Introduced/ updated
		information about localization, see localization.			
main.i18n	URL string or JSON	Either a path to a remote i18n.json language pack file or an inline JSON language pack definition. For more information about language packs, see localization.	en	Default English language strings are built into each widget and are displayed by default. Defining this i18n language pack overrides the built-in strings.	n/a
main.header	object	An object containing a key value pair for the authorization header.	n/a	n/a	9.0.002.06
main.preload	array	For use with lazy loading only. A list of plugins you want pre-loaded at startup. You may want certain plugins, such as SideBar, to be shown on screen as soon as possible; to do so, you may add *sidebar* to this preload plugins array so it will be loaded after Widgets starts up. The names you add to the list must match the first part of the plugin filename you wish to load. Example:	none	When lazy loading Widgets	

Name	Type	Description	Default	Required	Introduced/ updated
		sidebar will load **sidebar.min.js** from the **plugins/** folder. All filenames are lowercase. Important This preload array is intended for use when running widgets in lazy loading mode. You may also use this to pre-load your own custom-made plugins.			
main.mobileMode	boolean/string	Mobile Mode setting. *true* = Force Mobile Mode on all devices. *false* = Disable Mobile Mode completely. *auto* = Genesys Widgets Automatically switches between mobile and desktop modes using the *mobileModeBreakpoint* property and UserAgent detection.	auto	n/a	
main.timeFormat	number/string	This sets the time format for the timestamps. It can be 12 or 24.	12	n/a	
main.mobileModeBreakpoint	Breakpoint	The breakpoint width in pixels where Genesys Widgets will switch to Mobile Mode. Breakpoint checked at startup only.	600	n/a	
main.debug	boolean	Enable debug logging from	false	n/a	

Name	Type	Description	Default	Required	Introduced/ updated
		the bus to appear in the browser console.			
main.customStylesheetID	string	The HTML ID of a	n/a	n/a	
main.downloadGoogleFont	boolean	true	n/a		
main.deploymentID	String	The string used to customize cookie names so that multiple Widgets deployments can run in the same domain.	n/a	n/a	9.0.006.02
main.cookieOptions	object	An object containing cookie attributes that applies globally to all Widgets. The following cookie attributes are supported: 1. **secure** - Either true or false, indicating if the cookie transmission requires a secure protocol (https). 2. **domain** - A string indicating a valid domain where the cookie should be visible. 3. **path** - A string indicating the path where the	n/a	{sameSite:'Strict'}	9.0.017.01

Name	Type	Description	Default	Required	Introduced/ updated
		cookie is visible. 4. **expires** - Specifies the number of days, either from time of creation or from a date instance, until the cookie is to be removed. 'domain' and 'path' can be used to make cookies compatible with environments that use a non FQDN URL, such as an intranet hostname. However, the domain should only be manually set in production if the automated values are causing problems. Otherwise, rely on the automated domain and path. 5. **sameSite** - This maps to the cookie SameSite			

Name	Type	Description	Default	Required	Introduced/ updated
		attribute allowing the cookie to be restricted to a first-party or same-site context. It can take any of the supported values that SameSite attribute takes. Important The values are automatically set by Widgets to support cross-sub-domain cookies. Modifying these options overrides the automated values and might break cross-sub-domain cookie support if not properly set. For usage, please refer to the above example.			
onReady	function	A callback function that is invoked when the Widgets are ready and initialized with the configuration	none	n/a	

Name	Type	Description	Default	Required	Introduced/ updated
		provided.			

Localization

No localization options.

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('App.themeDemo');
```

setTheme

Sets the theme for Genesys Widgets from the list of registered themes. Default themes are 'light' and 'dark'. You can register as many new themes as you need.

Example

```
oMyPlugin.command('App.setTheme', {theme: 'light'}).done(function(e){

        // App set theme successfully

}).fail(function(e){

        // App failed to set theme
});
```

Options

Option	Type	Description
theme	string	Name of the theme you want to use. This name is specified in **window._genesys.main.themes**. Default themes are **light** and **dark**.

Resolutions

Status	When	Returns
resolved	Theme exists and is successfully changed.	The name of the theme that was chosen, for example *light*.
rejected	Theme does not exist.	Invalid theme specified.

getTheme

Get the CSS classname for the currently selected theme.

Example

```
oMyPlugin.command('App.getTheme').done(function(e){

        // App got theme successfully
        // e == CSS classname for current theme

}).fail(function(e){

        // App failed to get theme
});
```

Resolutions

Status	When	Returns
resolved	Always	CSS classname for the currently selected theme. For example: *cx-theme-light*
rejected	Never	n/a

reTheme

Accepts an HTML reference (either string or jQuery wrapped set) and applies the proper CSS Theme Classname to that HTML and returns it back. When widgets receive the 'theme' event from App, they pass-in their UI containers into App.reTheme to have the old theme classname stripped and new classname applied.

Example

```
oMyPlugin.command('App.reTheme', {html: '
Test Theme
'}).done(function(e){

        // App set theme successfully

}).fail(function(e){

        // App failed to set theme
});
```

Options

Option	Type	Description
html	string or jQuery Wrapped Set	HTML string or jQuery Wrapped Set you want to have modified.

Resolutions

Status	When	Returns
resolved	HTML is provided and theme is updated.	HTML that was passed-in and modified
rejected	No HTML is provided.	No HTML provided by [plugin name]

themeDemo

Start an automated demo of each theme. All registered themes will be applied with a default delay between themes of 2 seconds. You can override this delay. This command is useful for comparing themes or testing themes with official or custom widgets.

Example

```
oMyPlugin.command('App.themeDemo', {delay: 1000}).done(function(e){

        // App demo successfully started

}).fail(function(e){

        // App failed to start demo
});
```

Options

Option	Type	Description
delay	number	Number of milliseconds between theme changes. Default value is 2000 milliseconds.

Resolutions

Status	When	Returns
resolved	Always	n/a
rejected	Never	n/a

setLanguage

Changes the language

Example

```
oMyPlugin.command('App.setLanguage', {lang: 'eng'}).done(function(e){

        // App set language successfully started

}).fail(function(e){

        // App failed to set language
});
```

Options

Option	Type	Description
lang	string	Change the language of Genesys Widgets. Switches all strings in Widgets to selected language.

Resolutions

Status	When	Returns
resolved	Language is successfully changed.	n/a
rejected	No language code is provided.	No language code provided.
rejected	No matching language code is specified in your language pack.	No matching language code found in language pack.

closeAll

Publishes the **App.closeAll** event that requests all widgets to close.

Example

```
oMyPlugin.command('App.closeAll').done(function(e){

        // App closed all successfully

}).fail(function(e){
```

```
        // App failed to close all
});
```

Resolutions

Status	When	Returns
resolved	Always	n/a
rejected	Never	n/a

updateAJAXHeader

Introduced: 9.0.002.06

Updates the Authorization header.

Example

```
_genesys.widgets.bus.command('App.updateAJAXHeader', {header:
        {'Authorization': 'value'}
});
```

Resolutions

Status	When	Returns
resolved	Header is updated	n/a
rejected	Never	No request header found

removeAJAXHeader

Introduced: 9.0.002.06

Removes the set Authorization header.

Example

```
_genesys.widgets.bus.command('App.removeAJAXHeader');
```

Resolutions

Status	When	Returns
resolved	Always	n/a

registerExtension

Introduced: 9.0.002.06

Allows you to register and initialize new extensions at runtime instead of predefining extensions

before Genesys Widgets starts up.

Options

Option	Type	Description
undefined	function	Your extension function. Receives the following arguments: $ (jQuery), CXBus, Common.

Resolutions

Status	When	Returns
resolved	Valid *extension* object provided.	n/a
rejected	Invalid *extension* option provided.	n/a

registerAutoLoad

(For use with lazy loading only) Allows you to register a plugin into the preload plugins array so that it can be pre-loaded at the startup rather than lazy loading later. This can be useful when there is an active session maintained by your Widget and you would like to show it immediately at startup during page refresh or navigating across pages.

> **Important**
>
> This command is intended for use when running widgets in lazy loading mode. You may also use this to register and pre-load your own custom-made plugins.

Options

Option	Type	Description
name	string	The name of the plugin that needs to be registered for auto loading.

Resolutions

Status	When	Returns
resolved	A plugin is added into the preload list.	n/a
rejected	Never	n/a

deregisterAutoLoad

(For use with lazy loading only) Allows you to de-register a plugin from the preload plugins array so that it will not be pre-loaded at startup. This can be useful when there is no more active session maintained by your Widget and you don't want to show it on the screen immediately at startup.

```
Note: This command is intended for use when running widgets in lazy loading mode.
You may also use this to de-register your own custom-made plugins.
```

Options

Option	Type	Description
name	string	The name of the plugin that needs to be de-registered from auto loading.

Resolutions

Status	When	Returns
resolved	A plugin is removed from the preload list.	n/a
rejected	Never	n/a

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('App.ready', function(e){});
```

Name	Description	Data
ready	CallUs is initialized and ready to accept commands.	
i18n	Published when the language for	'(language code)'

Name	Description	Data
	Genesys Widgets is changed or is being set for the first time.	
theme	Published when the theme for Genesys Widgets is changed or is being set for the first time.	{theme: '(theme CSS classname)'}
timeFormat	Published when the time format for Genesys Widgets is changed or is being set for the first time.	{timeFormat: iTimeFormat}

Common

Contents

- Developer

Learn how to access Widgets utility functions and dynamically generate the common HTML containers used throughout Genesys Widgets.

Related documentation:

-

Common is a utility object available for import into Plugins/Widgets and Extensions. It is also accessible directly from the path **window._genesys.widgets.common**.

Common provides utility functions and dynamically generates common HTML Containers used throughout Genesys Widgets.

For all examples below, assume that **_genesys.widgets.common** has been stored in a local variable named *Common*.

```
var Common = _genesys.widgets.common;
```

Common.Generate.Container({options})

Dynamically generates a new HTML Container in matching the style of Genesys Widgets with the selected components you request in your options object. Returns the generated container HTML as a jQuery wrapped set.

Example

'Generate an Overlay Container'

```
var ndContainer = Common.Generate.Container({

        type: 'overlay',
        title: 'My Overlay',body: 'Some HTML here as a string or jQuery wrapped set',
        icon: 'call-outgoing',
        controls: 'close',
        buttons: false
}),
```

'Generate a Toast Container'

```
var ndContainer = Common.Generate.Container({

        type: 'generic',
        title: 'My Toast',body: 'Some HTML here as a string or jQuery wrapped set',
        icon: 'chat',
        controls: '',
        buttons: {
```

```
                type:'binary',
                primary: 'OK',
                secondary:'cancel'
        }
}),
```

Arguments

Argument	Type	Description
options	object	An object containing options to apply to the generated container.
options.type	string	`generic` or `overlay`. Overlay containers have special CSS properties for appearing inside the Overlay widget. Default is `generic`.
options.title	string	Title to apply to the container's titlebar area.
options.body	string or jQuery wrapped set	The HTML body you want the container to wrap.
options.icon	string	CSS Classname of icon to use.
options.controls	string	Select from a set of window control buttons to show at the top right. `close` = Show only the close button. `minimize` = Show only the minimize button. `all` = Show both close and minimize buttons.
options.buttons	object	Options for displaying action buttons at the bottom of the container, such as **OK** and **Cancel** buttons.
options.buttons.type	string	Currently, **binary** is the only supported button set at this time. Additional sets and arrangements will be available in a later release. Pass binary as the type here if you wish to show typical accept and dismiss buttons.
options.buttons.primary	string	Display name on the primary button. (for example **OK, Yes, Accept, Continue**, etc.)
options.buttons.secondary	string	Display name on the secondary button. (for example **Cancel, No, Dismiss, Reject**, etc.)

Common.Generate.Buttons({options})

Dynamically generates a new HTML Binary Button set in matching the style of Genesys Widgets with the selected options in your options object. Returns the buttons as a jQuery wrapped set.

Example

'Generate Binary Buttons'

```
var ndButtons = Common.Generate.Buttons({

        type: 'binary',
        primary: 'OK',
        secondary: 'Cancel'
}),
```

Arguments

Argument	Type	Description
options	object	Options for generating buttons, such as **OK** and **Cancel** buttons.
options.type	string	Currently **binary** is the only supported button set at this time. Additional sets and arrangements will be available in a later release. Please pass `binary` as the type here if you wish to show typical **accept** and **dismiss** buttons.
options.primary	string	Display name on the primary button. (for example **OK, Yes, Accept, Continue**, etc.)
options.secondary	string	Display name on the secondary button. (for example **Cancel, No, Dismiss, Reject**, etc.)

Common.Generate.Icon(name)

Dynamically generates an icon from the included icon set. Icons are in SVG format.

Example

'Generate Chat Icon'

```
var ndChatIcon = Common.Generate.Icon('chat');
```

'Insert Chat Icon'

```
$('#your_icon_container').append(Common.Generate.Icon('chat'));
```

Arguments

Argument	Type	Description
name	string	Select the icon you want to generate by name. See the icon reference page for icon names.

Common.Generate.Scrollbar(element, {options})

Dynamically generates a widget scrollbar for selected DOM element.

Example

'Generate Scrollbar for a container'

```
var scrollContainer = Common.Generate.Scrollbar($('#your_container'))
```

Arguments

Argument	Type	Description
element	DOM element or jQuery selector	Select the element to which you would like to apply scrollbar.
options	object	This is an iScroll component. So, all the options that iScroll supports can be passed here. For more details, refer to: http://iscrolljs.com/#configuring

Common.config(object)

Configure some debug options for Common at runtime.

Example

'Enable full debug logging'

```
Common.config({debug: true, debugTimestamps: true});
```

Arguments

Argument	Type	Description
object	object	Supported options are **debug** and **debugTimestamps**. Setting debug to `true` will enable debug messages created by **Common.log()**. Setting **debugTimestamps** to `true` will add timestamps to the front of each debug message created by **Common.log()**. Default value for both is `false`.

Common.checkPath(object, path)

Check for the existence of a sub-property of an object at any depth. Returns the value of that property; if found otherwise it returns **undefined**. Useful for checking configuration object paths without having to check each sub-property level individually.

Example

'Check for window._genesys.main'

```
var oMainConfig = false;

if(oMainConfig = Common.checkPath(window, '_genesys.main')){
        //... Utilize oMainConfig
}
```

Arguments

Argument	Type	Description
object	object	An object you want checked for a particular sub-property at any depth.
path	string	The object path in dot notation you wish to search for.

Common.createPath(object, path, value)

Related to checkPath, createPath lets you specify a target object and path string but lets you create the path and set a value for it. This saves you the pain of defining each node in the path individually. All nodes in your path will be created as objects. Your final node, the property you are trying to create, will be whatever value you assign it.

Example

'Create window._genesys.main'

```
var oMainConfig = false;

if(oMainConfig = Common.createPath(window, '_genesys.main', {debug:true})){
        //... Utilize oMainConfig
}
```

Arguments

Argument	Type	Description
object	object	An object you want to add your new path to.
path	string	The object path in dot notation you wish to create.
value	any	The value you want to assign to the final node (property) in your path.

Common.linkify(string, options)

Search for and convert URLs within a string into HTML links. Returns transformed string.

Example

'Check for window._genesys.main'

```
var sString = 'Please visit www.genesys.com';
sString = Common.linkify(sString, {target: 'self'});
// sString == 'Please visit www.genesys.com
```

Arguments

Argument	Type	Description
string	string	Any string you want to check for URLs and have them converted.
options	object	A list of options to apply to the linkify operation.
options.target	string	Choose the HTML TARGET attribute to apply to the generated links. Default is **_blank**. Set this option to `self` to apply the target **_self** to the generated links.

Common.log(mixed, type)

Log something to the browser's console. When using Common.log, _genesys.main.debug must be set to true to see your logs. This allows you to add debug logging to your code without worrying about unwanted debug messages in production. If timestamps are enabled, they will be prefixed to all messages printed through Common.log.

Example

'Check the contents of window._genesys.main'

```
var Common = _genesys.widgets.common;
Common.log(window._genesys.main);

if(!window._genesys.main){
        Common.log('window._genesys.main is not defined', 'error');
}
```

Arguments

Argument	Type	Description
mixed	Any	Any value or message you'd like to log.
type	string	You can specify the log type, such as `log`, `debug` and `error`. Default type is `log`. Note, if your browser doesn't support the `debug` or `error` log type, use `log` instead.

Common.sanitizeHTML(string)

Search for and escape characters within a string. Returns transformed string. Useful for escaping HTML.

Example

'Check for window._genesys.main'

```
var sString = 'Please visit www.genesys.com';

sString = Common.sanitizeHTML(sString);

 // sString == 'Please visit <a href='www.genesys.com' target='_self'>www.genesys.com</a>''
```

Arguments

Argument	Type	Description
string	string	Any string you want to be transformed.

Common.updateTemplateI18n(element, object)

Searches through an element's contents for i18n string elements to update with new strings. Used when updating the language in real-time. Works by searching for elements with the CSS classname 'i18n' and reading the custom attribute 'data-message' to match the string name in the language object. See example below.

Example

'Check for window._genesys.main'

```
var ndContainer = $('

');

Common.updateTemplateI18n(ndContainer, {CustomButton001: 'Accept'});

// ndContainer ==

```

Arguments

Argument	Type	Description
element	jQuery wrapped set	Element you want to search within to replace i18n strings.
object	Object of i18n Strings	The list of languages strings you want to update your UI with. This object comes from the App.i18n event or you can define your own custom object inline or using some other system. Object format is a simple name:value pair format. The **data-message** attribute on your HTML element must match one of these property names to be updated.

Common.debugIcons

Returns the list of all the Icons with their names that Widgets support.

Example

'Fetch and Display list of icons present in Widgets'

```
Common.debugIcons()
```

Common.debug

Adds debug logs in to the browser's console. When using Common.debug, _genesys.main.debug must be set to true to see your logs. This allows you to add debug logging to your code without worrying about unwanted debug messages in production. If timestamps are enabled, they will be prefixed to all messages printed through Common.debug.

Example

'Check the File upload limits in WebChatService'

```
Common.debug(data_server_returned_file_limits);
```

Arguments

Argument	Type	Description
mixed	Any	Any value or message you'd like to add debug log. **Note**: This is only supported if your browser supports debug log type.

Common.error

Adds error logs in to the browser's console. When using **Common.error**, **_genesys.main.debug** must be set to `true` to see your logs. This allows you to add error logging to your code without worrying about unwanted error messages in production.

Example

'Logging error messages'

```
Common.error('A widget plugin did not receive the following config: ....');
```

Arguments

Argument	Type	Description
mixed	Any	Any value or message you'd like to add error log. **Note**: This is only supported if your browser supports error log type.

Common.populateAllPlaceholders

Adds place holder content to the input elements in a form with the given text strings.

Example

'Show placeholders strings in a form'

```
Common.populateAllPlaceholders($('#your_form'), {strings})
```

Arguments

Argument	Type	Description
Form Selector	jQuery DOM selector for a form	Form containing input elements. Note: Input elements should contain i18n class name and data attribute **data-message-type** with value `placeholder` for the place holder details to appear.
Key/Value pairs	object	Placeholder messages that needs to be displayed. This is an object with key-value pairs where key should be equal to the **data-message** attribute value of an input element and value can be any text that you would like to display.

Common.populateLanguageStrings

Adds the preferred language place holder text to the given input elements in a form.

Example

'Show placeholders strings in a form'

```
Common.populateLanguageStrings($('#your_form'), {strings})
```

Arguments

Argument	Type	Description
Form Selector	jQuery DOM selector for a form	Form containing input elements. **Note**: Input elements should contain i18n class name and data attribute **data-message-type** with value `placeholder` for the place holder details to appear.
Key/Value pairs	object	Placeholder messages that needs to be displayed. This is an object with key-value pairs where key should be equal to the **data-message** attribute value of an input element and value can be any text that you would like to display.

Common.populateIcons

Show all the Icons on a Widget.

Example

'Populate all Widget Icons'

```
Common.populateIcons($('#your_continer'));
```

Arguments

Argument	Type	Description
element	jQuery DOM selector	Specify the widget container for which all the icons have to be displayed.

Common.insertIcon

Adds an icon before the selected element.

Example

'Insert a check mark icon to an element you desire.'

```
Common.insertIcon($('#your_element'), 'alert-checkmark', 'alert')
```

Arguments

Argument	Type	Description
element	jQuery DOM selector	An html element to which icon is to be displayed.
icon name	string	Name of the icon that you would like to display. **Note**: Refer to **Common.debugIcons** method to find out all the icon names that widgets support.
icon Aria Name	string	Name for the icon to be read by screen readers.

Common.injectScript

Injects javascript code dynamically into widgets with the help of a script tag.

Example

'Inject your Widget WebChat extension plugin.'

```
Common.injectScript('path/to/LoadWebChat.ext.js')
```

Arguments

Argument	Type	Description
Script file name	string path to JavaScript file	JavaScript file name that needs to be injected into widgets.

Common.mobileScreenScale

Re-sizes and fits widget to any mobile screen.

Example

'Fit your widget to any mobile screen.'

```
var mobileScaledWidget = Common.mobileScreenScale($('#your_widget'));
```

Arguments

Argument	Type	Description
element	jQuery DOM Selector	Your main widget wrapper container selector that contains the entire widget with **cx-titlebar, cx-body, cx-footer, cx-button-container** and **cx-message-container** classes in it.

Common.showLoading

Show loading spinner Icon.

Example

'Show loading spinner during an Ajax request'

```
Common.showLoading($('#your_container'))
```

Arguments

Argument	Type	Description
element	jQuery DOM Selector	An html container where loading spinner should appear. This adds a class name **cx-loading**.

Common.hideLoading

Remove loading spinner Icon.

Example

'Remove loading spinner after the Ajax request'

```
Common.hideLoading($('#your_container'))
```

Arguments

Argument	Type	Description
element	jQuery DOM Selector	An html container that contains the loading spinner.

Common.showWaiting

Show waiting icon.

Example

'Show waiting Icon when uploading a file.'

```
Common.showWaiting($('#your_container'),'waiting'))
```

Arguments

Argument	Type	Description
element	jQuery DOM Selector	An html container where waiting symbol should appear. This adds a class name **cx-waiting**.
Aria Label	string	The value of the **aria-label** attribute for the loading screen icon. The default value is `waiting`.

Common.hideWaiting

Remove waiting icon.

Example

'Remove waiting icon after file upload is done.'

```
Common.hideWaiting($('#your_container'))
```

Arguments

Argument	Type	Description
element	jQuery DOM Selector	An html container that contains the waiting symbol.

Common.watch

Repeat your function execution for every 'x' milliseconds (default 1 second) up to a maximum number of times (default - infinite) or till your function returns true.

Example

'Make Request Notifications until none are pending.'

```
Common.watch(function(iteration, maxIterations){

        if(bRequestNotificationsPending){
         // ..POST Request
        }
        return !bRequestNotificationsPending;
}, 3000, 30)
```

Arguments

Argument	Type	Description
function name	function	The function that you would like to execute. It should return `true/false`.
frequency	milliseconds	Execute the function for every *x* milliseconds until it returns `true`.
limit	number	The maximum number of times function is executed.

Common.addDialog

Create your own dialog box and append it in to the widget.

Example

'Add a dialog box on your preferred container div

```
Common.addDialog($('#your_container'), $('#your_dialog_box'), 'my_warning')
```

Arguments

Argument	Type	Description
element	jQuery selector	The parent container that holds the dialog box.
element	jQuery selector	The actual dialog box that you would like to display. This should contain the **data-dialog** attribute with the value equal to the dialog box name.
name	string	Dialog box name.

Common.showDialog

Show the dialog box that you prefer, using the dialog box name created with **Common.addDialog()**.

Example

'Show the dialog box created using Common.addDialog()'

```
Common.showDialog($('#your_container'), 'your_dialog_box_name');
```

Arguments

Argument	Type	Description
element	jQuery Selector	The parent container which has the dialog box appended in to it.
name	string	The actual dialog box name.

Common.hideDialog

Hide the dialog box that you showed using **Common.showDialog()**.

Example

'Hide dialog box'

```
Common.hideDialog($('#your_container'), 'your_dialog_box_name);
```

Arguments

Argument	Type	Description
element	jQuery Selector	The parent container that is showing the dialog box.
name	string	The actual dialog box name.

Common.hideDialogs

Hide all the dialog boxes. Dialog box name is not needed here.

Example

'Hide all dialog boxes.'

```
Common.hideDialogs($('#your_container'));
```

Arguments

Argument	Type	Description
element	jQuery Selector	The parent container that is showing all the dialog boxes.

Common.showAlert

Show a native alert dialog box on the widget you prefer with your own text message. By default, a primary button is added to dismiss the alert dialog.

Example

Show an alert dialog box on the Widget you prefer. But default it adds the dismiss button.

```
Common.showAlert($('.cx-widget.cx-webchat'), {text: 'your alert message', buttonText: 'Ok'})
```

Arguments

Argument	Type	Description
element	jQuery selector	The widget plugin container that should display the alert dialog. This should be the top level container wrapper holding the widget.
options	object	The data options containing the text to be shown on the Alert dialog box.
options.text	string	Display text on the Alert dialog box.
options.buttonText	string	Display text on the primary button (for example: **OK**).

Common.bytesToSize

Convert any number in bytes to Kilobytes, Megabytes, Gigabytes and Terabytes.

Example

'bytes to KB, MB, GB or TB.'

```
var fileSize = Common.bytesToSize(parseInt(fileSizeInBytes));
```

Arguments

Argument	Type	Description
bytes	number	Number in bytes size.

Common.getFormattedTime

Returns time in 12-hour or 24-hour format from the actual date timestamp. If no timestamp is provided, it uses current time.

Example

'convert date timestamp to return time in 12 hrs format'

```
var formattedTime = Common.getFormattedTime(timestamp, 12);
```

Arguments

Argument	Type	Description
timestamp	Date	JavaScript Date timestamp object.
format	number	Time format with value 12 or 24.

Overlay

Contents

- Developer

Learn how to use an overlay window control that widgets can inject their UI into.

Related documentation:

-

Overview

The Overlay plugin provides an overlay window control that widgets can inject their UI into, accepting the HTML UI, placing it inside an overlay control, and displaying the UI onscreen in a uniform overlay window fashion. This prevents individual widgets from managing the overlay themselves. It also means that each widget's UI can be moved between different container types.

Overlay provides these benefits:

- Shows the UI in the center of the window.
- Open and close transition animations.
- No overlapping overlays. Only one at a time. Automatically managed by the Overlay plugin.
- Auto-recenter as the browser window size is changed.
- Automatic application of mobile styles when running in mobile mode.

Usage

Overlay is easy to use; you simply open and close it. When you call **Overlay.open**, you pass in the HTML content you want to show. If you call **Overlay.open** again while an overlay is already open, it will automatically close the previous overlay before showing yours (unless the previous overlay has reserved the overlay to prevent new overlays).

> **Important**
>
> By default, the overlay has no visible styles or content. You must pass in the HTML you want to show inside the Overlay area. Typically you should create an overlay-type container using **Common.Generate.Container**, put your content inside that, then send the whole thing into **Overlay.open**.

Customization

Overlay does not have customization options.

Mobile support

Overlay automatically applies mobile CSS styles to its outer container to affect the content within the overlay view. It is up to the content inside the overlay view to dynamically change when the Genesys Widgets .cx-mobile CSS classname is applied to an outer container.

Configuration

Overlay does not have configuration options.

Localization

Overlay does not have localization options.

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('Overlay.close');
```

open

Opens the provided HTML in an Overlay View. When successful, it returns back the HTML and a custom close event for you to subscribe to. This alerts you when your overlay instance has been closed. You can also make your overlay immutable so that new overlay instances don't close yours. Only your widget can close its overlay when immutable is set to true.

Example

```
oMyPlugin.command('Overlay.open', {

        html: '
Template
',
```

```
        immutable: false,
        group: false

}).done(function(e){

        // Overlay opens successfully

}).fail(function(e){

        // Overlay failed to open
});
```

Options

Option	Type	Description
html	string	HTML string template for overlay window.
immutable	boolean	When set to `true`, overlay cannot be closed by other plugins.
group	string	The name of the overlay window group you want to add a new overlay view into.

Resolutions

Status	When	Returns
resolved	Overlay is successfully opened.	{html: , events: , group: }
rejected	No html template is passed.	No HTML content was provided. Overlay has ignored your command.
rejected	Overlay is already opened.	Overlay view is currently reserved.

close

Closes the Overlay UI. Publishes the appropriate custom close event for current overlay being closed.

Example

```
oMyPlugin.command('Overlay.close').done(function(e){

        // Overlay closed successfully

}).fail(function(e){

        // Overlay failed to close
});
```

Resolutions

Status	When	Returns
resolved	Overlay is successfully closed.	n/a
rejected	Overlay is already closed.	Overlay view is already closed.
rejected	Overlay view is immutable.	Overlay view is currently reserved.

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('Overlay.ready', function(e){});
```

Name	Description	Data
ready	The Overlay plugin is initialized and ready to accept commands	n/a

Toaster

Contents

- Developer

Learn how to use a toast view control into which widgets can inject their UI.

Related documentation:

-

Overview

The Toaster plugin provides a toast view control that widgets can inject their UI into, accepting the HTML UI, placing it inside a toast view, and displaying the UI onscreen in the lower-bottom-right of the screen. When it is opened, it slides up from the bottom. When it is closed, it slides down until it is offscreen.

Toaster provides these benefits:

- Shows UI as a slide-up toast view in the lower-bottom-right of the screen.
- Open and close transition animations.
- No overlapping toasts; only one at a time. Automatically managed by the Toaster plugin.

Usage

Toaster is easy to use; you simply open and close it. When you call **Toaster.open**, you pass in the HTML content you want to show. If you call **Toaster.open** again while a toast is already open, it will automatically close the previous toast before showing yours (unless the previous toast has reserved the view to prevent new toasts).

Namespace

The Toaster plugin has the following namespaces tied to each of the following types.

Type	Namespace
CXBus—API commands & API events	Toaster
CSS	.cx-toaster

Customization

Toaster does not have customization options.

Mobile support

Toaster does not have mobile-specific styles at this time.

Configuration

Toaster does not have configuration options.

Localization

Toaster does not have localization options.

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('Toaster.close');
```

open

Opens the Toaster UI.

Example

```
oMyPlugin.command('Toaster.open', {

        type: 'generic',
        title: 'Toaster Title',
        body: 'Toaster Body',
        icon: 'chat',
        controls: 'close',
        immutable: false,
        buttons:{
```

```
                type: 'binary',
                primary: 'Accept',
                secondary: 'Decline'
        }

}).done(function(e){

        // Toaster opened successfully

}).fail(function(e){

        // Toaster failed to open properly
});
```

Options

Option	Type	Description
type	string	Specifies the type of body content that can be provided to Toaster window. Generic type shows the default body content and custom type overrides the default html body content.
title	string	Heading title to display on the Toaster window.
body	string	Holds text value for Generic Toaster type and html string template for Custom Toaster type.
icon	string	The CSS class name for an icon.
controls	string	Show close and minimize controls on Toaster window.
buttons	object	Define the type of buttons.
buttons.type	string	Shows two buttons on the Toaster .
buttons.primary	string	Text to be shown on primary button.
buttons.secondary	string	Text to be shown on secondary button.
immutable	boolean	When set to true, Toaster cannot be closed by other plugins.

Resolutions

Status	When	Returns
resolved	Toaster is successfully opened.	n/a
rejected	No Toaster type is specified.	No content was provided. Toaster has ignored your command.
rejected	Toaster is already opened.	Toaster view is currently reserved.

close

Closes the Toaster UI.

Example

```
oMyPlugin.command('Toaster.close').done(function(e){

        // Toaster closed successfully

}).fail(function(e){

        // Toaster failed to close
});
```

Resolutions

Status	When	Returns
resolved	Toaster is successfully closed.	n/a
rejected	Toaster is already closed.	Toaster view is already closed.
rejected	Toaster view is immutable.	Toaster view is currently reserved.

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('Toaster.ready', function(e){});
```

Name	Description	Data
ready	The Toaster plugin is initialized and ready to accept commands.	n/a
closed	The Toaster plugin has been removed from the screen.	n/a

WindowManager

Contents

- Developer

Learn how to use the WindowManager plugin, which provides a controller for several different types of window groups in Genesys Cloud CX.

Related documentation:

-

Overview

The WindowManager plugin provides a controller for several types of window groups. HTML UIs added to these WindowManager groups are arranged and managed in accordance with each group's purpose.

One group type is *Dock View*. WebChat utilizes this group to show the toast-like UI docked in the lower-bottom-right of the screen. This group automatically stacks the widgets **horizontally**. When one of the widgets closes, the stack collapses toward the right. Widgets can register themselves into this WindowManager group and let it do all the work.

Another group type is *Side Button*. WebChat uses this group to show the launcher button on the right side of the screen. Like the Dock View, buttons are stacked, but in this case they are stacked **vertically**. As buttons are added and removed from the group, the button stack collapses to fill in the gaps.

Usage

WindowManager has "register" commands for registering your UI into different groups. They all accept one argument, the HTML you want to be handled by WindowManager. You can use 'registerDockView' or 'registerSideButton' at this time. More window management groups will be added in upcoming releases.

Customization

WindowManager does not have customization options.

Configuration

WindowManager does not have configuration options.

Localization

WindowManager does not have localization options.

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('WindowManager.registerDockView', {html: '
HTML
'});
```

registerDockView

Creates a docked view container to show a widget on the bottom right corner. Its position is adjusted (stacked) to appear beside another widget if already present and is indexed with a tabindex.

Example

```
oMyPlugin.command('WindowManager.registerDockView', {html: '
Template
'}).done(function(e){

        // WindowManager registered a dockView successfully

}).fail(function(e){

        // WindowManager failed to register a dock view
});
```

Options

Option	Type	Description
html	string	A Widget HTML string template that needs to be shown in dock view.

Resolutions

Status	When	Returns
resolved	The html template is successfully opened and registered in dock view.	n/a
rejected	No HTML template is found.	No html content

registerSideButton

Registers a button to show on the right side of the screen for a particular plugin. Its position is based on the respective plugin order defined in the array configuration. Currently, this is not supported for external plugins.

Example

```
oMyPlugin.command('WindowManager.registerSideButton', {template: '
Button Text
'}).done(function(e){

        // WindowManager registered a side button successfully

}).fail(function(e){

        // WindowManager failed to register a side button
});
```

Options

Option	Type	Description
template	string	Custom HTML string template for a button.

Resolutions

Status	When	Returns
resolved	The HTML button is successfully registered.	n/a
rejected	No HTML template is found.	No button template found to register

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('WindowManager.ready', function(e){});
```

Name	Description	Data
ready	WindowManager is initialized and ready to accept commands.	n/a
changed	WindowManager publishes this event when there is any change in the position of widgets on the screen.	{registry: (object)}

WebChatService

Contents

Learn how to use Genesys chat services in Genesys Cloud CX.

Related documentation:

-

Feature coming soon: Web messaging

If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat. To use web messaging, you configure tracking through the Messenger JavaScript SDK instead of deploying a tracking snippet.

Overview

WebChatService exposes high-level API access to Genesys chat services, so you can monitor and modify a chat session on the front end, or develop your own custom WebChat Widget. Compared to developing a custom chat UI and using the chat REST API, WebChatService dramatically simplifies integration—improving the reliability, feature set, and compatibility of every widget on the bus.

Usage

WebChatService and the matching WebChat Widget work together right out of the box and they share the same configuration object. Using WebChat uses WebChatService.

You can also use WebChatService as a high-level API using bus commands and events to build your own WebChat Widget or other UI features based on WebChatService events.

Namespace

The WebChatService plugin has the following namespaces tied to each of the following types:

Type	Namespace
Configuration	webchat
CXBus— API commands & API events	WebChatService

Customization

WebChatService has many configuration options but no customization options. It is a plug-and-play plugin and works as is.

Configuration

WebChat and WebChatService share the **_genesys.widgets.webchat** configuration namespace. WebChat contains the UI options and WebChatService contains the connection options.

> **Important**
>
> Starting with version 9.0.008.04, WebChatService allows you to choose between the types of chat services available in Genesys via the transport section configuration options.

For Genesys Cloud CX, the **transport.type** property should always be set to `purecloud-v2-sockets`.

Example

- Applicable to Genesys Cloud CX - Guest Chat APIs

```
window._genesys.widgets.webchat = {  transport: {
      type: 'purecloud-v2-sockets',
      dataURL: 'https://api.mypurecloud.com', // replace with API URL matching your region
      deploymentKey : 'YOUR_DEPLOYMENTKEY_HERE', // replace with your Deployment ID
      orgGuid : 'YOUR_ORGGUID_HERE', // replace with your Organization ID
      interactionData: {
         routing: {
         targetType: 'QUEUE',
         targetAddress: 'YOUR_QUEUENAME_HERE',
         priority: 2
         }
      }
   },
   userData: {
      addressStreet: '64472 Brown Street',
      addressCity: 'Lindgrenmouth',
      addressPostalCode: '50163-2735',
      addressState: 'FL',
      phoneNumber: '1-916-892-2045 x293',
      phoneType: 'Cell',
      customerId: '59606'
   }
}
```

Options

Name	Type	Description	Default	Required	Introduced/ updated
transport	object	Object containing the transport service configuration	N/A	Yes	9.0.008.04

Name	Type	Description	Default	Required	Introduced/ updated
		options.			
transport.type	string	Always set to `purecloud-v2-sockets` for use with Genesys Cloud CX. For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.	N/A	Yes	9.0.008.04
transport.dataURL	string (URL)	The Genesys Cloud CX WebChatService URL for your region. A list of API URLs per region is available in the Platform API section. For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.	N/A	Yes	9.0.008.04
transport.deploymentKey	string	Genesys Cloud CX widget deployment key. Identifies the widget on your web page as the one you created in the previous task (Create a widget configuration object). For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.	N/A	Yes	9.0.008.04
transport.orgGuid	string	Genesys Cloud CX organization ID; a unique GUID.	N/A	Yes	9.0.008.04

Name	Type	Description	Default	Required	Introduced/ updated
		For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.			
transport.pagination	boolean	Enable/disable pagination capability to restore the chat messages based on `transport.maxMessagePageSize` option. If set to false, chat messages will be restored all at once.	true	No	9.0.008.04
transport.maxMessagePageSize	number	Number of messages to be received per page during chat restore.	100	No	9.0.008.04
transport.interactionData.routing.targetType	string	Always set to 'QUEUE' to route to a queue.	N/A	Yes	9.0.008.04
transport.interactionData.routing.targetAddress	string	The queue name that receives chat messages. Example: *Support* For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.	N/A	Yes	9.0.008.04
transport.interactionData.routing.priority	integer	Priority level from 0 (lowest) to 10 (highest). For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.	N/A	No	9.0.008.04
transport.interactionData.routing.skills	array	List of skills. Example: [*Computers*, *Printers*].	N/A	No	9.0.008.04

Name	Type	Description	Default	Required	Introduced/ updated
		For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.			
transport.interactionData.routing.language	string	Requested agent language skill. Example: *English - Written*. For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.	N/A	No	9.0.008.04
userData	object	An object of key/value pairs of arbitrary custom data. For more details see Widget - Version 2 in Genesys Cloud CX Developer Center.	N/A	No	9.0.008.04

Localization

WebChatService doesn't have any localization options.

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');
```

```
oMyPlugin.command('WebChatService.getAgents');
```

> **Important**
>
> Starting with version 9.0.008.04, WebChatService allows you to choose between the types of chat API services available in Genesys via the transport section configuration options. For more information, see the Options table in configuration.

configure

Internal use only. The main App plugin shares configuration settings to widgets using each widget's `configure` command. The `configure` command can only be called once at startup. Calling `configure` again after startup may result in unpredictable behavior.

startChat

Initiates a new chat session with the chat server via GES or with the service configured under the transport section.

Example

```
oMyPlugin.command('WebChatService.startChat', {

        nickname: 'Jonny',
        firstname: 'Johnathan',
        lastname: 'Smith',
        email: 'jon.smith@mail.com',
        subject: 'product questions',
        userData: {}

}).done(function(e){

        // WebChatService started a chat successfully

}).fail(function(e){

        // WebChatService failed to start chat
});
```

Options

Option	Type	Description
nickname	string	Chat Entry Form Data: 'nickname'.
firstname	string	Chat Entry Form Data: 'firstname'.
lastname	string	Chat Entry Form Data: 'lastname'.
email	string	Chat Entry Form Data: 'email'.

Option	Type	Description
subject	string	Chat Entry Form Data: 'subject'.
userData	object	Arbitrary data to attach to the chat session (AKA attachedData). Properties defined here will be merged with default userData set in the configuration object.

Resolutions

Status	When	Returns
resolved	Server confirms session started.	(AJAX Response Object)
rejected	A chat session is already active.	There is already an active chat session.
rejected	AJAX exception occurs.	(AJAX Response Object)
rejected	Server exception occurs.	(AJAX Response Object)
rejected	userData is invalid.	malformed data object provided in userData property.

endChat

Ends the chat session with the chat server via GES or with the service configured under transport section.

Example

```
oMyPlugin.command('WebChatService.endChat').done(function(e){

        // WebChatService ended a chat successfully

}).fail(function(e){

        // WebChatService failed to end chat
});
```

Resolutions

Status	When	Returns
resolved	Active session is ended successfully.	(AJAX Response Object)
rejected	No chat session is currently active.	There is no active chat session.

sendMessage

Sends a message from the client to the chat session.

Example

```
oMyPlugin.command('WebChatService.sendMessage', {message: 'hi'}).done(function(e){

        // WebChatService sent a message successfully

}).fail(function(e){

        // WebChatService failed to send a message
});
```

Options

Option	Type	Description
message	string	The message you want to send.

Resolutions

Status	When	Returns
resolved	Message is successfully sent.	(AJAX Response Object)
rejected	No message text provided.	No message text provided.
rejected	No chat session is currently active.	There is no active chat session.
rejected	AJAX exception occurs.	(AJAX Response Object)

sendCustomNotice

Sends a custom notice from the client to the chat server. This request is used to deliver any custom notification between a custom client application and a custom agent desktop. Neither Genesys Widgets, nor Workspace, uses this out of the box.

Example

```
oMyPlugin.command('WebChatService.sendCustomNotice', {message: 'bye'}).done(function(e){

        // WebChatService sent a custom message successfully

}).fail(function(e){

        // WebChatService failed to send a custom message
});
```

Options

Option	Type	Description
message	string	A message you want to send along with the custom notice.

Resolutions

Status	When	Returns	Introduced/updated
resolved	Message is successfully sent.	(AJAX Response Object)	
rejected	AJAX exception occurs.	(AJAX Response Object)	
rejected	The server doesn't support receiving custom notices.	This transport doesn't support sendCustomNotice command.	9.0.008.04

sendTyping

Sends a "*Customer typing*" notification to the chat session. A visual indication will be shown to the agent.

Example

```
oMyPlugin.command('WebChatService.sendTyping').done(function(e){

        // WebChatService sent typing successfully

}).fail(function(e){

        // WebChatService failed to send typing
});
```

Options

Option	Type	Description
Message	String	The message you want to send along with the typing notification.

Resolutions

Status	When	Returns
resolved	AJAX request is successful.	(AJAX Response Object)
rejected	AJAX exception occurs.	(AJAX Response Object)
rejected	No chat session is currently active.	There is no active chat session.

sendFilteredMessage

Sends a message along with a regular expression to match the message and hide it from the client. Useful for sending codes and tokens through the WebChat interface to the Agent Workspace.

> **Important**
> Filters are now automatically stored and recalled on chat restore for the duration of the session.

Example

```
oMyPlugin.command('WebChatService.sendFilteredMessage', {

        message: 'filtered message',
        regex: /[a-zA-Z]/

}).done(function(e){

        // WebChatService sent filtered message successfully

}).fail(function(e){

        // WebChatService failed to send filtered message
});
```

Options

Option	Type	Description
message	string	Message you want to send but don't want to appear in the transcript.
regex	RegExp	Regular expression to match the message.

Resolutions

Status	When	Returns
resolved	There is an active session.	n/a
rejected	No chat session is currently active.	No active chat session.

addPrefilter

Adds a new pre-filter regular expression to the pre-filter list. Any messages matched using the pre-filters will not be shown in the transcript

> **Important**
> Filters are now automatically stored and recalled on chat restore for the duration of the session.

Example

```
oMyPlugin.command('WebChatService.addPrefilter', {filters: /[a-zA-Z]/}).done(function(e){

        // WebChatService added filter successfully
        // e == Object of registered prefilters

}).fail(function(e){

        // WebChatService failed to add filter
});
```

Options

Option	Type	Description
filters	RegExp or Array of RegExp	Regular Expression(s) to add to the prefilter list.

Resolutions

Status	When	Returns
resolved	Valid filters are provided.	Array of all registered prefilters.
rejected	Invalid or missing filters provided.	Missing or invalid filters provided. Please provide a regular expression or an array of regular expressions.

updateUserData

Updates the userData properties associated with the chat session. If this command is called before a chat session starts, it will update the internal userData object and will be sent when a chat session starts. If this command is called after a chat session starts, a request to the server will be made to update the userData on the server associated with the chat session.

Example

```
oMyPlugin.command('WebChatService.updateUserData', {firstname: 'Joe'}).done(function(e){

        // WebChatService updated user data successfully

}).fail(function(e){

        // WebChatService failed to update user data
});
```

Options

Option	Type	Description
n/a	object	userData object you want to send to the server for this active session.

Resolutions

Status	When	Returns	Introduced/updated
resolved	Session is active and userData is successfully sent.	(AJAX Response Object)	
rejected	Session is active and AJAX exception occurs.	(AJAX Response Object)	
resolved	Session is not active and internal userData object is merged with new userData properties provided.	The internal userData object that will be sent to the server.	
rejected	Session is active and the server doesn't support updating userData.	This transport doesn't support updating userData during an active chat session.	9.0.008.04

poll

Internal use only. Starts polling for new messages.

Example

```
oMyPlugin.command('WebChatService.poll').done(function(e){

        // WebChatService started polling successfully

}).fail(function(e){

        // WebChatService failed to start polling
});
```

Resolutions

Status	When	Returns	Introduced/updated
resolved	There is an active session.	n/a	
rejected	WebChatService isn't calling this command.	Access Denied to private command. Only WebChatService is allowed to invoke this command.	
rejected	No chat session is currently active.	previous poll has not finished.	
rejected	The server doesn't support polling.	This transport doesn't support polling.	9.0.008.04

startPoll

Starts automatic polling for new messages.

Example

```
oMyPlugin.command('WebChatService.startPoll').done(function(e){

        // WebChatService started polling successfully

}).fail(function(e){

        // WebChatService failed to start polling
});
```

Resolutions

Status	When	Returns	Introduced/updated
resolved	There is an active session.	n/a	
rejected	No chat session is currently active.	No active chat session.	
rejected	The server doesn't support polling.	This transport doesn't support polling.	9.0.008.04

stopPoll

Stops automatic polling for new messages.

Example

```
oMyPlugin.command('WebChatService.stopPoll').done(function(e){

        // WebChatService stopped polling successfully

}).fail(function(e){

        // WebChatService failed to stop polling
});
```

Resolutions

Status	When	Returns	Introduced/updated
resolved	There is an active session.	n/a	
rejected	No chat session is currently active.	No active chat session.	
rejected	The server doesn't support polling.	This transport doesn't support polling.	9.0.008.04

resetPollExceptions

Resets the poll exception count to 0. pollExceptionLimit is set in the configuration.

Example

```
oMyPlugin.command('WebChatService.resetPollExceptions').done(function(e){

        // WebChatService reset polling successfully

}).fail(function(e){

        // WebChatService failed to reset polling
});
```

Resolutions

Status	When	Returns	Introduced/updated
resolved	Always.	n/a	
rejected	The server doesn't support polling.	This transport doesn't support resetPollExceptions command.	9.0.008.04

restore

Internal use only. You should not invoke this manually unless you are using Async mode.

Example

```
oMyPlugin.command('WebChatService.restore').done(function(e){

        // WebChatService restored successfully

}).fail(function(e){

        // WebChatService failed to restore
});
```

Options

Option	Type	Description	Accepted values	Introduced/ updated
sessionData	string	The session data that is needed to restore the WebChat in Async mode. It is a JWT token string value. Applicable only when using WebChat with Genesys Multicloud CX v3	(JWT string token)	9.0.008.04

Option	Type	Description	Accepted values	Introduced/ updated
		API. For more information, see the **Genesys Multicloud CX v3** tab in the **Options** table in configuration.		

Resolutions

Status	When	Returns	Introduced/updated
resolved	Session has been found.	n/a	
rejected	Session cannot be found.	n/a	
rejected	Restoring chat session is in progress.	Already restoring. Ignoring request.	9.0.002.06
rejected	Chat session is already active.	Chat session is already active, ignoring restore command.	9.0.002.06
rejected	Trying restore chat session manually.	Access Denied to private command. Only WebChatService is allowed to invoke this command in Non-Async mode.	9.0.002.06

getTranscript

Fetches an array of all messages in the chat session.

> **Important**
>
> For more information on the fields included in JSON response, see Digital Channels Chat V2 Response Format.

Example

```
oMyPlugin.command('WebChatService.getTranscript').done(function(e){

        // WebChatService got transcript successfully
        // e == Object with an array of messages

}).fail(function(e){

        // WebChatService failed to get transcript
});
```

Resolutions

Status	When	Returns
resolved	Always	Object with an array of messages.

getAgents

Return a list of agents that have participated in the chat. Includes agent metadata.

Example

```
oMyPlugin.command('WebChatService.getAgents').done(function(e){

        // WebChatService got agents successfully
        // e == Object with agents information in chat

}).fail(function(e){

        // WebChatService failed to get agents
});
```

Resolutions

Status	When	Returns
resolved	Always	(Object List) {name: (String), connected: (Boolean), supervisor: (Boolean), connectedTime: (int time),disconnectedTime: (int time)}

getStats

Returns stats on chat session including start time, end time, duration, and list of agents.

Example

```
oMyPlugin.command('WebChatService.getStats').done(function(e){

        // WebChatService got stats successfully
        // e == Object with chat session stats

}).fail(function(e){

        // WebChatService failed to get stats
});
```

Resolutions

Status	When	Returns
resolved	Always	{agents: (Object), startTime: (int time), endTime: (int time),

Status	When	Returns
		duration: (int time)}

sendFile

[Introduced: 9.0.008.04]

Sends the file from the client machine to the agent.

Example

```
oMyPlugin.command('WebChatService.sendFile', {files: $('').attr('type', 'file') /* Only works
on UI, can not dynamically change */ }).done(function(e){

        // WebChatService sent file successfully

}).fail(function(e){

        // WebChatService failed to send file
});
```

Options

Option	Type	Description
files	File	A reference to a file input element (for example)

Resolutions

Status	When	Returns
resolved	The file sent is a valid type and size.	(AJAX Response Object)
rejected	The file sent is an invalid type.	(AJAX Response Object)
rejected	The number of uploads is exceeded.	(AJAX Response Object)
rejected	The file size exceeds the limit.	(AJAX Response Object)
rejected	The file size is too large or an unknown error occurs.	(AJAX Response Object)
rejected	The server doesn't support file uploads.	This transport doesn't support file uploads.

downloadFile

Downloads the file to the client machine. Example

```
oMyPlugin.command('WebChatService.downloadFile', {fileId: '1', fileName:
'myfile.txt'}).done(function(e){

        // WebChatService sent file successfully
```

```
}).fail(function(e){

        // WebChatService failed to send file
});
```

Options

Option	Type	Description
fileId	string	This is the ID of the file to be downloaded from the session.

Resolutions

Status	When	Returns
resolved	The file is downloaded successfully.	n/a

getSessionData

[Introduced: 9.0.002.06]

Retrieves the active session data at any time.

Example

```
oMyPlugin.command('WebChatService.getSessionData')
```

Resolutions

Status	When	Returns	Introduced/updated
resolved	Always, when using Chat via GMS API. For more information, see the **GMS** tab in the **Options** table in configuration.	{secureKey: (string), sessionID: (number/string), alias: (number/string), userId: (number/string)}	
resolved	Always, when using Chat via Genesys Multicloud CX v3 API. For more information, see the **Genesys Multicloud CX v3** tab in the **Options** table in configuration.	{participantId: (string), sessionId: {string), token: (string), transportId: (string)}	9.0.008.04
rejected	Never	undefined	

fetchHistory

[Introduced: 9.0.008.04]

This applies only in Asynchronous mode to fetch older chat messages. It does not fetch all of the

messages at once; rather a certain number of messages are fetched every time this command is called. Response data will be available in the messageReceived event.

Example

```
oMyPlugin.command('WebChatService.fetchHistory')
```

Resolutions

Status	When	Returns
resolved	Old messages are retrieved.	(AJAX Response Object)
rejected	Request fails.	(AJAX Response Object)
rejected	Asynchronous mode is not enabled.	Fetching history messages applies only to Asynchronous chat.
rejected	All messages are received.	No more messages to fetch.

registerTypingPreviewInput

Selects an HTML input to watch for key events. Used to trigger startTyping and stopTyping automatically.

Example

```
oMyPlugin.command('WebChatService.registerTypingPreviewInput', {input: $('input')
}).done(function(e){

        // WebChatService registered input area successfully

}).fail(function(e){

        // WebChatService failed to register typing preview
});
```

Options

Option	Type	Description
input	HTML Reference	An HTML reference to a text or textarea input.

Resolutions

Status	When	Returns
resolved	Valid HTML input reference is provided.	n/a
rejected	Invalid or missing HTML input reference.	Invalid value provided for the **input** property. An HTML element reference to a textarea or text input is required.

registerPreProcessor

Registers a function that receives the message object, allowing you to manipulate the values before it is rendered in the transcript.

Example

```
oMyPlugin.command('WebChatService.registerPreProcessor', {preprocessor: function(message){
         message.text = message.text + ' some preprocessing text';
         return message;
 } }).done(function(e){
         // WebChatService registered preprocessor function
         // e == function that was registered
 }).fail(function(e){
         // WebChatService failed to register function
 });
```

Options

Option	Type	Description
preprocessor	function	The preprocessor function you want to register.

Resolutions

Status	When	Returns
resolved	A valid preprocessor function is provided and is registered.	The registered preprocessor function.
rejected	An invalid preprocessor function is provided.	No preprocessor function provided. Type provided was ''.

verifySession

Checks for existing WebChat session before triggering a proactive invite.

Example

```
oMyPlugin.command('WebChatService.verifySession').done(function(e){

        if(e.sessionActive) {

                 // dont show chat invite

        } else if(!e.sessionActive) {

                 if(oMyPlugin.data('WebChat.open') == false){

                         // show chat invite

                } else {

                         // dont trigger chat invite
```

```
            }
        }
});
```

Resolutions

Status	When	Returns
resolved	A session exists or not.	A boolean `sessionActive` which holds the session state.

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('WebChatService.ready', function(e){});
```

Name	Description	Data	Introduced/updated
Started	Chat session has successfully started.	(AJAX Response containing session data)	9.0.008.04
restoreTimeout	Chat session restoration attempted was denied after user navigated away from originating website for longer than the time limit: default 60 seconds.	N/A	9.0.008.04
restoreFailed	Could not restore chat session after page navigation or refresh.	N/A	9.0.008.04
restored	Chat session has been restored after page navigation or refresh.	N/A	9.0.008.04
reconnected	Connection restored. This event is only	N/A	9.0.008.04

Name	Description	Data	Introduced/updated
	published after *disconnected*.		
ready	WebChatService is initialized and ready to accept commands.	N/A	9.0.008.04
messageReceived	A new message has been received from the server. Includes text messages, status messages, notices, and other message types.	{originalMessages: (object), messages: (array of objects), restoring: (boolean), sessionData: (object)}	9.0.008.04
error	An error occurred between the client and the server.	(AJAX Response)	9.0.008.04
ended	Chat session has successfully ended.	N/A	9.0.008.04
disconnected	Cannot reach servers. No connection. Either the user is offline or the server is offline.	N/A	9.0.008.04
clientTypingStopped	After a user stops typing, a countdown begins. When the countdown completes, the typing notification will clear for the agent.	N/A	9.0.008.04
clientTypingStarted	The user has started typing. Sends an event to the agent.	N/A	9.0.008.04
clientDisconnected	Indicates the user has been disconnected form the chat session.	{message: (object), agents: (object), numAgentsConnected: (number)}	9.0.008.04
clientConnected	Indicates the user has been connected to the chat session.	{message: (object), agents: (object), numAgentsConnected: (number)}	9.0.008.04
agentTypingTimeout	Agent typing event has been timed out.	(AJAX Response)	9.0.008.04
agentTypingStopped	Agent has stopped typing.	(AJAX Response)	9.0.008.04
agentTypingStarted	Agents has started typing a new message.	(AJAX Response)	9.0.008.04
agentDisconnected	Indicates an agent has disconnected from the chat.	{message: (object), agents: (object), numAgentsConnected: (number)}	9.0.008.04
agentConnected	Indicates an agent has connected to the chat.	{message: (object), agents: (object),	9.0.008.04

Name	Description	Data	Introduced/updated
		numAgentsConnected: (number)}	

CallUs

Contents

Learn how to display an overlay screen showing one or more phone numbers for customer service, as well as the hours that this service is available in Genesys Cloud CX.

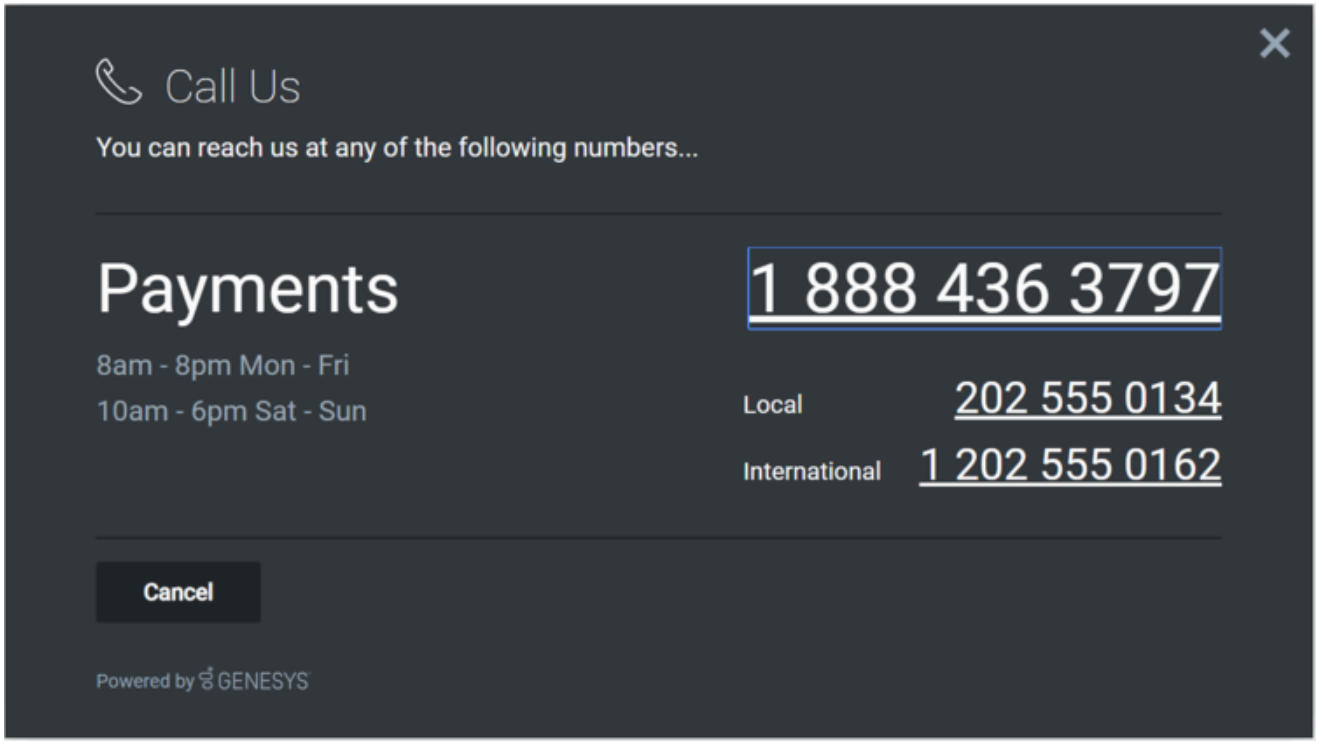
Related documentation:

-

Overview

The CallUs Widget provides an overlay screen showing one or more phone numbers for customer service, as well as the hours that this service is available. The arrangement of numbers in this layout starts with a main phone number, which can be followed by alternative or additional phone numbers. Each number can be named, and there is no limit to the number of phone numbers you can include. If the list of numbers doesn't fit in the widget, the user can scroll down to see the rest.

> **Important**
>
> A user can tap the phone numbers specified in the CallUs Widget in mobile browsers. Once the user taps any of the phone numbers, the mobile device will allow the user to dial the number through the mobile voice network.



Usage

Launch CallUs manually by using the following methods:

- Call the **CallUs.open** command
- Configure ChannelSelector to show CallUs as a channel
- Create your own custom button or link to open CallUs (using the "CallUs.open" command

> **Important**
> By default, a user has no way of launching the CallUs Widget. You must choose a suitable method for launching this widget.

Customization

You can customize and localize all the text, titles, names, and numbers shown in the CallUs Widget by adding entries into your configuration and localization options. There are no formatting requirements. Text will appear as you entered it.

> **Important**
>
> If you do not configure the CallUs Widget it will appear as an empty overlay. You must configure this Widget before using it.

CallUs supports themes. You can create and register your own themes for Genesys Widgets.

Namespace

The CallUs plugin has the following namespaces tied up with each of the following types:

Type	Namespace
Configuration	callus
i18n - Localization	callus
CXBus - API commands & API events	CallUs
CSS	.cx-call-us

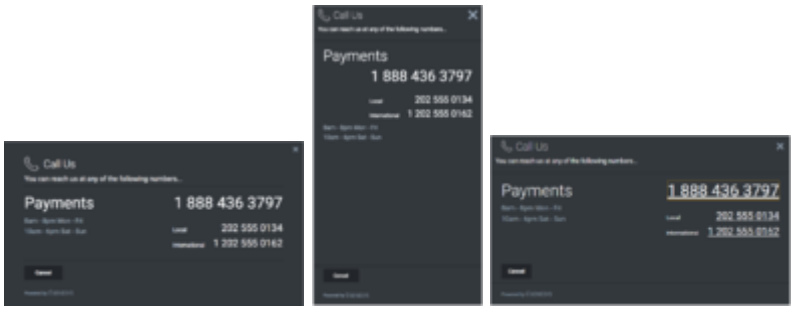
Mobile support

CallUs supports both desktop and mobile devices. Like all Genesys Widgets, there are two main modes: Desktop & Mobile. Desktop is employed for monitors, laptops, and tablets, and Mobile is employed for smartphones. When a smartphone is detected, CallUs switches to special full-screen templates that are optimized for both portrait and landscape orientations.

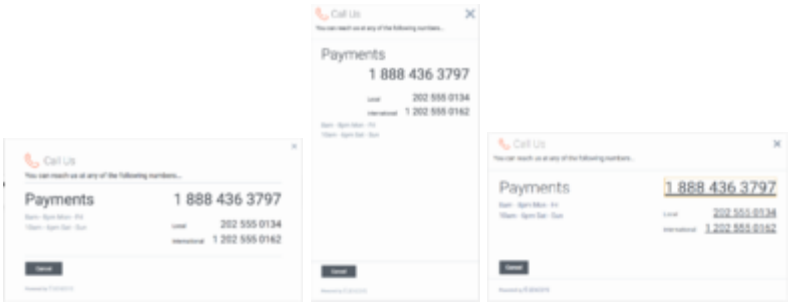
Switching between desktop and mobile mode is done automatically by default. You may configure Genesys Widgets to switch between Desktop and Mobile mode manually if necessary.

Screenshots

Dark theme



Light theme



Configuration

CallUs uses the **_genesys.widgets.callus** configuration property. You must specify all the numbers and labels that appear in the CallUs UI.

Example

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

        contacts: [

                {
                        displayName: 'Payments',
                        i18n: 'Number001',
                        number: '1 202 555 0162'
                },
                {
                        displayName: 'Local',
                        i18n: 'Number002',
                        number: '202 555 0134'
                },
                {
                        displayName: 'International',
                        i18n: 'Number003',
                        number: '0647 555 0131'
                }
        ],

        hours: [

                '8am - 8pm Mon - Fri',
                '10am - 6pm Sat - Sun'
        ]
};
```

Options

Name	Type	Description	Default	Required
contacts	array	An array of objects that represent	[]	true

Name	Type	Description	Default	Required

		phone numbers and their labels. The first number in this list displays as the larger, main number. Phone labels can be set directly using the 'displayName' property or you can use String Names from your localization file by setting the String Name in the 'i18n' property. 'i18n' overrides 'displayName'.
		
		Example
		
		`{`
		
		`"displayName": "Payments",`
		`"i18n": "Number001",`
		
		`"number": "1 202 555 0162"`
		`}`		

hours	array	Array of strings to appear stacked in the business hours section. Strings here are free-form. See screenshots for ideas.	[]	

Localization

> **Important**
>
> For information on how to set up localization, please refer to the Localize widgets and services guide.

Usage

Use the **callus** namespace when defining localization strings for the CallUs plugin in your i18n JSON file.

The following example shows how to define new strings for the **en** (English) language. You can use any language codes you wish; there is no standard format. When selecting the active language in your configuration, you must match one of the language codes defined in your i18n JSON file. Please note that you must only define a language code once in your i18n JSON file. Inside each language object you should define new strings for each widget.

Example i18n JSON

```
{
        "en": {
                "callus": {
                        "CallUsTitle": "Call Us",
                        "SubTitle": "You can reach us at any of the following NUMBERS...",
                        "CancelButtonText": "Cancel",
                        "AriaWindowLabel": "Call Us Window",
                        "AriaCallUsClose": "Call Us Close",
                        "AriaBusinessHours": "Business Hours",
                        "AriaCallUsPhoneApp": "Opens the phone application",
                        "AriaCancelButtonText": "Call Us Cancel"
                }
        }
}
```

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('CallUs.open');
```

open

Opens the CallUs UI.

Example

```
oMyPlugin.command('CallUs.open').done(function(e){

        // CallUs opened successfully

}).fail(function(e){

        // CallUs failed to open
});
```

Resolutions

Status	When	Returns
resolved	CallUs is successfully opened	n/a
rejected	CallUs is already open	'Already opened'

close

Closes the CallUs UI.

Example

```
oMyPlugin.command('CallUs.close').done(function(e){

        // CallUs closed successfully

}).fail(function(e){

        // CallUs failed to close
});
```

Resolutions

Status	When	Returns
resolved	CallUs successfully closed	n/a
rejected	CallUs is already closed	'Already closed'

configure

Internal use only. The main App plugin shares configuration settings to widgets using each widget's `configure` command. The `configure` command can only be called once at startup. Calling `configure` again after startup may result in unpredictable behavior.

Example

```
oMyPlugin.command('CallUs.configure', {

        contacts: [
                {
                        displayName: 'Payments',
                        i18n: 'Number001',
                        number: '1 888 436 3797'
```

```
                }
        ],
        hours: ['8am - 8pm Mon - Fri']

}).done(function(e){

        // CallUs configured successfully

}).fail(function(e){

        // CallUs failed to configure
});
```

Options

Option	Type	Description
contacts	Array	An array of objects that represent phone numbers and their labels. The first number in this list will display as the larger, main number.
hours	Array	Array of strings to appear stacked in the business hours section. Strings here are free-form.

Resolutions

Status	When	Returns
resolved	CallUs configuration is provided	n/a
rejected	No configuration provided	'Invalid Configuration'

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('CallUs.ready', function(e){});
```

Name	Description	Data
ready	CallUs is initialized and ready to accept commands	
opened	CallUs UI has been opened	
closed	CallUs UI has been closed	

ChannelSelector

Contents

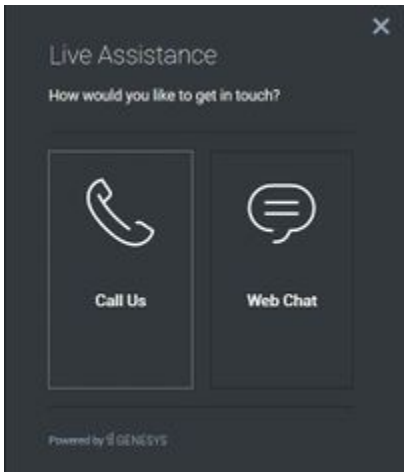
Learn how to provide your customers with a configurable list of channels as an entry point for contacting customer service in Genesys Cloud CX.

Related documentation:

-

Overview

The ChannelSelector Widget displays a configurable list of channels, as an entry point for customers to contact customer service. Channels are not limited to Genesys Widgets; you can add your own custom channels to open applications or open new windows as necessary.



Usage

Use the following methods to open ChannelSelector manually:

- Call the **ChannelSelector.open** command
- Create your own custom button, or link to open ChannelSelector (using the **ChannelSelector.open** command)

> **Important**
> By default, ChannelSelector has no channels configured. If not configured, the UI appears empty. See the configuration for examples and information on how to set up your own custom channels.

Customization

You can customize and localize the static text shown in the ChannelSelector Widget by adding entries into your configuration and localization options.

ChannelSelector supports themes. You can create and register your own themes for Genesys Widgets.

Namespace

The Channel Selector plugin has the following namespaces tied to each of the following types:

Type	Namespace
Configuration	channelselector
i18n—Localization	channelselector
CXBus—API commands & API events	ChannelSelector
CSS	.cx-channel-selector

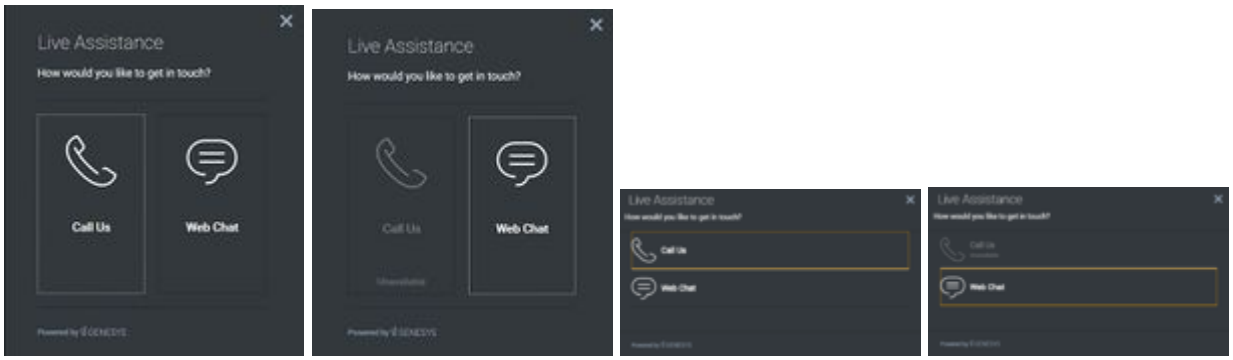
Mobile support

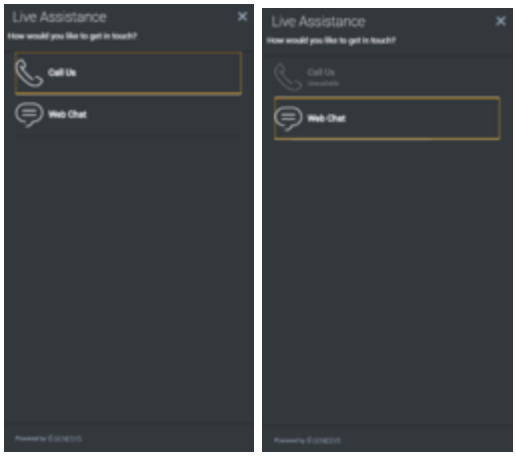
ChannelSelector supports both desktop and mobile devices. Like all Genesys Widgets, there are two main modes: Desktop and Mobile. Desktop is employed for monitors, laptops, and tablets. Mobile is employed for mobile devices. When ChannelSelector detects a mobile device, ChannelSelector switches to special full-screen templates, optimized for both portrait and landscape orientations.

Switching between Desktop and Mobile modes is automatic, by default. If necessary, configure Genesys Widgets to switch between Desktop and Mobile modes manually.

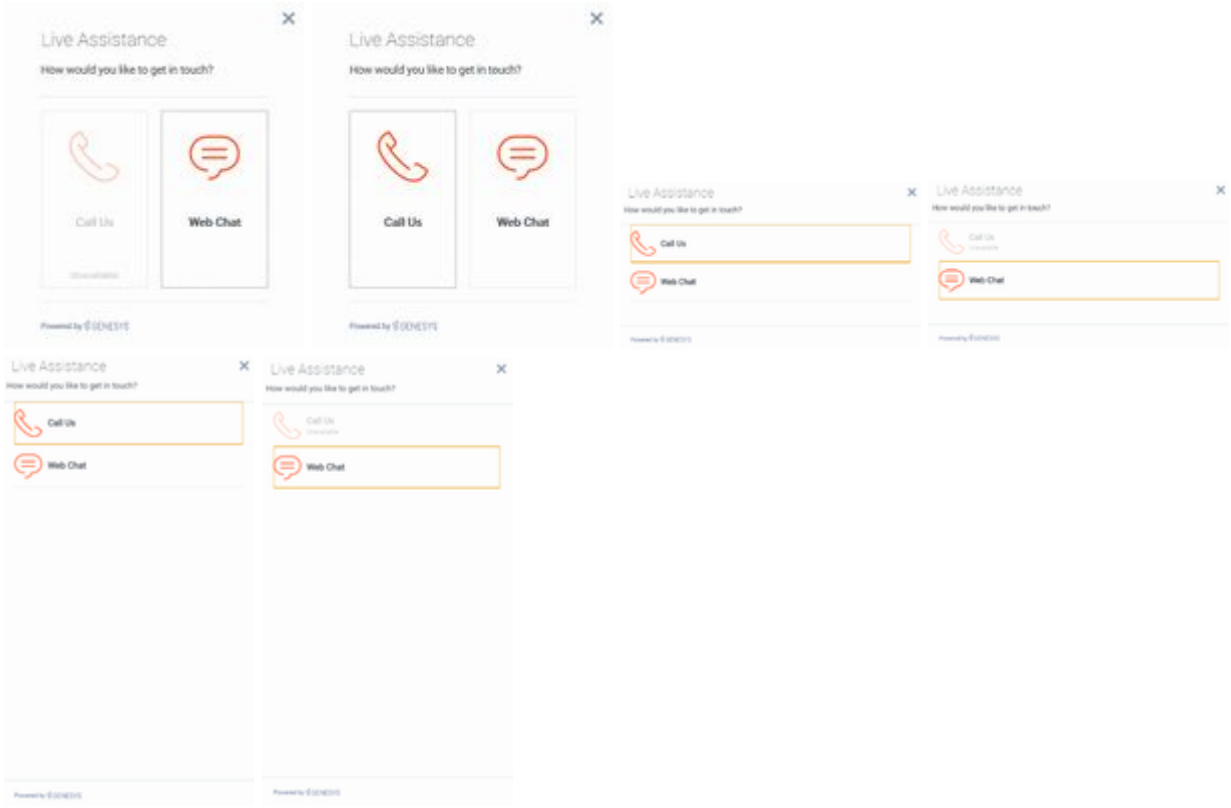
Screenshots

Dark theme





Light theme



Configuration

ChannelSelector shares the **_genesys.widgets.channelselector** configuration namespace. ChannelSelector has UI options to enable and disable channels, hide channels, add new channels. All

the channels are displayed based on the array of objects order defined in the channel's configuration. To hide a particular channel, simply remove the corresponding array object.

Example

```
  window._genesys.widgets.channelselector = {
         channels: [{

                  enable: true,
                  clickCommand: 'CallUs.open',
                  displayName: 'Call Us',
                  i18n: 'CallusTitle',
                  icon: 'call-outgoing',
                  html: '',
                  },
                 {
                  enable: true,
                  clickCommand: 'WebChat.open',
                  displayName: 'Web Chat',
                  i18n: 'ChatTitle',
                  icon: 'chat',
                  html: '',
                  }]
 };
```

Options

Name	Type	Description	Default	Required
channels[].enable	boolean	Enable/disable a channel.	true	N/A
channels[].clickCommand	string	The CXBus command name for opening a particular widget when this channel is clicked.	none	Always
channels[].displayName	string	A channel name to display on the ChannelSelector Widget.	none	Always
channels[].i18n	string	To support localization of the channel display name, this takes a key parameter of the channelselector section in the language pack file. Overrides above displayName.	none	N/A
channels[].icon	string	Select from one of the Genesys Widgets icons by specifying icon css class name.	none	Always

Name	Type	Description	Default	Required
channels[].html	string	Overrides and replaces the icon section of a channel with the html (image tag) defined here.	none	N/A

Localization

> **Important**
> For information on how to set up localization, refer to the Localize widgets and services guide.

Usage

Use the **channelselector** namespace when you define localization strings for the ChannelSelector plugin in your i18n JSON file.

The following example shows how to define new strings for the **en** (English) language. You can use any language codes you wish; there is no standard format. When selecting the active language in your configuration, you must match one of the language codes defined in your i18n JSON file. You must only define a language code once in your i18n JSON file. Inside each language object, you must define new strings for each widget.

Example i18n JSON

```
{
        "en": {
                "channelselector": {
                        "Title": "Live Assistance",
                        "SubTitle": "How would you like to get in touch?",
                        "UnavailableTitle": "Unavailable",                "AriaClose": "Live
Assistance Close",
                        "AriaWindowLabel": "Live Assistance Window"
                }
        }
}
```

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new

plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');
oMyPlugin.command('ChannelSelector.open');
```

close

Closes the ChannelSelector UI.

Example

```
oMyPlugin.command('ChannelSelector.close').done(function(e){

        // ChannelSelector closed successfully

}).fail(function(e){

        // ChannelSelector failed to close
});
```

Resolutions

Status	When	Returns
resolved	ChannelSelector is successfully closed	N/A
rejected	ChannelSelector is already closed	Already closed

open

Opens the ChannelSelector UI.

Example

```
oMyPlugin.command('ChannelSelector.open').done(function(e){

        // ChannelSelector opened successfully

}).fail(function(e){

        // ChannelSelector failed to open
});
```

Resolutions

Status	When	Returns
resolved	ChannelSelector Widget is successfully opened	N/A
rejected	ChannelSelector Widget is already open	'Already open'

configure

Modifies the ChannelSelector configuration.

Example

```
oMyPlugin.command('ChannelSelector.configure', {

        channels: [
                {
                        enable: true,
                        clickCommand: 'CallUs.open',
                        readyEvent: 'CallUs.ready',
                        displayName: 'Call Us',
                        i18n: 'CallusTitle',
                        icon: 'call-outgoing',
                        html:  ''
                }
        ]

}).done(function(e){

        // ChannelSelector configured successfully

}).fail(function(e){

        // ChannelSelector failed to configure
});
```

Options

Option	Type	Description
channels	array	Array containing each channel configuration object. The order of channels is displayed based on the order defined here.
channels[].enable	boolean	Enable/disable chat channel.
channels[].clickCommand	string	The CXBus command name for opening a particular widget when this channel is clicked.
channels[].readyEvent	string	Subscribes to this ready event published by a plugin and enables the channel when that plugin is ready.
channels[].displayName	string	A channel name to display in the

Option	Type	Description
		ChannelSelector Widget.
channels[].i18n	string	To support localization of channel display name, this takes a key parameter of the channelselector section in the language pack file. Overrides above displayName.
channels[].icon	string	Select from one of the Genesys Widgets icons by specifying icon css class name.
channels[].html	string	Overrides and replaces the icon section of a channel with the html (image tag) defined here.

Resolutions

Status	When	Returns
resolved	Configuration options are provided and set	N/A
rejected	No configuration options are provided	'Invalid configuration'

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('ChannelSelector.ready', function(e){});
```

Name	Description	Data
ready	ChannelSelector plugin is initialized and ready to accept commands	N/A
opened	ChannelSelector Widget has appeared on screen	N/A

Name	Description	Data
closed	ChannelSelector Widget has been removed from the screen	N/A

Console

Contents

- Developer

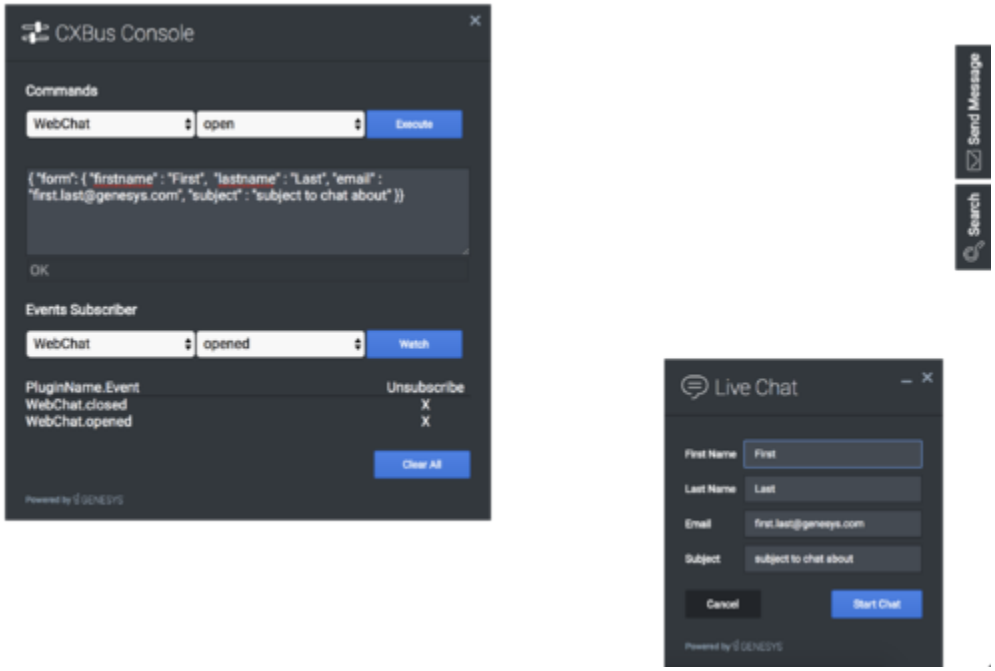
Learn how to debug commands and events on the widget bus.

Related documentation:

-

Overview

Use the Console Widget to debug commands and events on the widget bus. You can use dynamically populated lists to test, debug, or demo all of the commands. You can also create event watch lists that alert you when an event has fired.



Console provides an easy-to-use interface for debugging the widget bus that complements the standard command line methods. You can drag and drop the console anywhere on your screen, and when you refresh the page or move to another one, Console reappears right where you left it, as you left it. It is a great tool for getting to know the widget bus, the API for each widget, and debugging issues.

Usage

Launch WebChat manually by using the following methods:

- Call the **Console.open** command
- Configure the settings to show Console when the browser window is opened.
- Create your own custom button or link to open Console (using the **Console.open** command)

Configuration

Description

Console option to open on initial loading.

Example

```
window._genesys.widgets.console = {open: true};
```

Options

Name	Type	Description	Default	Required
open	boolean	Set to `true` for console to open at start.	false	false

Localization

> **Important**
> For information on how to set up localization, please refer to Localize widgets and services.

Strings

```
{
        "ConsoleTitle": "CXBus Console",
        "Commands": "Commands",
        "Plugin": "Plugin",
        "ConsoleErrorButton": "OK",
        "Execute": "Execute",
        "Event": "Event",
        "SubscribeTo": "Subscribe to",
        "Unsubscribe": "Unsubscribe",
        "ReturnData": "Return Data",
        "EventsSubscriber": "Events Subscriber",
```

```
        "Watch": "Watch",
        "pluginNameEvent": "PluginName.Event",
        "ClearAll": "Clear All",
        "OptionsSample": "JSON Formatted Options {'option': value}"
}
```

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('Console.open');
```

open

Opens the Console UI.

Example

```
oMyPlugin.command('Console.open').done(function(e){

        // Console opened successfully

}).fail(function(e){

        // Console failed to open
});
```

Resolutions

Status	When	Returns
resolved	Console is successfully opened	n/a
rejected	Console is already open	'Already opened'

close

Closes the Console UI.

Example

```
oMyPlugin.command('Console.close').done(function(e){

        // Console closed successfully

}).fail(function(e){

        // Console failed to close
});
```

Resolutions

Status	When	Returns
resolved	Console successfully closed	n/a
rejected	Console is already closed	'Already closed'

configure

Modifies the Console configuration options. See the Console configuration page.

Example

```
oMyPlugin.command('Console.configure', {

        open: false

}).done(function(e){

        // Console configured successfully

}).fail(function(e){

        // Console failed to configure
});
```

Options

Option	Type	Description
open	boolean	If setting is open: true, the console will automatically be open when Widgets is launched and the console is ready.

Resolutions

Status	When	Returns
resolved	Console configuration is provided	n/a
rejected	No configuration is provided	'Invalid Configuration'

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('Console.ready', function(e){});
```

Name	Description	Data
ready	Console is initialized and ready to accept commands.	n/a
opened	The Console Widget has appeared on screen.	n/a
closed	The Console Widget has been removed from the screen.	n/a

SideBar

Contents

Learn about the Sidebar widget, which customers use to launch other widgets with a single click.

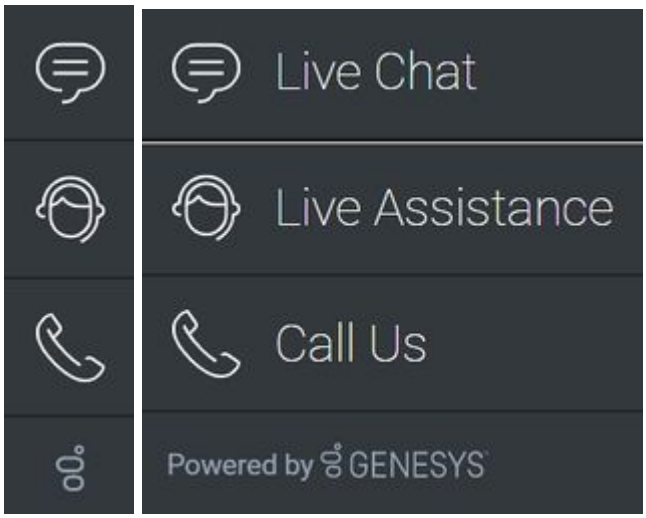
Related documentation:

-

Overview

Use the SideBar to launch other widgets with a single click. By default, SideBar is displayed on the right side of the screen, and you can configure any launchable widgets onto SideBar, including your custom extension widgets. The SideBar UI expands when you hover your cursor over it, and contracts when you move the cursor away. Other features include configurable positioning and mobile support. You can also add new configurations on the fly, which automatically re-renders the SideBar.

The image on the left shows SideBar when it is initially loaded, while the one on the right shows what it looks like when it's expanded.



Usage

Use the following methods to launch SideBar manually:

- Call the **SideBar.open** command
- Configure SideBar to show and launch custom widgets

Dependency

You must configure at least one customer-facing UI widget in order to use the SideBar Widget.

Customization

You can customize and localize all the text shown in the SideBar Widget by adding entries to your configuration and localization options.

SideBar also supports themes. You can create and register your own themes for Genesys Widgets.

Namespace

The SideBar plugin has the following namespaces tied to each of the following types:

Type	Namespace
Configuration	sidebar
i18n - Localization	sidebar
CXBus -API commands & API events	SideBar
CSS	.cx-sidebar

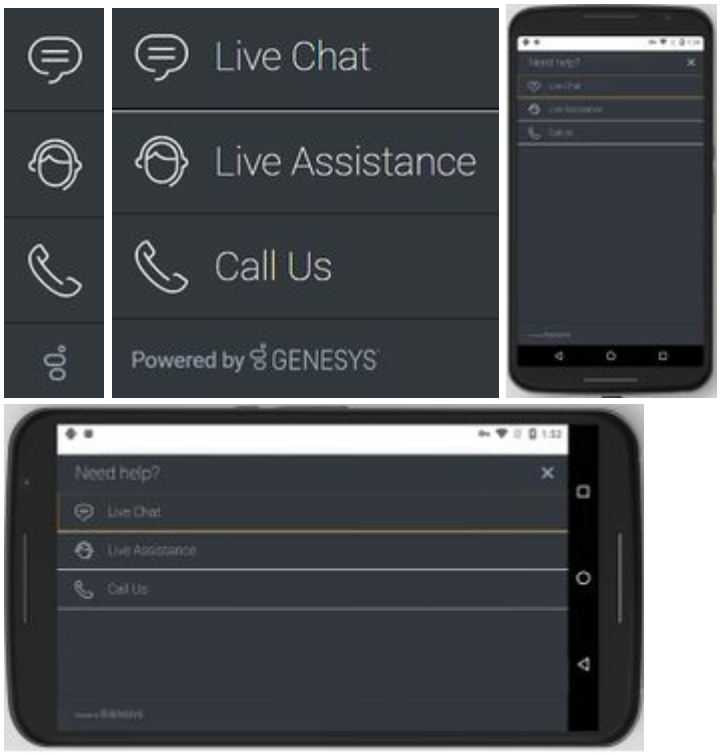
Mobile support

SideBar supports both desktop and mobile devices. In mobile mode, the SideBar launcher button displays at the bottom of the screen. When triggered, it expands to the full screen of the mobile device and shows all channels configured with a scrollbar when necessary. Like all Genesys Widgets, there are two main modes: desktop and mobile. Desktop is for monitors, laptops, and tablets, and mobile is for smartphones. When a smartphone is detected, SideBar switches to special full-screen templates that are optimized for both portrait and landscape orientations.

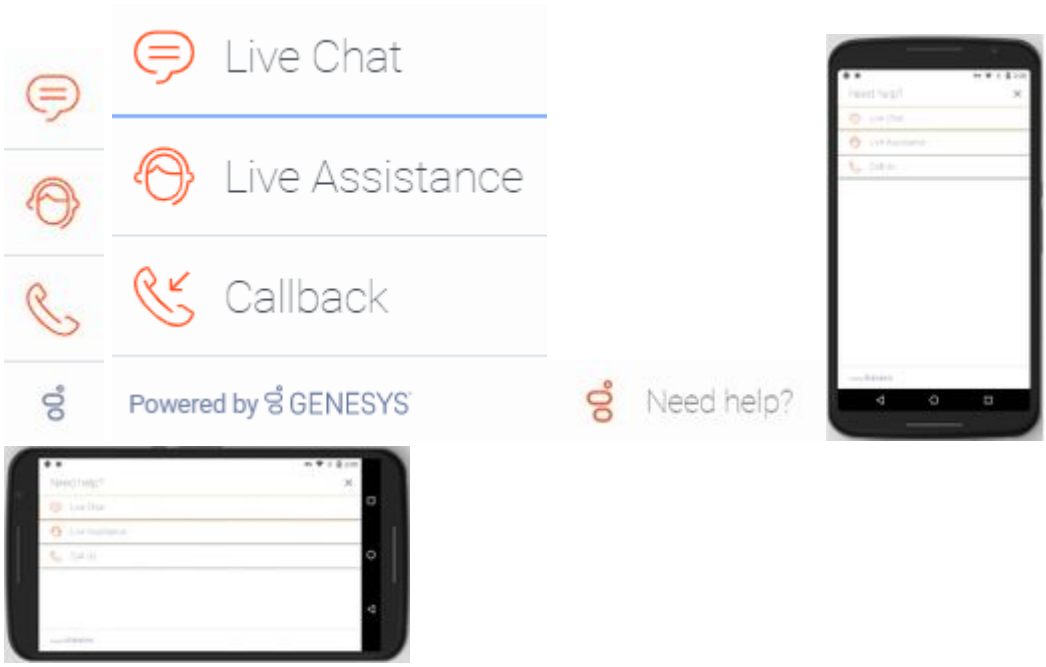
Switching between desktop and mobile mode is done automatically by default. You may configure Genesys Widgets to switch between desktop and mobile mode manually if necessary.

Screenshots

Dark theme


Live Chat
Live Assistance
Call Us
Powered by GENESYS
Need help?
Live Chat
Live Assistance
Call Us

Light theme


Live Chat
Live Assistance
Callback
Powered by GENESYS
Need help?

Configuration

SideBar shares the **_genesys.widgets.sidebar** configuration namespace. SideBar has UI options to handle its position on the screen, disable the expand feature, hide SideBar, and add new channels on the fly. The order of channels that display is based on the order defined in the channel's configuration array.

Example

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

        showOnStartup: true,

        position: 'left',

        expandOnHover: true,

        channels: [{

                        name: 'ChannelSelector',
                        clickCommand: 'ChannelSelector.open',
                        clickOptions: {},

                         //use your own static string or i18n query string for the below two
display properties
                        displayName: 'Live Assist',
                        displayTitle: 'Get live help',

                        icon: 'agent'
                },

                {
                        name: 'WebChat'
                }
        ]
};
```

Options

Name	Type	Description	Default	Required
channels[index].clickCommand	string	Change the default command that is triggered when clicked.	n/a	false
channels[index].clickOptions	object	Pass valid command options that are used in above click command execution.	n/a	n/a
channels[index].displayName	string or i18n query string	Change the default display name for this channel with your own static	n/a	false

		string or to achieve localization, use i18n query string. Syntax: @i18n:.		
channels[index].displayTitle	string or i18n query string	Change the default tooltip content for this channel with your own static string or to achieve localization, use i18n query string. Syntax: @i18n:.	n/a	false
channels[index].icon	string	Change the default icon for this channel. For the list of icon names see **Customize icons** in Customize appearance.	n/a	false
channels[index].name	string	Name of the channel. It can be found in the namespace section documentation of each widget. Used to identify official channels vs custom channels. If a reserved name is used here, SideBar will apply default values for that channel. A plugin name defined in the new custom plugin can also be given here. To override the default values or when defining a new custom channel/plugin, use the below following properties.	n/a	true
channels[index].onClick	function	Define a custom onclick function; this overrides clickCommand and clickOptions.	n/a	false
channels[index].readyEvent	string	Subscribes to this	n/a	false

		ready event published by a plugin.		
expandOnHover	boolean	Enables the expand (slide-out) or contract (slide-in) behavior of SideBar.	true	false
position	string	Defines the position of SideBar on the screen. Acceptable values are `left` or `right`.	right	false
showOnStartup	boolean	Shows the SideBar on the screen when Widgets is launched.	true	false

Localization

For your custom plugins, you can define string key names and values for Name and Title (tooltip) to display on SideBar. The key format requires the plugin name, followed by "Title" or "Name". For example, a plugin named "MyPlugin" will have keys called "MyPluginName" and "MyPluginTitle".

> **Important**
> For information on how to set up localization, refer to the Localize widgets and services guide.

Strings

```
{
        "SidebarTitle": "Need help?",
        "ChannelSelectorName": "Live Assistance",
        "ChannelSelectorTitle": "Get assistance from one of our agents right away",
        "CallUsName": "Call Us",
        "CallUsTitle": "Call Us details",
        "WebChatName": "Live Chat",
        "WebChatTitle": "Live Chat",
        "AriaClose": "Close the menu Need help"
}
```

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('SideBar.open');
```

configure

Internal use only. The main App plugin shares configuration settings to widgets using each widget's configure command. The SideBar Widget has to be configured at minimum with one channel. The `configure` command can also be called at runtime with a new configuration, which will override the existing configuration, showing new channels on screen.

Example

```
oMyPlugin.command('SideBar.configure', {

        showOnStartup: false,
        position: 'left',
        expandOnHover: false,
        channels: [
                {
                        name: 'ChannelSelector',
                        clickCommand: 'ChannelSelector.open',
                        clickOptions: {},

                          /* use your own static string or i18n query string for the below
two display properties. Example for i18n query string: '@i18n:sidebar.ChannelSelectorName'
where 'sidebar' refers to plugin namespace and ChannelSelectorName' name refers to the
property key containing the actual text.*/

                        displayName: '@i18n:sidebar.ChannelSelectorName',
                        displayTitle: 'Get assistance from one of our agents right away', //
Your own static string
                        readyEvent: 'ChannelSelector.ready',
                        icon: 'agent',
                        onClick: function($, CXBus, Common) {
                                _genesys.widgets.bus.command('MyPlugin.open');
                        }
                }
                ...
        ]

}).done(function(e){

        // Sidebar configured successfully
```

```
}).fail(function(e){

        // Sidebar failed to configure properly
});
```

Options

Option	Type	Description
showOnStartup	boolean	Shows SideBar on the screen when Widgets is launched.
position	string	Defines the position of SideBar on the screen.
expandOnHover	boolean	Enables the expand or contract behavior of SideBar.
channels	array	Array containing each channel configuration object. The order of channels is displayed based on the order defined here.
channels[index].name	string	Name of the channel. It can be found in the *Namespace* section documentation of each widget. Used to identify official channels vs custom channels. If a reserved name is used here, SideBar will apply default values for that channel. To override the default values or when defining a new custom channel, use the below following properties.
channels[index].clickCommand	string	Change the default command that is triggered when clicked.
channels[index].clickOptions	object	Pass valid command options that are used in above click command execution.
channels[index].displayName	string or i18n query string	Change the default display name for this channel with your own static string or to achieve localization, use i18n query string. Syntax: @i18n:..
channels[index].displayTitle	string or i18n query string	Change the default tooltip content for this channel with your own static string or to achieve localization, use i18n query string. Syntax: @i18n:..
channels[index].readyEvent	string	Subscribes to this ready event published by a plugin.
channels[index].icon	string	Change the default icon for this channel. For the list of icon names, see *Customize icons* in the Customize appearance guide.

channels[index].onClick	function	Define a custom onclick function, which overrides clickCommand and clickOptions.

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('SideBar.ready', function(e){ /* sample code */ });
```

Name	Description	Data
ready	SideBar is initialized and ready to accept commands.	n/a
opened	SideBar Widget has appeared on screen. For desktop, it displays on the sides of the screen, and in mobile, at the bottom corner as a button.	n/a
closed	SideBar Widget has been removed from the screen.	n/a
expanded	SideBar Widget has expanded, showing channel icon and name.	n/a
contracted	SideBar Widget has contracted, showing channel icons only.	n/a

Resolutions

Status	When	Returns
resolved	Configuration options are provided and set	n/a
rejected	No configuration options are provided	'Invalid configuration. Please ensure at least one channel is configured.'

open

Opens the SideBar UI. In desktop mode, it opens as an actual SideBar and shows the configured channels, whereas in mobile it opens as a button at the bottom to start.

Example

```
oMyPlugin.command('SideBar.open');
```

Resolutions

Status	When	Returns
resolved	SideBar is successfully opened	n/a
rejected	SideBar is already opened	'Already opened'

close

Closes the Sidebar UI.

Example

```
oMyPlugin.command('SideBar.close');
```

Resolutions

Status	When	Returns
resolved	SideBar is successfully closed	n/a
rejected	SideBar is already closed	'already closed'

expand

To show more details about the channels, SideBar slides out from the sides of the screen on desktop machines, but expands to full screen in mobile devices.

Example

```
oMyPlugin.command('SideBar.expand');
```

Resolutions

Status	When	Returns
rejected	SideBar is already expanded	'sidebar already expanded'
resolved	SideBar is successfully expanded	n/a

contract

Retracts the expanded version of SideBar, showing only the channel buttons on desktop machines and the SideBar launcher button on mobile devices.

Example

```
oMyPlugin.command('SideBar.contract');
```

Resolutions

Status	When	Returns
resolved	SideBar is successfully contracted	n/a
rejected	SideBar is already contracted	sidebar already contracted

WebChat

Contents

Learn how to enable live chats between customers and agents in Genesys Cloud CX.

Related documentation:

-

Feature coming soon: Web messaging

If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat. To use web messaging, you configure tracking through the Messenger JavaScript SDK instead of deploying a tracking snippet.

Overview

The WebChat Widget allows a customer to start a live chat with a customer service agent. The UI appears within the page and follows the customer as she explores your website. Other features include minimize/maximize, auto-reconnect, and a built-in invite feature.

Usage

You can launch WebChat manually by using the following methods:

- Call the WebChat.open command
- Configure ChannelSelector to show WebChat as a channel
- Enable the built-in launcher button for WebChat that appears on the right side of the screen
- Create your own custom button or link to open WebChat (using the WebChat.open command)

Customization

You can customize and localize all of the static text shown in the WebChat Widget by adding entries to your configuration and localization options.

WebChat also supports themes. You can create and register your own themes for Genesys Widgets.

Namespace

The WebChat plugin has the following namespaces:

Type	Namespace
Configuration	webchat

Type	Namespace
i18n - Localization	webchat
CXBus - API commands & API events	WebChat
CSS	.cx-webchat

Mobile support

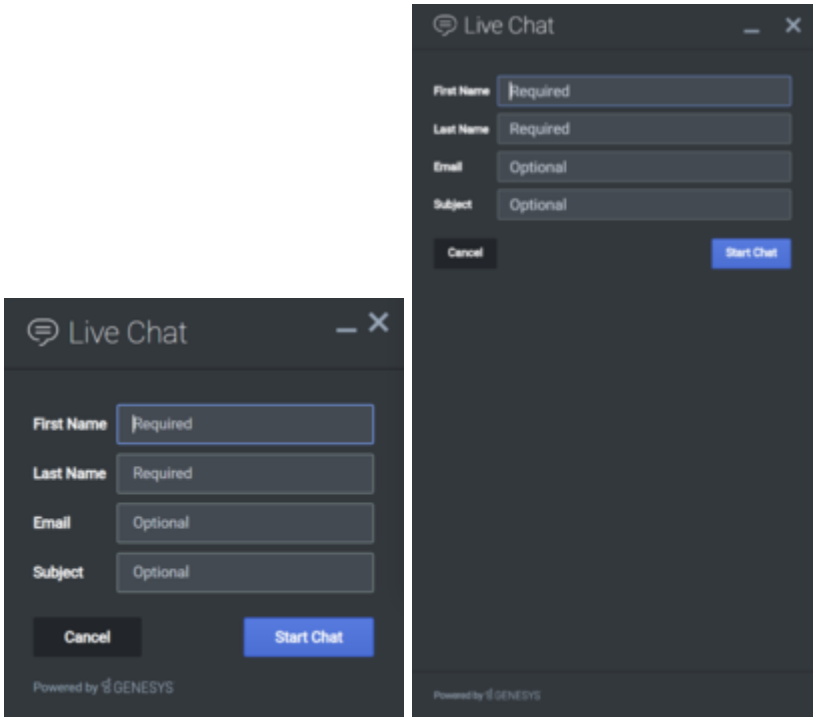
WebChat supports both desktop and mobile devices. Like all Genesys Widgets, there are two main modes: desktop and mobile. Desktop is employed for monitors, laptops, and tablets, and mobile is employed for smartphones. When a smartphone is detected, WebChat switches to special full-screen templates that are optimized for both portrait and landscape orientations.

Switching between desktop and mobile mode is done automatically by default. You may configure Genesys Widgets to switch between desktop and mobile mode manually if necessary.

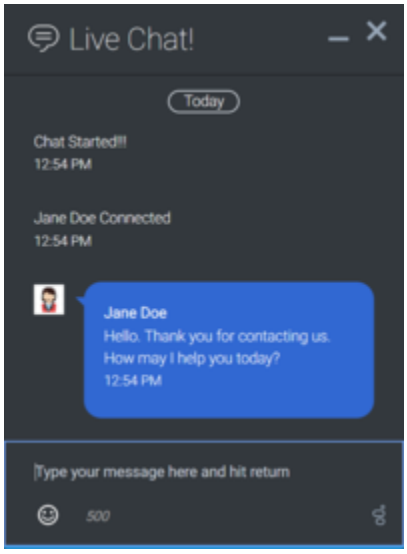
Screenshots

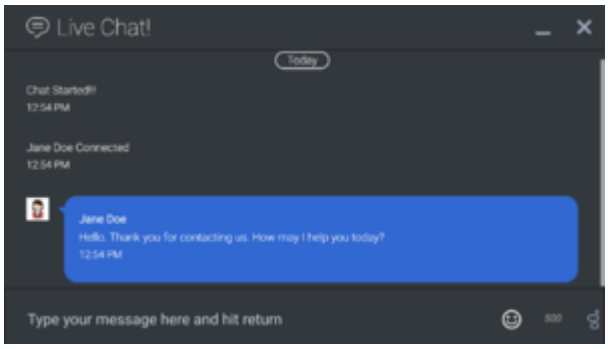
Dark theme

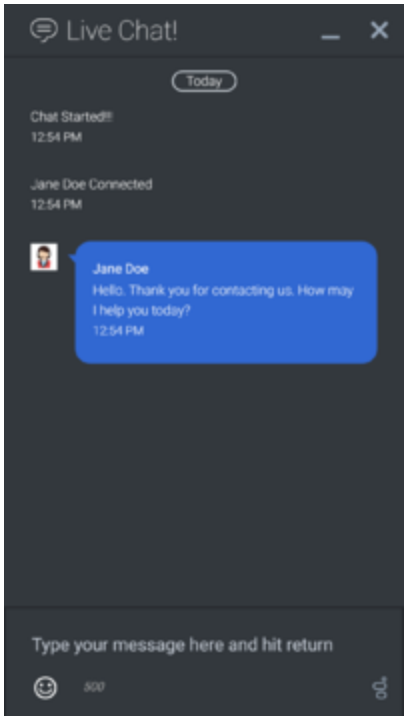
WebChat forms



WebChat transcripts

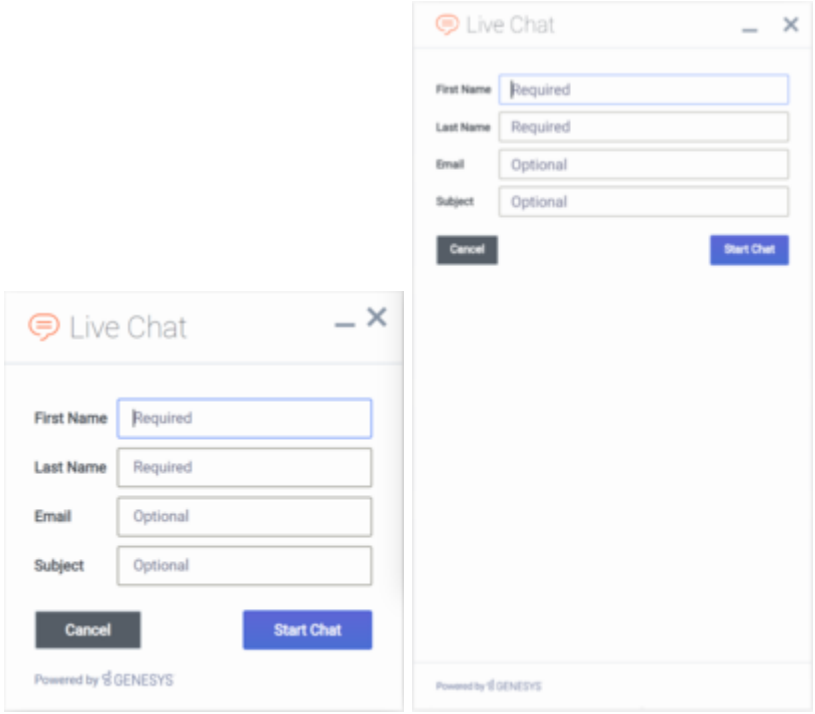


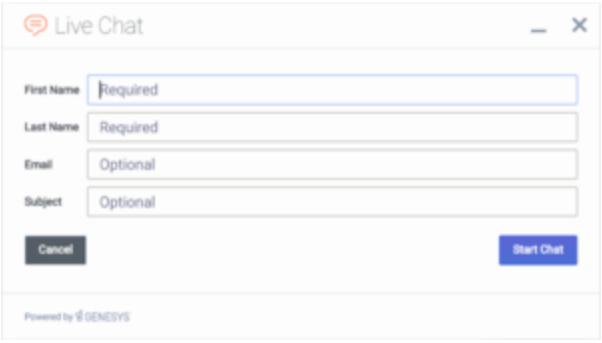




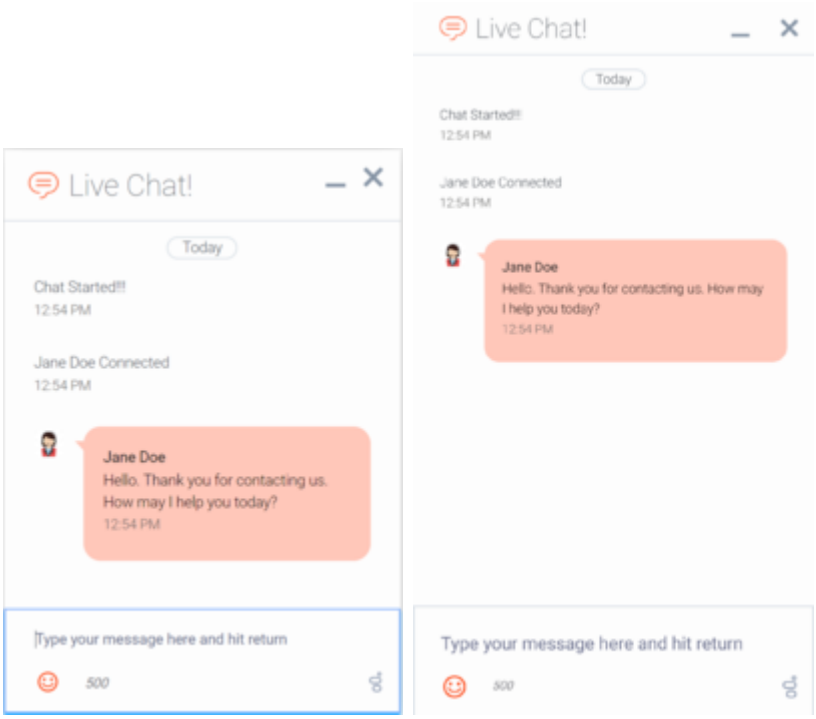
Light theme

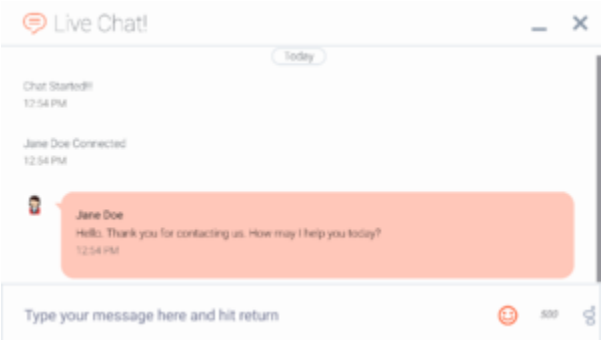
WebChat forms





WebChat transcripts





> **Important**
> The dark theme is active by default. You may also change colors/themes for widgets by following the instructions on the Customize appearance page.

Configuration

WebChat and WebChatService share the **_genesys.widgets.webchat** configuration namespace. WebChat has UI options while WebChatService has connection options.

Options

Name	Type	Description	Default	Required	Introduced/ updated
emojis	boolean	Enable/disable emoji menu inside chat message input. Emojis are supported using Unicode characters.	false	N/A	
form	object	A JSON object containing a custom registration form definition. The JSON definition placed here becomes the default registration form layout for WebChat. See Customizable chat registration form.	A basic registration form is defined internally by default	N/A	
confirmFormCloseEnabled	boolean	Enable or disable displaying a confirmation message	true	N/A	

Name	Type	Description	Default	Required	Introduced/ updated
		before closing WebChat if information has been entered into the registration form.			
timeFormat	number/string	This sets the time format for the timestamps in this widget. It can be 12 or 24.	12	false	
maxMessageLength	number	Set a character limit that the user can input into the message area during a chat. When the max is reached, user cannot type any more.	500	N/A	
charCountEnabled	boolean	Show/hide the number of characters remaining in the input message area while the user is typing.	false	N/A	
autoInvite.enabled	boolean	Enable/disable auto-invite feature. Automatically invites user to chat after user idles on page for preset time. **Important** When running Widgets in lazy load mode, this option requires that you pre-load the WebChat plugin.	false	N/A	
autoInvite.timeToInviteSec	number	Number of seconds of idle time before inviting	5	N/A	

Name	Type	Description	Default	Required	Introduced/ updated
		customer to chat.			
autoInvite.inviteTimeoutSeconds	number	Number of seconds to wait, after showing invite, before closing chat invite. **Important** When the focus is on the Invite window, the chat invite will not auto close upon the specified timeout. In this scenario, you must click the Close button to manually close the Invite window. This behavior is implemented to support the logical and predictable focus order as recommended by the WCAG 2.4.3:Focus Order.	30	N/A	
chatButton.enabled	boolean	Enable/disable chat button on screen. **Important** When running Widgets in lazy load mode, this option requires that you pre-load the WebChat plugin.	false	N/A	
chatButton.template	string	Custom HTML string template for chat button.		N/A	
chatButton.effect	string	Type of animation effect when revealing chat button: `slide` or `fade`.	fade	N/A	
chatButton.openDelay	number	Number of	1000	N/A	

Name	Type	Description	Default	Required	Introduced/ updated
		milliseconds before displaying chat button on screen.			
chatButton.effectDuration	number	Length of animation effect in milliseconds.	300	N/A	
chatButton.hideDuringInvite	boolean	When the auto-invite feature is activated, the chat button hides. When invite is dismissed, the chat button reveals again.	true	N/A	
minimizeOnMobileRestore	boolean	Enable/disable the minimized state of WebChat on chat restore. Note: This option is only for mobile mode.	false	N/A	
markdown	boolean	Enable/disable the markdown feature for chat messages.	false	N/A	9.0.014.02
ariaCharRemainingArray	array/boolean	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	[50, 25, 10]	N/A	9.0.016.11

Name	Type	Description	Default	Required	Introduced/ updated
		value of false will let the screen reader call out remaining characters for every change.			
metaDataEnabled	boolean	Enable or disable WebChat MetaData.	true	n/a	9.0.017.26

Localization

You can define string key names and values to match the system messages that are received from the chat server. If a customer system message is received as **SYS001** in the message body, WebChat checks to determine if any keys match in the language pack, and then replaces the message body accordingly. **SYS001** is an example format. There are no format restrictions on custom message keys. The purpose of this feature is to allow localization for the user interface and server to be kept in the same file.

Special values for localization

You can inject the special value. When used, the agent's name is rendered in its place at runtime.

Error handling

Customers can define their own error messages in the **Errors** section found in the above WebChat localization. If no error messages are defined, default error messages are used.

> **Important**
> For information on how to set up localization, refer to Localize widgets and services.

Usage

You must use the *webchat* namespace for defining localization strings for the WebChat plugin in your i18n JSON file.

The following example shows how to define new strings for the **en** (English) language. You can use any language codes you wish; there is no standard format. When selecting the active language in your configuration, you must match one of the language codes defined in your i18n JSON file. Please

note that you must only define a language code once in your i18n JSON file. Inside each language object you should define new strings for each widget.

Default i18n JSON

```
 {
        "en": {
                "webchat": {
                        "ChatButton": "Chat",
                        "ChatStarted": "Chat Started",
                        "ChatEnded": "Chat Ended",
                        "AgentNameDefault": "Agent",
                        "AgentConnected": " Connected",
                        "AgentDisconnected": " Disconnected",
                        "BotNameDefault": "Bot",
                        "BotConnected": " Connected",
                        "BotDisconnected": " Disconnected",
                        "AgentTyping": "...",
                        "AriaAgentTyping": "Agent is typing",
                        "AgentUnavailable": "Sorry. There are no agents available. Please
try later.",
                        "ChatTitle": "Live Chat",
                        "ChatEnd": "X",
                        "ChatClose": "X",
                        "ChatMinimize": "Min",
                        "ChatFormFirstName": "First Name",
                        "ChatFormLastName": "Last Name",
                        "ChatFormNickname": "Nickname",
                        "ChatFormEmail": "Email",
                        "ChatFormSubject": "Subject",
                        "ChatFormPlaceholderFirstName": "Required",
                        "ChatFormPlaceholderLastName": "Required",
                        "ChatFormPlaceholderNickname": "Optional",
                        "ChatFormPlaceholderEmail": "Optional",
                        "ChatFormPlaceholderSubject": "Optional",
                        "ChatFormSubmit": "Start Chat",
                        "AriaChatFormSubmit": "Start Chat",
                        "ChatFormCancel": "Cancel",
                        "AriaChatFormCancel": "Cancel Chat",
                        "ChatFormClose": "Close",
                        "ChatInputPlaceholder": "Type your message here",
                        "ChatInputSend": "Send",
                        "AriaChatInputSend": "Send",
                        "ChatEndQuestion": "Are you sure you want to end this chat session?",
                        "ChatEndCancel": "Cancel",
                        "ChatEndConfirm": "End chat",
                        "AriaChatEndCancel": "Cancel",
                        "AriaChatEndConfirm": "End",
                        "ConfirmCloseWindow": "Are you sure you want to close chat?",
                        "ConfirmCloseCancel": "Cancel",
                        "ConfirmCloseConfirm": "Close",
                        "AriaConfirmCloseCancel": "Cancel",
                        "AriaConfirmCloseConfirm": "Close",
                        "ActionsDownload": "Download transcript",
                        "ActionsEmoji": "Send Emoji",
                        "ActionsTransfer": "Transfer",
                        "ActionsInvite": "Invite",
                        "InstructionsTransfer": "Open this link on another device to
transfer your chat session>",
                        "InstructionsInvite": "Share this link with another person to add
them to this chat session",
```

```
                        "InviteTitle": "Need help?",
                        "InviteBody": "Let us know if we can help out.",
                        "InviteReject": "No thanks",
                        "InviteAccept": "Start chat",
                        "AriaInviteAccept": "Accept invite and start chat",
                        "AriaInviteReject": "Reject invite",
                        "ChatError": "There was a problem starting the chat session. Please
retry.",
                        "ChatErrorButton": "OK",
                        "AriaChatErrorButton": "OK",
                        "ChatErrorPrimaryButton": "Yes",
                        "ChatErrorDefaultButton": "No",
                        "AriaChatErrorPrimaryButton": "Yes",
                        "AriaChatErrorDefaultButton": "No",
                        "RestoreTimeoutTitle": "Chat ended",
                        "RestoreTimeoutBody": "Your previous chat session has timed out.
Would you like to start a new one?",
                        "RestoreTimeoutReject": "No thanks",
                        "RestoreTimeoutAccept": "Start chat",
                        "AriaRestoreTimeoutAccept": "Accept invite and start chat",
                        "AriaRestoreTimeoutReject": "Reject invite",
                        "EndConfirmBody": "Would you really like to end your chat session?",
                        "EndConfirmAccept": "End chat",
                        "EndConfirmReject": "Cancel",
                        "AriaEndConfirmAccept": "End chat",
                        "AriaEndConfirmReject": "Cancel",
                        "SurveyOfferQuestion": "Would you like to participate in a survey?",
                        "ShowSurveyAccept": "Yes",
                        "ShowSurveyReject": "No",
                        "AriaShowSurveyAccept": "Yes",
                        "AriaShowSurveyReject": "No",
                        "UnreadMessagesTitle": "unread",
                        "AriaYouSaid": "You said",
                        "AriaSaid": "said",
                        "AriaSystemSaid": "System said",
                        "AriaWindowLabel": "Live Chat Window",
                        "AriaMinimize": "Live Chat Minimize",
                        "AriaMaximize": "Live Chat Maximize",
                        "AriaClose": "Live Chat Close",
                        "AriaEmojiStatusOpen": "Emoji picker dialog is opened",
                        "AriaEmojiStatusClose": "Emoji picker dialog is closed",
                        "AriaEmoji": "emoji",
                        "AriaEmojiPicker": "Emoji Picker",
                        "AriaCharRemaining": "Characters remaining",
                        "AriaMessageInput": "Message box",
                        "DayLabels": [
                                "Sun",
                                "Mon",
                                "Tue",
                                "Wed",
                                "Thur",
                                "Fri",
                                "Sat"
                        ],
                        "MonthLabels": [
                                "Jan",
                                "Feb",
                                "Mar",
                                "Apr",
                                "May",
                                "Jun",
                                "Jul",
                                "Aug",
```

```
                                "Sept",
                                "Oct",
                                "Nov",
                                "Dec"
                        ],
                        "todayLabel": "Today",
                        "Errors": {
                                "204": "We're sorry but your message is too long. Please
write a shorter message.",
                                "240": "We're sorry but we cannot start a new chat at this
time. Please try again later.",
                                "401": "We're sorry but we are not able to authorize the chat
session. Would you like to start a new chat?",
                                "404": "We're sorry but we cannot find your previous chat
session. Would you like to start a new chat?",
                                "500": "We're sorry, an unexpected error occurred with the
service. Would you like to close and start a new Chat?",
                                "503": "We're sorry, the service is currently unavailable or
busy. Would you like to close and start a new Chat again?",
                                "ChatUnavailable": "We're sorry but we cannot start a new
chat at this time. Please try again later.",
                                "CriticalFault": "Your chat session has ended unexpectedly
due to an unknown issue. We apologize for the inconvenience.",
                                "StartFailed": "There was an issue starting your chat
session. Please verify your connection and that you submitted all required information
properly, then try again.",
                                "MessageFailed": "Your message was not received successfully.
Please try again.",
                                "RestoreFailed": "We're sorry but we were unable to restore
your chat session due to an unknown error.",
                                "InviteFailed": "Unable to generate invite at this time.
Please try again later.",
                                "Disconnected": "
                                        Connection lost
",
                                "Reconnected": "
                                     Connection restored
",
                                "Generic": "
                                 An unexpected error occurred.
",
                                "purecloud-v2-sockets-400": "Sorry, something went wrong.
Please verify your connection and that you submitted all required information properly, then
try again.",
                                "purecloud-v2-sockets-500": "We're are sorry, an unexpected
error occurred with the service.",
                                "purecloud-v2-sockets-503": "We're sorry, the service is
currently unavailable."
 } } } }
```

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('WebChat.open');
```

configure

Internal use only. The main App plugin shares configuration settings to widgets using each widget's `configure` command. The `configure` command can only be called once at startup. Calling `configure` again after startup may result in unpredictable behavior.

open

Opens the WebChat UI.

Example

```
oMyPlugin.command('WebChat.open', {

        userData: {},
        form: {

                autoSubmit: false,
                firstname: 'John',
                lastname: 'Smith',
                email: 'John@mail.com',
                subject: 'Customer Satisfaction'
        }
        formJSON: {...}

}).done(function(e){

        // WebChat opened successfully

}).fail(function(e){

        // WebChat isn't open or no active chat session
});
```

Options

Option	Type	Description
form	object	Object containing form data to prefill in the chat entry form and optionally auto-submit the form.
form.autoSubmit	boolean	Automatically submit the form. Useful for bypassing the entry

Option	Type	Description
		form step.
form.firstname	string	Value for the first name entry field.
form.lastname	string	Value for the last name entry field.
form.email	string	Value for the email entry field.
form.subject	string	Value for the subject entry field.
formJSON	object	An object containing a custom registration form definition. See Customizable chat registration form.
userData	object	Object containing arbitrary data that gets sent to the server. Overrides userData set in the webchat configuration object.

Resolutions

Status	When	Returns
resolved	WebChat is successfully opened	N/A
rejected	WebChat is already open	*already opened*

close

Closes the WebChat UI.

Example

```
oMyPlugin.command('WebChat.close').done(function(e){

        // WebChat closed successfully

}).fail(function(e){

        // WebChat is already closed or no active chat session
});
```

Resolutions

Status	When	Returns
resolved	WebChat is successfully closed	N/A
rejected	WebChat is already closed	*already closed*

minimize

Minimizes or un-minimizes the WebChat UI.

Example

```
oMyPlugin.command('WebChat.minimize').done(function(e){

        // WebChat minimized successfully

}).fail(function(e){

        // WebChat ignores command
});
```

Options

Option	Type	Description
minimized	boolean	Rather than toggling the current minimized state, you can specify the minimized state directly: true = minimized, false = un-minimized.

Resolutions

Status	When	Returns
resolved	Always	N/A
rejected	Never	*Invalid configuration*

endChat

Starts the **end chat** procedure. User may be prompted to confirm.

Example

```
oMyPlugin.command('WebChat.endChat').done(function(e){

        // WebChat ended a chat successfully

}).fail(function(e){

        // WebChat has no active chat session
});
```

Resolutions

Status	When	Returns
resolved	There is an active chat session to end	N/A
rejected	There is no active chat session to end	*There is no active chat session to end*

invite

Shows an invitation to chat using the toaster popup element. The text shown in the invitation can be edited in the localization file.

Example

```
oMyPlugin.command('WebChat.invite').done(function(e){

        // WebChat invited successfully

}).fail(function(e){

        // WebChat is already open and will be ignored
});
```

Resolutions

Status	When	Returns
resolved	WebChat is closed and the toast element is created successfully	N/A
rejected	WebChat is already open (prevents inviting a user that is already in a chat)	*Chat is already open. Ignoring invite command.*

reInvite

When an active chat session cannot be restored, this invitation offers to start a new chat for the user. The text shown in the invitation can be edited in the localization file.

Example

```
oMyPlugin.command('WebChat.reInvite').done(function(e){

        // WebChat reinvited successfully

}).fail(function(e){

        // WebChat is already open and will be ignored
});
```

Resolutions

Status	When	Returns
resolved	WebChat is closed, the config item **webchat.inviteOnRestoreTimeout** is set, and the toast element is created successfully	N/A
rejected	WebChat is already open (prevents inviting a user that is already in a chat)	*Chat is already open. Ignoring invite command.*

injectMessage

Injects a custom message into the chat transcript. Useful for extending WebChat functionality with other Genesys products.

Example

```
 oMyPlugin.command('WebChat.injectMessage', {

         type: 'text',
         name: 'person',
         text: 'hello',
         custom: false,
         bubble:{

                 fill: '#00FF00',
                 radius: '4px',
                 time: false,
                 name: false,
                 direction: 'right',
                 avatar:{

                         custom: '
word
',

                         icon: 'email'
                 }
         }

 }).done(function(e){

         // WebChat injected a message successfully
         // e.data == The message HTML reference (jQuery wrapped set)

 }).fail(function(e){

         // WebChat isn't open or no active chat
 });
```

Options

Option	Type	Description
type	string	Switch the rendering type of the injected message between text and HTML.
name	string	Specify a name label for the message to identify what service or widget has injected the message.
text	string	The content of the message. Either plain text or HTML.
custom	boolean	If set to true, the default message template will not be used, allowing you to inject a highly customized HTML block unconstrained by the normal

Option	Type	Description
		message template.
bubble.fill	string of valid CSS color value	The content of the message. Either plain text or HTML.
bubble.radius	string of valid CSS border radius vale	The border radius you'd like for the bubble.
bubble.time	boolean	If you'd like to show the timestamp for the bubble.
bubble.name	boolean	If you'd like to show the name for the bubble.
bubble.direction	string	Which direction you want the message bubble to come from.
bubble.avatar.custom	string or HTML reference	Change the content of the HTML that would be the avatar for the chat bubble.
bubble.avatar.icon	class name	Generated common library provided for icon name.

Resolutions

Status	When	Returns
resolved	WebChat is open and there is an active chat session	*An HTML reference (jQuery wrapped set) to the new injected message.*
rejected	WebChat is not open and/or there was no active chat session	*No chat session to inject into.*

showChatButton

Displays the standalone chat button using either the default template and CSS, or customer-defined ones.

Example

```
 oMyPlugin.command('WebChat.showChatButton', {

         openDelay: 1000,
         duration: 1500

 }).done(function(e){

         // WebChat shows chat button successfully

 }).fail(function(e){

         // WebChat button is already visible, side bar is active and overrides the chat
button, or chat button is disabled in configuration
 });
```

Options

Option	Type	Description
openDelay	number	Duration in milliseconds to delay showing the chat button on the page.
duration	number	Duration in milliseconds for the show and hide animation.

Resolutions

Status	When	Returns
resolved	The chat button is enabled in the configuration, is currently not visible, and the SideBar plugin is not initialized	N/A
rejected	The chat button is not enabled in the configuration, or it's already visible, or the SideBar plugin is initialized	*Chat button is already visible. Ignoring command.*
rejected	The SideBar plugin is active, the standalone chat button will be disabled automatically	*SideBar is active and overrides the default chat button*

hideChatButton

Hides the standalone chat button.

Example

```
oMyPlugin.command('WebChat.hideChatButton', {duration: 1500}).done(function(e){

        // WebChat hid chat button successfully

}).fail(function(e){

        // WebChat button is already hidden
});
```

Options

Option	Type	Description
duration	number	Duration in milliseconds for the show and hide animation.

Resolutions

Status	When	Returns
resolved	The chat button is currently	N/A

Status	When	Returns
	visible	
rejected	The chat button is already hidden	*Chat button is already hidden. Ignoring command.*

showOverlay

Opens a slide-down overlay over WebChat's content. You can fill this overlay with content such as disclaimers, articles, and other information.

Example

```
 oMyPlugin.command('WebChat.showOverlay', {

         html: '
Example text
',
         hideFooter: false

 }).done(function(e){

         // WebChat successfully shows overlay

 }).fail(function(e){

         // WebChat isn't open
 });
```

Options

Option	Type	Description
html	string or HTML reference	The HTML content you want to display in the overlay. **Important** The `id` attribute value of the HTML content can be set to `cx_chat_information`. This supports a screen reader's ability to announce the overlay's content to the user, as recommended by WCAG.
hideFooter	boolean	Normally the overlay appears between the title bar and footer bar. Set this to true to have the overlay overlap the footer to gain a bit more vertical space. This should only be used in special cases. For general use, don't set this value.

Resolutions

Status	When	Returns
resolved	WebChat is open and the overlay opens	N/A
rejected	WebChat is not currently open	*WebChat is not currently open. Ignoring command.*

hideOverlay

Hides the slide-down overlay.

Example

```
oMyPlugin.command('WebChat.hideOverlay').done(function(e){

        // WebChat hid overlay successfully

}).fail(function(e){

        // WebChat isn't open
});
```

Resolutions

Status	When	Returns
resolved	WebChat is open and the overlay closes	N/A
rejected	WebChat is not currently open	*WebChat is not currently open. Ignoring command.*

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');
oMyPlugin.subscribe('WebChat.ready', function(e){});
```

Name	Description	Data
ready	WebChat is initialized and ready to accept commands.	N/A
opened	The WebChat widget has appeared on screen.	N/A
started	The WebChat has successfully started.	Metadata
submitted	The user has submitted the form.	Metadata
rejected	When the chat session fails to start. Typically due to form validation or network errors.	Metadata
completed	The chat session ended after the agent is successfully connected to WebChat.	Metadata
cancelled	The chat session ended before the agent is connected to WebChat.	Metadata
closed	The WebChat Widget has been removed from the screen.	Metadata
minimized	The WebChat Widget has been changed to a minimized state.	N/A
unminimized	The WebChat Widget has been restored from a minimized state to the standard view.	N/A
messageAdded	When a message is added to the transcript, this event will fire.	Returns an object containing two properties: *data* and *html*; *data* contains the JSON data for the message, while *html* contains a reference to the visible message inside the chat transcript.

Metadata

Interaction Lifecycle

Every WebChat interaction has a sequence of events we call the *Interaction Lifecycle*. This is a sequence of events that tracks progress and choices from the beginning of an interaction (opening WebChat), to the end (closing WebChat), and every step in between.

The following events are part of the Interaction Lifecycle:

```
ready
opened
started
cancelled
submitted
rejected
completed
```

```
closed
```

Lifecycle scenarios

An Interaction Lifecycle can vary based on each user's intent and experience with WebChat. Here are several sequences of events in the lifecycle that correspond to different scenarios.

The user opened WebChat but changed their mind and closed it without starting a chat session:

```
ready -> opened -> cancelled -> closed
```

The user started a chat session but ended it before an agent connected. Perhaps it was taking too long to reach someone:

```
ready -> opened -> started -> cancelled -> closed
```

The user started a chat, but the chat fails to start:

```
ready -> opened -> started -> submitted -> rejected
```

The user started a chat, met with an agent, and the session ended normally:

```
ready -> opened -> started -> submitted -> completed -> closed
```

> **Tip**
> For a list of all WebChat events, see API events.

Metadata

Each event in the Interaction Lifecycle includes the following block of metadata. By default, all values are set to false. As the user progresses through the lifecycle of a WebChat interaction, these values will be updated.

The metadata block contains boolean state flags, counters, timestamps, and elapsed times. These values can be used to track and identify trends or issues with chat interactions. During run-time, the metadata can help you offer a smart and dynamic experience to your users.

Reference

Name	Type	Description
proactive	boolean	Indicates this chat session was started proactively.
prefilled	boolean	Indicates the registration form was prefilled with info automatically.
autoSubmitted	boolean	Indicates the registration form was submitted automatically, usually after being prefilled.

Name	Type	Description
numAgents	integer	Current number of agents that have connected to the chat session.
userMessages	integer	Current number of messages sent by user.
agentMessages	integer	Current number of messages sent by agents.
systemMessages	integer	Current number of system messages received.
errors	array/boolean	An array of error codes encountered during a chat session. If no errors, this value will be `false`.
form	object	An object containing the form parameters when the form is submitted.
opened	integer (timestamp)	Timestamp indicating when WebChat was opened.
started	integer (timestamp)	Timestamp indicating when chat session started.
cancelled	integer (timestamp)	Timestamp indicating when the chat session was cancelled. Cancelled refers to when a user ends a chat session before an agent connects.
rejected	integer (timestamp)	Timestamp indicating when the chat session was rejected. Rejected refers to when a chat session fails to start.
completed	integer (timestamp)	Timestamp indicating when the chat session ended normally. Completed refers to when a user or agent ends a chat after an agent connected.
closed	integer (timestamp)	Timestamp indicating when WebChat was closed.
agentReached	integer (timestamp)	Timestamp indicating when the first agent was reached, if any.
elapsed	integer (milliseconds)	Total elapsed time in milliseconds from when the user started the chat session to when the chat session ended.
waitingForAgent	integer (milliseconds)	Total time in milliseconds waiting for an agent from when the user started the chat session to when an agent connected to the session.
id	string	A unique identifier of a chat

Name	Type	Description
		session that helps to identify the instance of that session and its associated events.

Customizable chat registration form

WebChat allows you to customize the registration form shown to users prior to starting a session. The following form inputs are currently supported:

- Text
- Select
- Hidden
- Checkbox
- Textarea

Customization is done through a JSON object structure that defines the layout, input type, label, and attributes for each input. You can set the default registration form definition in the **_genesys.widgets.webchat.form** configuration option. Alternately, you can pass a new registration form definition through the **WebChat.open** command:

```
_genesys.widgets.bus.command("WebChat.open", {formJSON: oRegFormDef});
```

Inputs are rendered as stacked rows with one input and one optional label per row.

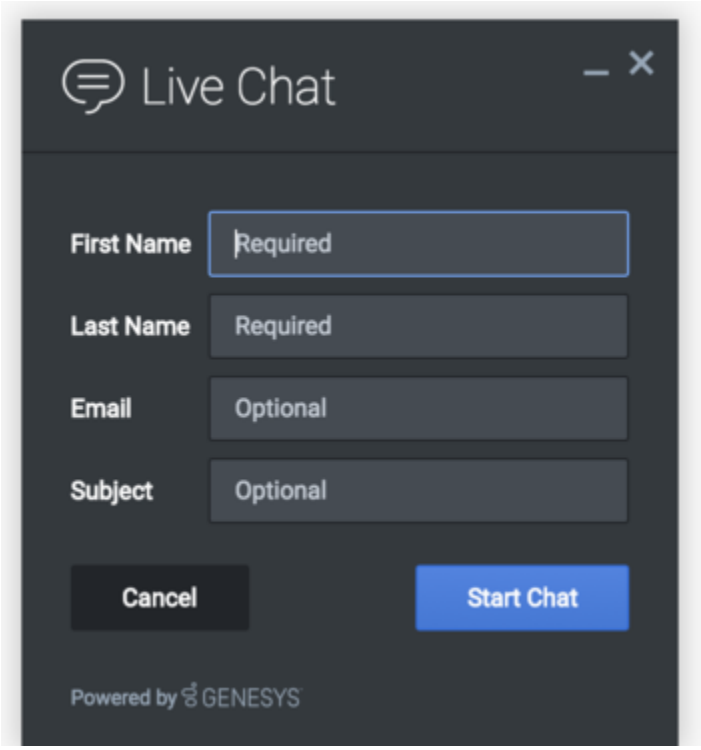
Default example

The following example is the default JSON object used to render WebChat's registration form. This is a very simple definition that does not use many properties.

```
 {
        wrapper: "
",
        inputs: [

                {
                        id: "cx_webchat_form_firstname",
                        name: "firstname",
                        maxlength: "100",
                        placeholder: "@i18n:webchat.ChatFormPlaceholderFirstName",
                        label: "@i18n:webchat.ChatFormFirstName"
                },

                {
                        id: "cx_webchat_form_lastname",
                        name: "lastname",
                        maxlength: "100",
                        placeholder: "@i18n:webchat.ChatFormPlaceholderLastName",
                        label: "@i18n:webchat.ChatFormLastName"
                },
```

```
                {
                        id: "cx_webchat_form_email",
                        name: "email",
                        maxlength: "100",
                        placeholder: "@i18n:webchat.ChatFormPlaceholderEmail",
                        label: "@i18n:webchat.ChatFormEmail"
                },

                {
                        id: "cx_webchat_form_subject",
                        name: "subject",
                        maxlength: "100",
                        placeholder: "@i18n:webchat.ChatFormPlaceholderSubject",
                        label: "@i18n:webchat.ChatFormSubject"
                }
        ]
}
```

This JSON definition generates the following output:



Properties

Each input definition can contain any number of properties. These are categorized in two groups: "Special Properties", which are custom properties used internally to handle rendering logic, and "HTML Attributes" which are properties that are applied directly as HTML attributes on the input element.

Special properties

Property	Type	Default	Description
type	string	"text"	Sets the type of input to render. Possible values are currently "text", "hidden", "select", "checkbox", and "textarea".
label	string	N/A	Set the text for the label. If no value provided, no label will be shown. You may use localization query strings to enable custom localization (for example, label: "@i18n:namespace.StringName"). Localization query strings allow you to use strings from any widget namespace or to create your own namespace in the localization file (i18n.json) and use strings from there (for example, label: "@i18n:myCustomNamespace.myCusto For more information, see the Labels section.
wrapper	HTML string	“ "	Each input exists in its own row in the form. By default this is a table-row with the label in the left cell and the input in the right cell. You can redefine this wrapper and layout by specifying a new HTML row structure. See the Wrappers section for more info. The default wrapper for an input is "
validate	function	N/A	Define a validation function for the input that executes when the input loses focus (blur) or changes value. Your function must return true or false. True to indicate it passed, false to indicate it failed. If your validation fails, the

Property	Type	Default	Description
			form will not submit and the invalid input will be highlighted in red. See the Validation section for more details and examples.
validateWhileTyping	boolean	false	Execute validation on keypress in addition to blur and change. This ignores non-character keys like shift, ctrl, and alt.
options	array	[]	When ‘type’ is set to ‘select’, you can populate the select by adding options to this array. Each option is an object (for example, {text: ‘Option 1’, value: ‘1’} for a selectable option, and {text: "Group 1", group: true} for an option group).

HTML attributes

With the exception of special properties, all properties will be added as HTML attributes on the input element. You can use standard HTML attributes or make your own.

Example

```
{
        id: "cx_webchat_form_firstname",
        name: "firstname",
        maxlength: "100",
        placeholder: "@i18n:webchat.ChatFormPlaceholderFirstName",
        label: "@i18n:webchat.ChatFormFirstName"
}
```

In this example, **id**, **name**, **maxlength**, and **placeholder** are all standard HTML attributes for the text input element. Whatever values are set here will be applied to the input as HTML attributes.

> **Important**
>
> The default input type is "text", so type does not need to be defined if you intend to make a text input.

HTML output

Disabling autocomplete

Since the custom form feature supports adding any HTML attributes to your inputs, you can control standard HTML features like disabling autocomplete. To disable autocomplete, add **autocomplete: "off"** to your input definition.

Example

```
{
        id: "cx_webchat_form_firstname",
        name: "firstname",
        maxlength: "100",
        placeholder: "@i18n:webchat.ChatFormPlaceholderFirstName",
        label: "@i18n:webchat.ChatFormFirstName",
        autocomplete: "off"
}
```

Labels

A label tag will be generated for your input if you specify label text and if your custom input wrapper includes a ‘{label}’ designation. If you have added an ID attribute for your input, the label will automatically be linked to your input so that clicking on the label selects the input or, for checkboxes, toggles it.

Labels can be defined as static strings or localization queries.

Wrappers

Wrappers are HTML string templates that define a layout. There are two kinds of wrappers: f*orm wrappers* and i*nput wrappers.*

Form wrapper

You can specify the parent wrapper for the overall form in the top-level "wrapper" property. The following example specifies this value as “

". This is the default wrapper for the WebChat form:

```
{
        wrapper: "
", /* form wrapper */
        inputs: []
}
```

Input wrapper

Each input is rendered as a table row inside the form wrapper. You can change this by defining a new wrapper template for your input row. Inside your template, you can specify where you want the input and label to be by adding the identifiers `label` and `input` to your wrapper value. See the example below:

```
{
        id: "cx_webchat_form_firstname",
        name: "firstname",
        maxlength: "100",
        placeholder: "@i18n:webchat.ChatFormPlaceholderFirstName",
        label: "@i18n:webchat.ChatFormFirstName",
        wrapper: "{label}{input}" /* input row wrapper */
}
```

The `label` identifier is optional. Omitting it will allow the input to fill the row. If you decide to keep the label, you can move it to any location within the wrapper, such as putting the label on the right, or stacking the label on top of the input. You can control the layout of each row independently, depending on your needs.

You are not restricted to using a table for your form. You can change the form wrapper to "

" and then change the individual input wrappers from a table-row to your own specification. Be aware though that when you move away from the default table wrappers, you are responsible for styling and aligning your layout. Only the default table-row wrapper is supported by default themes and CSS.

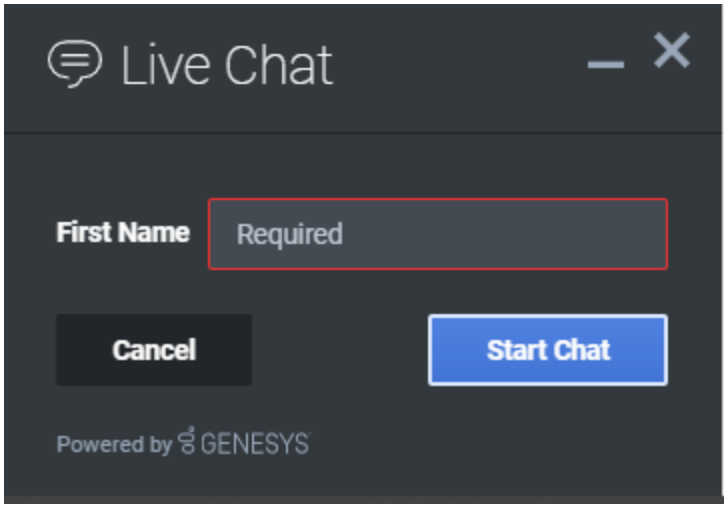
Validation

You can apply a validation function to each input that lets you check the value after a change has been made and/or the user has moved to a different input (on change and on blur). You can enable validation on key press by setting **validateWhileTyping** to `true` in your input definition.

Here is how to define a validation function:

```
{
        id: "cx_webchat_form_firstname",
        name: "firstname",
        maxlength: "100",
        placeholder: "@i18n:webchat.ChatFormPlaceholderFirstName",
        label: "@i18n:webchat.ChatFormFirstName",

        validateWhileTyping: true, // default is false

        validate: function(event, form, input, label, $, CXBus, Common){

                return true; // or false
        }
}
```

You must return `true` or `false` to indicate that validation has passed or failed, respectively. If you return `false`, the WebChat form will not submit, and the input will be highlighted in red. This is achieved by adding the CSS class **cx-error** to the input. The image below displays the the field where a user input validation error has occurred, with the field highlighted in red.



Validation function arguments

Argument	Type	Description
event	JavaScript event object	The input event reference object related to the form input field. This event data can be helpful to perform actions like active validation on an input field while the user is typing.
form	HTML reference	A jquery reference to the form wrapper element.
input	HTML reference	A jquery reference to the input element being validated.
label	HTML reference	A jquery reference to the label for the input being validated.
$	jquery instance	Widget's internal jquery instance. Use this to help you write your validation logic, if needed.
CXBus	CXBus instance	Widget's internal CXBus reference. Use this to call commands on the bus, if needed.
Common	Function Library	Widget's internal Common library of functions and utilities. Use if needed.

Form submit

Custom input field form values are submitted to the server as key value pairs under the userData section of the form submit request, where input field names will be the property keys. During the submit, this data is merged along with the userData defined in the WebChat open command.

> **Important**
>
> Depending on the API used (PureEnagage V2 API or PureCloud) the payload structure in the request can vary for each, but the section below explains how the form data is submitted by the WebChat UI plugin when using custom forms. Below is the internal form data object defined in the WebChat plugin by default. Since firstname, lastname, nickname, email, and subject are reserved keywords, users are not allowed to have custom fields with the same name.

```
{
 firstname: '',
   lastname: '',
   nickname: '',
   email: '',
   subject: '',
   userData: {}
 }
```

Example

The example below shows how the custom form data given in the WebChat form fields have been mapped as form data object.

The form fields with reserved keywords like **firstname**, **lastname**, and **email** will be sent as top level, and the rest of the fields will be sent under userData to the WebChatService plugin.

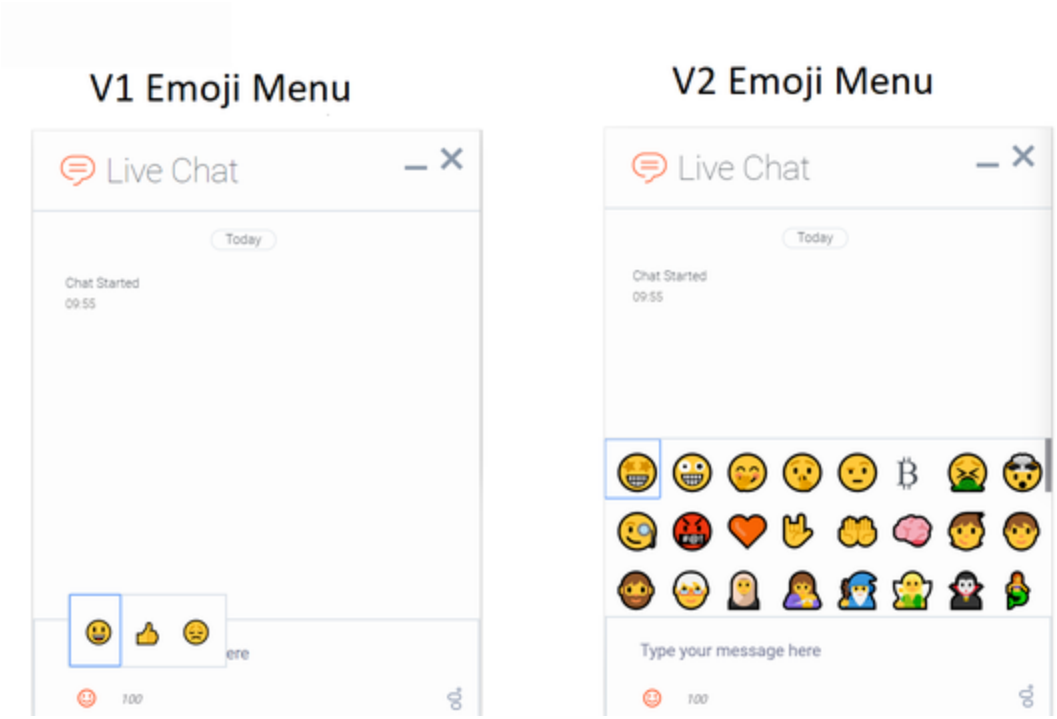
Once the form data object is sent to the WebChatService plugin, it will parse and send in the payload request.

```
 {
   firstname: 'John',
   lastname: 'Smith',
   email: 'john.smith@company.com',
   userData: {
     phonenumber: '9256328346',
     enquirytype: 'Sales' //value selected from the dropdown
   }
 }
```

Customizable emoji menu

Introduction

WebChat offers a v2 emoji menu that lets you choose which emojis to include in the emoji menu.



Differences between v1 and v2

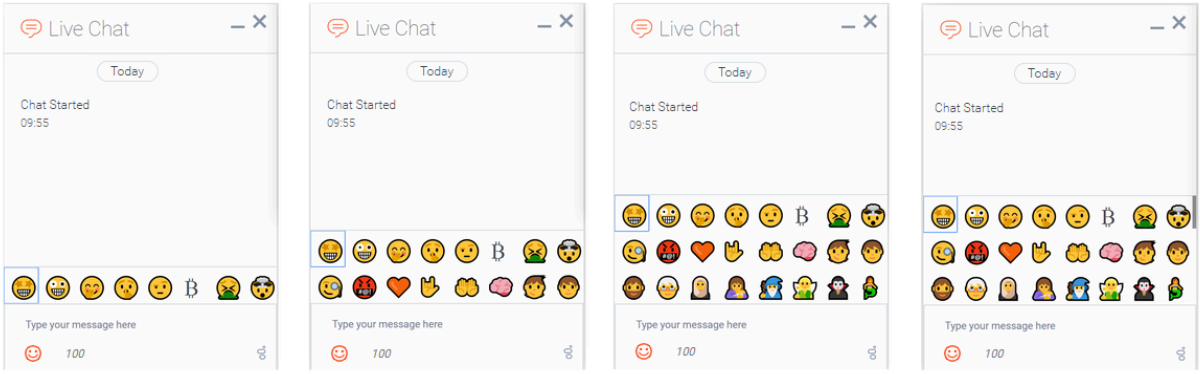
- v1 shows as a tooltip-style overlay; v2 shows as a new block between the transcript and the message input.
- v1 closes when you select an emoji or click outside the menu; v2 lets you choose multiple emojis and only closes if you click the emoji menu button again.
- v1 has three fixed emojis to choose from; v2 can show hundreds of customizable emojis in a grid layout.
- v1 menu appears in mobile mode; v2 menu is not available in mobile mode (when v2 is configured, no emoji menu button is present in mobile mode).
- v1 menu has default emojis; v2 menu does not have default emojis. It must be explicitly configured with a list of emojis.

Configuring the emoji menu

Click the emoji menu icon at the bottom-left corner of the WebChat UI to open the v2 emoji menu. The transcript will be resized to fit the emoji menu, which can vary in height depending on the number of emojis configured:

- When 1-8 emojis are configured, the menu has one row, and no scrollbar appears.
- When 9-16 emojis are configured, the menu has two rows, and no scrollbar appears.
- When 17-24 emojis are configured, the menu has three rows, and no scrollbar appears.

- When 25 or more emojis are configured, the menu has three rows, and a scrollbar appears.



Configure the v2 emoji menu by passing a string containing emoji into the WebChat configuration or through localization.

> **Important**
>
> If you define an emoji list in the WebChat configuration, it will override any emoji lists defined in localization files.

You configure the emoji list by specifying a string of emoji characters, like "🙂😀😁😂". WebChat will parse this string and arrange them in the emoji menu.

```
// Configure a flat list of emoji characters
_genesys.widgets.webchat.emojiList =
"😀😁😂😃😄😅😆😇😈😉😊😋😌😍😎😏😐😑😒😓😔😕😖😗😘😙😚😛😜😝😞😟😠😡😢😣😤😥😦😧😨😩😪😫😬😭😮😯😰😱😲😳😴😵😶😷🙁🙂🙃🙄🤐🤑🤒🤓🤔🤕🤠🤡🤢🤣🤤🤥🤧🤨🤩🤪🤫🤬🤭🤮🤯🧐";
```

Add emoji display names

You can also add names to emojis so that their names will appear when you hover over them. To add a name to an emoji, simply add a colon after the question mark symbol, and then type the name. Separate each name with a semicolon.

The format is ;😀:name;

You can only add one name to an emoji. The following sample shows the format for configuring several emojis.

```
// Configure an emoji list with emoji names
_genesys.widgets.webchat.emojiList = "😀:Star-Struck;😁:Zany Face;😂:Face With Hand Over
 Mouth;😃:Shushing Face;😄:Face With Raised Eyebrow;😅:Bitcoin;😆:Face Vomiting;
😇:Exploding Head;😈:Face With Monocle;😉:Face With Symbols on Mouth;😊:Orange Heart;
😋:Love-You Gesture;😌:Palms Up Together;😍:Brain;😎:Child;😏:Person;😐:Man: Beard;
😑:Older Person;😒:Woman With Headscarf;😓:Breast-Feeding;😔:Mage;😕:Fairy;😖:Vampire;
😗:Merperson;😘:Elf;😙:Genie;😚:Zombie;😛:Person in Steamy Room;😜:Person Climbing;
😝:Person in Lotus Position;😞:Zebra;😟:Giraffe;😠:Hedgehog;😡:Sauropod;😢:T-Rex;😣:Cricket;
```

```
🥥:Coconut;🥦:Broccoli;🥨:Pretzel;🥩:Cut of Meat;🇦🇺:Australia Day;🇫🇷:Bastille
 Day;🎂:Birthday;🛍️:Black Friday;🇨🇦:Canada Day;🎪:Carnival;🧧:Chinese New Year;🎄:Christmas;
🇲🇽:Cinco de Mayo;🪔:Diwali;🐉:Dragon Boat Festival;🐰:Easter;🍿:Emoji Movie;🍂:Fall/Autumn;
👨:Father’s Day;🥫:Festivus;🎓:Graduation;🧨:Guy Fawkes;🎃:Halloween;🕎:Hanukkah;
💘:Hearts;🎨:Holi;🇺🇸:Independence Day;👩:Mother’s Day;🎆:New Year’s Eve;🏅:Olympics;
🏳️‍🌈:Pride;👑:Queen’s Birthday;☪️:Ramadan;🌷:Spring;☘️:St Patrick’s Day;☀️:Summer;
🏈:SuperBowl;🦃:Thanksgiving;💌:Valentine’s Day;💒:Wedding / Marriage;❄️:Winter;⛷️:Winter
 Olympics;🌍:World Cup;🌍:World Emoji Day;";
```

Partially named lists

You don't have to add names for every emoji. You can add titles to only a select few.

```
// Configure an emoji list with only a few emoji names
_genesys.widgets.webchat.emojiList = "🥥🥦🥨🥩🇦🇺🇫🇷;🎂:Palms Up Together;
🎂🛍️🇨🇦🎪🧧🎄🇲🇽🪔🐉🐰🍿🍂;🛍️:Black Friday;
🛍️🎄🇲🇽🪔🐉🐰🍿🍂👨🥫🎓🧨🎃🕎💘🎨🇺🇸👩🎆🏅🏳️‍🌈👑☪️🌷☘️☀️🏈🦃💌💒❄️⛷️🌍🌍;🎂:Snowman;🎂🎂";
```

Localization

Emojis can be localized so that each language has a preferred set of emojis and emoji titles.

> **Important**
>
> If you define an emoji list in the WebChat configuration, it will override any emoji lists defined in localization files.

The key name for defining an emoji list is "EmojiList". Emoji lists are defined in a localization file using the same syntax as the WebChat configuration.

```
{
    "en": {
        "webchat": {
            "EmojiList": "🤩:Star-Struck;🤪:Zany Face;🤭:Face With Hand Over Mouth;🤫:Shushing
Face;"
        }
    }
}
```

Engage

Contents

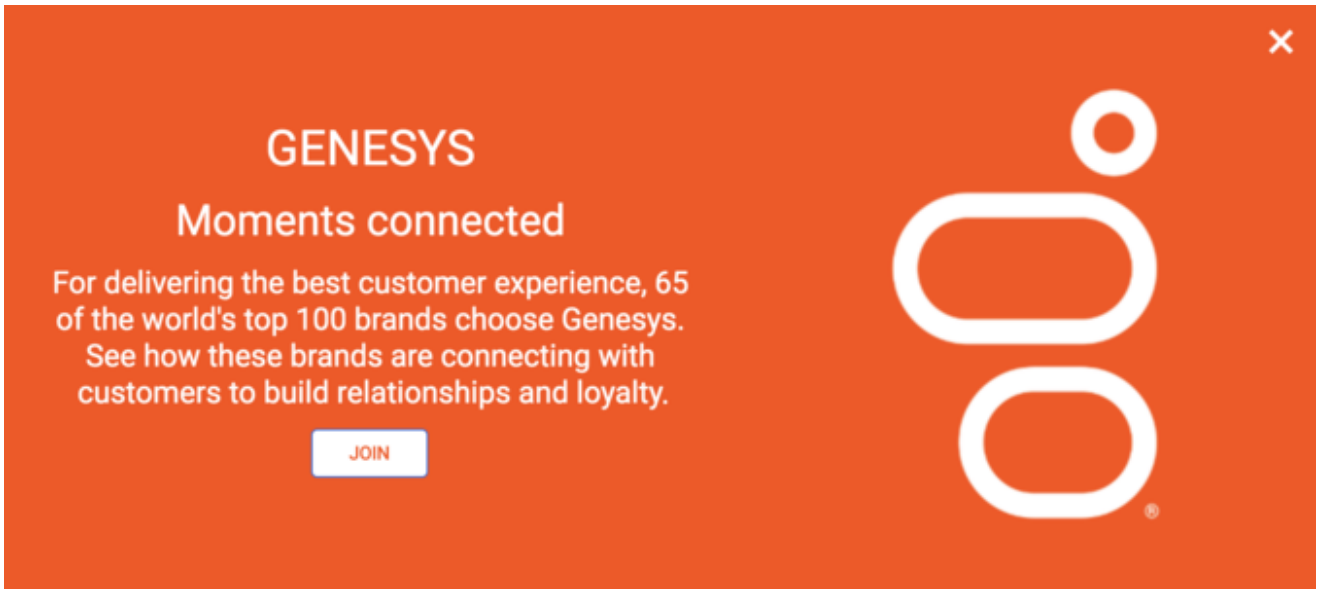
- Developer

Learn how to use the Genesys Multicloud CX plugin to integrate any Engage solution with Genesys Widgets.

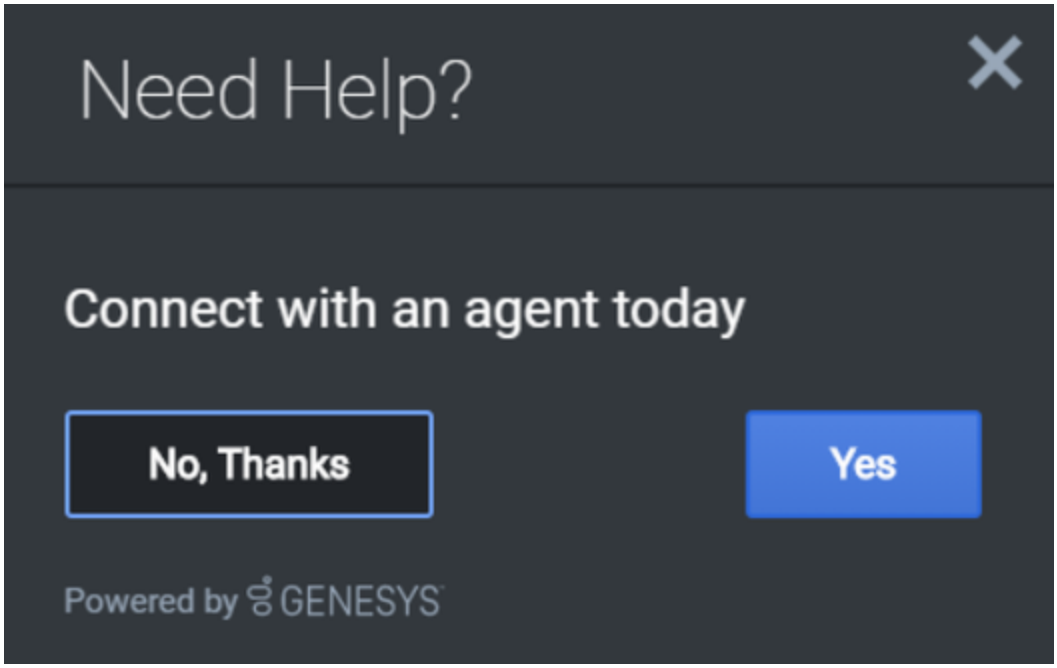
Related documentation:

-

Overview



The Genesys Multicloud CX plugin is generic and contains commands that automate customer engagement within Genesys Widgets. Starting with version 9.0.015.11, the Engage plugin includes Offers, which allows a customer to view a product or promotion on a page. It comes with many display modes and rendering options, such as overlay/toaster mode with text or image-only layouts, or both.



Usage

Use the Engage plugin to show either an invite or an offer via the following methods:

- Calling the Engage.invite command
- Calling the Engage.offer command

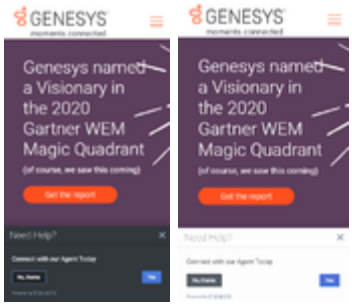
Namespaces

The Engage plugin uses the following namespaces.

Type	Namespace
i18n - Localization	Engage
CXBus - API commands & API events	Engage
CSS	.cx-engage

Screenshots

Engage Invite



Engage Offer

Configuration

The Genesys Multicloud CX plugin doesn't have any configuration options.

Localization

The Genesys Multicloud CX plugin doesn't have any localization options.

API commands

Once you've registered your plugin on the bus, you can call commands on other registered plugins. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing Widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.command('Engage.invite');
```

invite

Opens the Engage Widget and renders the text based on the options provided. If no options are provided, the widget doesn't open.

Example

```
oMyPlugin.command('Engage.invite', {
        'type':'toast',
        'timeout':3000,
        'title':'Engage Title',
        'ariaTitle':'Engage Invite',
        'body':'Engage invite body content',
        'accept':'Yes',
        'decline':'No, thanks',
        'ariaAccept':'Yes',
        'ariaDecline':'No, thanks',
        'ariaClose':'Close',
        'command': 'WebChat.open',
        'options':{'proactive': true, 'userData': {'category': 'shoes'}}
});

oMyPlugin.command('Engage.invite',{
        'type':'toast',
        'timeout':3000,
        'force': true,
        'title':'Engage Title',
        'ariaTitle':'Engage Invite',
        'body':'Engage invite body content',
        'accept':'Yes',
        'decline':'No, thanks',
        'ariaAccept':'Yes',
        'ariaDecline':'No, thanks',
        'ariaClose':'Close'
```

```
}).done(function(response){

        // Act upon the received response code

        switch(response){
        case 'accepted':oMyPlugin.command('WebChat.open');
                        break;
        case 'declined': break;
        case 'closed': break;
        case 'timeout': break;
}

});
```

Options

Option	Type	Description	Accepted values	Default	Introduced/ updated
type	string	Widget display type.	toast		
timeout	number	Timeout integer in milliseconds.	n/a		
title	string	String for widget title.	n/a		
ariaTitle	string	Aria label text for the Engage invite window.	n/a		9.0.015.04
body	string	String for offer body text.	n/a		
accept	string	String for Accept button text.	n/a		
ariaAccept	string	Aria label text for the Accept button.	n/a		9.0.016.10
decline	string	String for Decline button text.	n/a		
ariaDecline	string	Aria label text for the Decline button.	n/a		9.0.016.10
ariaClose	string	Aria label text for the Engage Close button.	n/a		9.0.016.10
command	string	Command to execute.	n/a		
options	object	Options related to the command provided.	n/a		

Option	Type	Description	Accepted values	Default	Introduced/ updated
priority	number	Replace the active lower priority Engage invite with the higher priority Engage invite.	n/a	0	9.0.015.11
force	boolean	Replace the active Engage invite with the new Engage invite irrespective of priorities.	n/a	false	9.0.015.11

Resolutions

Status	When	Returns
resolved	Engage invite is accepted by user.	accepted
resolved	Engage invite is declined by user.	declined
resolved	Engage invite widget is closed by user.	closed
resolved	Engage invite widget closes due to timeout.	timeout

offer

Opens a widget for a product offer using the data sent through the command options provided below. The widget can include both rendering options and the actual data that needs to be displayed in the Offer Widget. If no options are provided, the widget will not open.

Example

```
 oMyPlugin.command('Engage.offer', {

         mode:'overlay',
         modal:true,
         layout:'leftText',
         title: 'GRAB WHAT YOU NEED!!',
        ariaTitle: 'Offers',
         headline:'We Got All!',
         description:'Get free NextDay delivery on orders of $35 or more. Start shopping
now!',

         cta:{
                 text:'Join',
                 url:'https://www.genesys.com',
                 target:'_blank'
         },
```

```
        image:{
                src:'https://picsum.photos/id/237/300/300',
                alt:'Alternate Text for Image'
        },

        styles:{
                closeButton:{
                        'color':'red'
                }
        },
       ariaCTA:'Join',
       ariaClose:'Close Offer'
});
```

Options

Option	Type	Description	Accepted values	Default	Introduced/ updated
mode	string	The display type of the Offer widget.	overlay, toaster	toaster	9.0.015.04
modal	boolean	Applicable only when mode is 'overlay'. A smokescreen will be shown in the background of overlay modal window. This window can be dismissed by clicking anywhere in the smokescreen area.	n/a	false	9.0.015.04
layout	string	Additional layout options are supported for all modes.	minimal, leftText, rightText, topText, bottomText	leftText	9.0.015.04
headline	string	The Offer title header text.	n/a	n/a	9.0.015.04
ariaTitle	string	Aria label text for the Offer window.	n/a	n/a	9.0.015.04
description	string	The Offer body description text.	n/a	n/a	9.0.015.04
cta	object	An object containing HTML attributes and/	n/a	n/a	9.0.015.04

Option	Type	Description	Accepted values	Default	Introduced/ updated
		or CXBus commands for the CTA (call to action) button.			
cta.text	string	The CTA button text.	n/a	n/a	9.0.015.04
cta.url	string	The URL string for the CTA button. Note: The URL must be properly defined with the complete Protocol URL Address. For example, https://www.genesys.com.	_blank, _parent, _self, _top, framename	n/a	9.0.015.04
cta.target	string	Specifies where the URL is opened.	n/a	n/a	9.0.015.04
cta.command	string	A CXBus command to execute.	n/a	n/a	9.0.015.04
cta.commandOptions	string	Options related to CXBUs command.	n/a	n/a	9.0.015.04
image	object	An object containing image tag attributes.	n/a	n/a	9.0.015.04
image.src	string	The URL of the image.	n/a	n/a	9.0.015.04
image.alt	string	Alternate text for the image.	n/a	n/a	9.0.015.04
image.title	string	To indicate the screen reader user whether the image opens the URL in a new window.	n/a	n/a	9.0.016.10
insertAfter	string	Applicable only in mobile mode. An ID or class name of an HTML selector from the host page. The offer will be inserted after this	n/a	n/a	9.0.015.04

Option	Type	Description	Accepted values	Default	Introduced/ updated
		element. Precede the value mentioned here with the standard Class ('.') and ID selector ('#') character.			
insertBefore	string	Applicable only in mobile mode. An ID or class name of an HTML selector from the host page. The offer will be inserted before this element. Precede the value mentioned here with the standard Class ('.') and ID selector ('#') character.	n/a	n/a	9.0.015.04
insertInto	string	Applicable only in mobile mode. An ID or class name of an HTML selector from the host page. The offer will be appended inside this element. Precede the value mentioned here with the standard Class ('.') and ID selector ('#') character.	n/a	n/a	9.0.015.04
styles	object	An object containing styles for the offer content.	n/a	n/a	9.0.015.04
styles.closeButton	object	An object containing	n/a	n/a	9.0.015.04

Option	Type	Description	Accepted values	Default	Introduced/ updated
		styles for the close button.			
styles.closeButton.color	string	The color of the close button.	n/a	n/a	9.0.015.04
styles.closeButton.opacity	number	The CSS 'opacity' property for the close button.	n/a	n/a	9.0.015.04
styles.overlay	object	An object containing styles for the overlay container.	n/a	n/a	9.0.015.04
styles.overlay.top	string	The CSS 'top' property for the overlay container.	n/a	n/a	9.0.015.04
styles.overlay.right	string	The CSS 'right' property for the overlay container.	n/a	n/a	9.0.015.04
styles.overlay.bottom	string	The CSS 'bottom' property for the overlay container.	n/a	n/a	9.0.015.04
styles.overlay.left	string	The CSS 'left' property for the overlay container. Note: When all the position values are provided, the order of precedence will be top, right, bottom, and left.	n/a	n/a	9.0.015.04
styles.overlay.center	boolean	Aligns the overlay container to the center of the screen.	n/a	true	9.0.015.04
styles.offer	object	An object containing styles for the Offer window.	n/a	n/a	9.0.015.04
styles.offer.backgroundColor	string	The background	n/a	n/a	9.0.015.04

Option	Type	Description	Accepted values	Default	Introduced/ updated
		color of the offer.			
styles.offer.color	string	The text color of the offer.	n/a	n/a	9.0.015.04
styles.offer.padding	string	The padding for the offer container.	n/a	0	9.0.015.04
styles.title	object	An object containing styles for the title.	n/a	n/a	9.0.015.04
styles.title.font	string	The CSS 'font' property for the title.	n/a	n/a	9.0.015.04
styles.title.textAlign	string	The CSS 'text-align' property for the title.	n/a	n/a	9.0.015.04
styles.headline	object	An object containing styles for the header text.	n/a	n/a	9.0.015.04
styles.headline.font	string	The CSS 'font' property for the header text.	n/a	n/a	9.0.015.04
styles.headline.textAlign	string	The CSS 'text-align' property for the header text.	n/a	n/a	9.0.015.04
styles.description	object	An object containing styles for the offer description text.	n/a	n/a	9.0.015.04
styles.description.font	string	The CSS 'font' property for the description text.	n/a	n/a	9.0.015.04
styles.description.textAlign	string	The CSS 'text-align' property for the description text.	n/a	n/a	9.0.015.04
styles.ctaButton	object	An object containing styles for call to action button in the	n/a	n/a	9.0.015.04

Option	Type	Description	Accepted values	Default	Introduced/ updated
		offer window.			
styles.ctaButton.font	string	The CSS 'font' property for the text in CTA button.	n/a	n/a	9.0.015.04
styles.ctaButton.textAlign	string	The CSS 'text-align' property for the text in CTA button.	n/a	n/a	9.0.015.04
styles.ctaButton.background	string	The CSS 'background' property for the CTA button.	n/a	n/a	9.0.015.04
styles.ctaButton.color	string	The CSS 'color' property for the text in CTA button.	n/a	n/a	9.0.015.04
styles.ctaButton.fontSize	string	The CSS 'font-size' property for the text in CTA button.	n/a	n/a	9.0.015.04
ariaCTA	string	Aria label text for the Offer CTA button.	n/a	n/a	9.0.016.10
ariaClose	string	Aria label text for the Offer Close button.	n/a	n/a	9.0.016.10
priority	number	Replace the active lower priority Engage Offer with the higher priority Engage Offer.	n/a	0	9.0.015.11
force	boolean	Replace the active Engage Offer with the new Engage Offer irrespective of priorities.	n/a	false	9.0.015.11

API events

Once you've registered your plugin on the bus, you can subscribe to and listen for published events. Here's how to use the global bus object to register a new plugin on the bus.

> **Important**
>
> The global bus object is a debugging tool. When implementing widgets on your own site, do not use the global bus object to register your custom plugins. Instead, see Genesys Widgets Extensions for more information about extending Genesys Widgets.

```
var oMyPlugin = window._genesys.widgets.bus.registerPlugin('MyPlugin');

oMyPlugin.subscribe('Engage.ready', function(e){});
```

Name	Description	Data	Introduced/updated
ready	The Engage widget is initialized and ready to accept commands on the bus.	n/a	
opened	The Engage widget opens. `Note: Applicable only to Engage.offer command`	Metadata	9.0.015.04
CTA	When the user clicks the CTA button in the Engage widget. `Note: Applicable only to Engage.offer command`	Metadata	9.0.015.04
hover	When the user first hovers over the Engage widget. `Note: Applicable only to Engage.offer command`	Metadata	9.0.015.04
dismissed	When the user closes the Engage widget by clicking the Close button. `Note: Applicable only to Engage.offer command`	Metadata	9.0.015.04
closed	The Engage widget closes. `Note: Applicable only to Engage.offer command`	Metadata	9.0.015.04

> **Important**
> Applicable only for Engage.offer command.

Interaction Lifecycle

Every offer interaction has a sequence of events we describe as the *Interaction Lifecycle*. These events track progress and user choices from the beginning of an interaction (opening Offers), to the end (closing Offers), and every step in between.

The following events comprise the Interaction Lifecycle:

```
ready
opened
CTA
hover
dismissed
closed
```

Lifecycle scenarios

An Interaction Lifecycle can vary based on each user's intent and experience with the Offer widget. Here are several sequences of events in the lifecycle that correspond to different scenarios.

The user opened the Offer widget but changed their mind and closed it without seeing the offer details:

```
ready -> opened -> dismissed -> closed
```

The user opened the Offer widget, hovered over the offer details, and then closed it:

```
ready -> opened -> hover -> dismissed -> closed
```

The user opened the Offer widget and clicked on the button, which triggers CTA:

```
ready -> opened -> CTA -> closed
```

> **Tip**
> For a list of all Offer events, see API events.

Metadata

Each event in the Interaction Lifecycle includes the following block of metadata. By default, all values are set to false. As the user progresses through the lifecycle of an Offer Engage interaction, these values are updated.

The metadata block contains Boolean state flags, timestamps, and elapsed times. These values can be used to track and identify trends or issues with interactions. During runtime, the metadata can help you offer a smart and dynamic experience to your users.

Reference

Name	Type	Description	Introduced/updated
opened	integer (timestamp)	Timestamp indicating when the offer was opened.	9.0.015.04
closed	integer (timestamp)	Timestamp indicating when the offer was closed.	9.0.015.04
dismissed	integer (timestamp)	Timestamp indicating when the user dismissed the offer by clicking the close button.	9.0.015.04
triggeredCTA	integer (timestamp)	Timestamp indicating when the CTA was triggered.	9.0.015.04
timeBeforeCTA	integer (milliseconds)	Total time in milliseconds from when the user opened the offer to when the CTA is triggered.	9.0.015.04
timeFirstHover	integer (timestamp)	Timestamp indicating when the user first hovered over the offer.	9.0.015.04
timeBeforeHover	integer (milliseconds)	Total time in milliseconds from when the user opened the offer to when the user first hovered over the offer.	9.0.015.04
timeElapsedHover	integer (milliseconds)	Total time in milliseconds when the user hovered over the offer.	9.0.015.04
elementClicked	string	Name of CTA element that was clicked.	9.0.015.04