Widgets Developer Resources

WebChat

3/20/2023

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

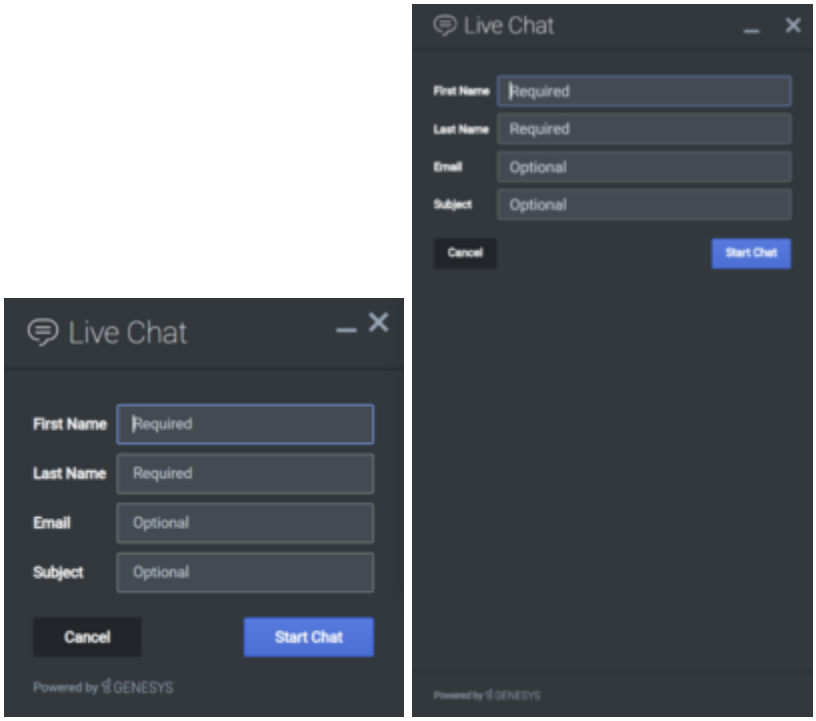
WebChat supports both desktop and mobile devices. Like all Genesys Widgets, there are two main modes: desktop and mobile. Desktop is employed for monitors, laptops, and tablets, and mobile is employed for smartphones. When a smartphone is detected, WebChat switches to special full-screen templates that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

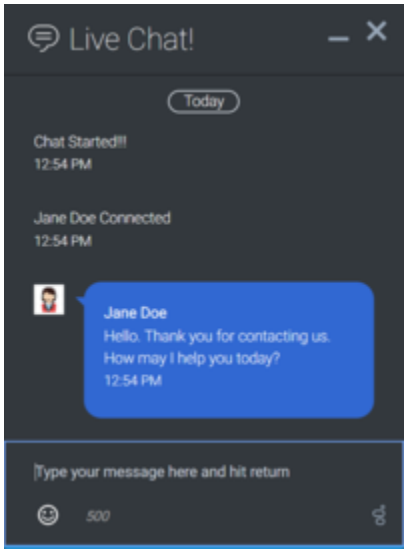
Screenshots

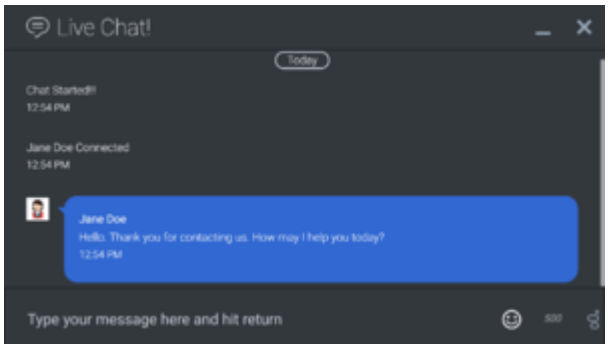
Dark theme

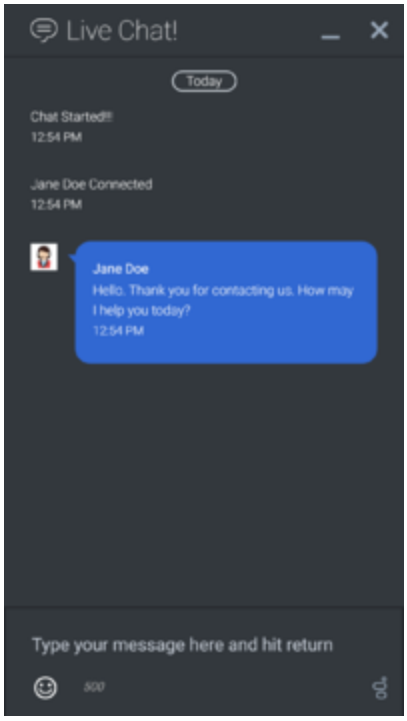
WebChat forms



WebChat transcripts

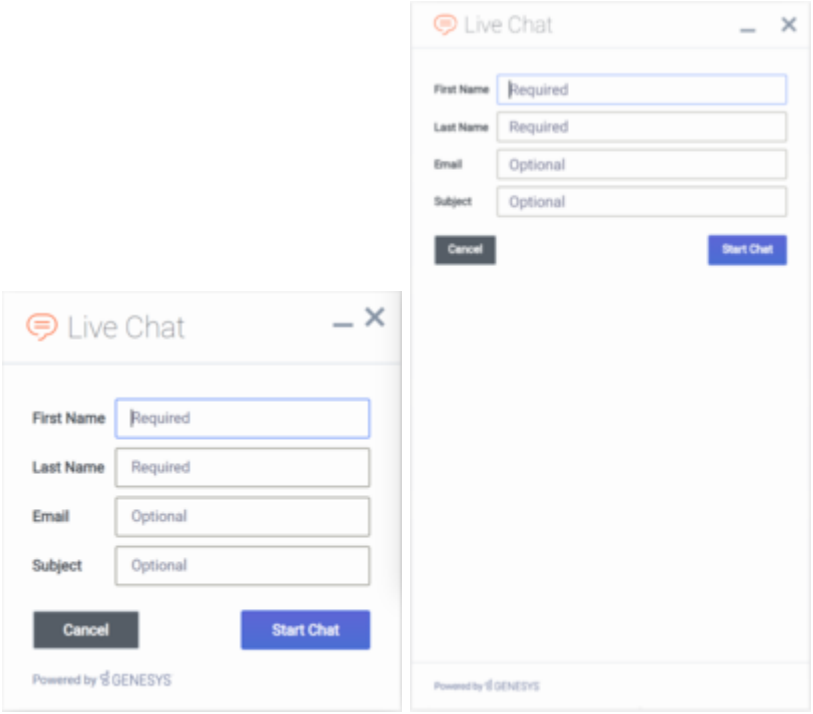


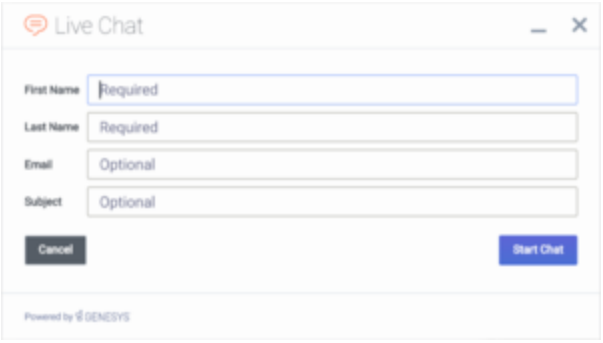




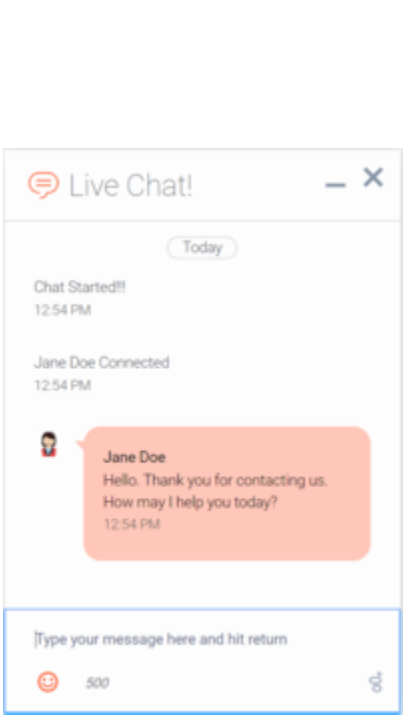
Light theme

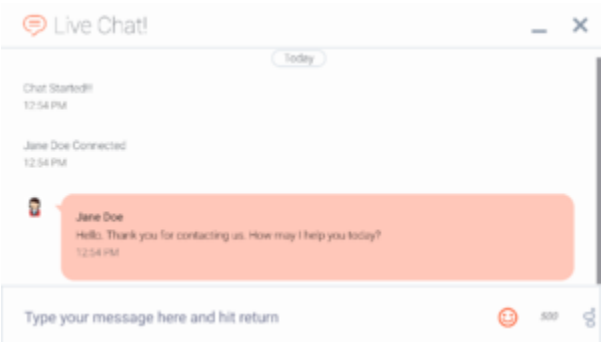
WebChat forms





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.

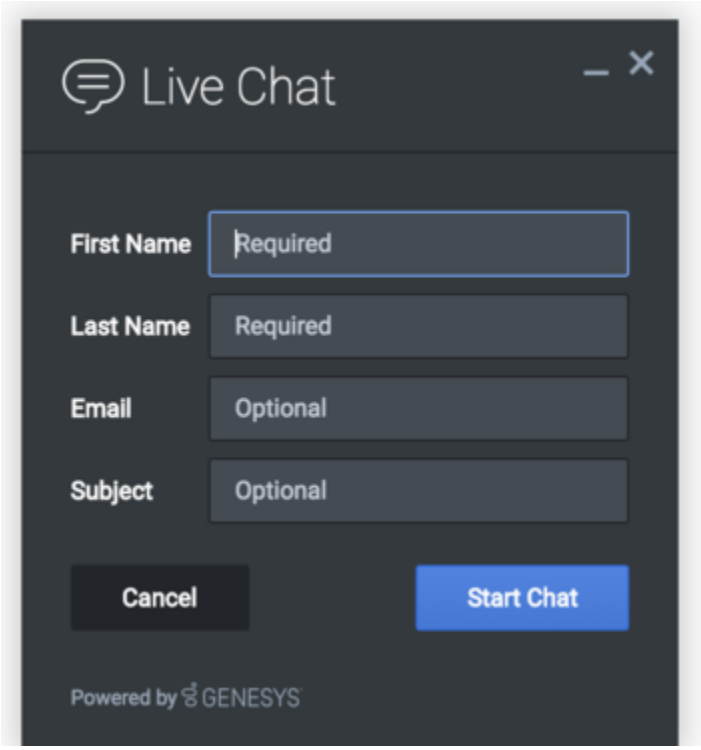
Default example

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.

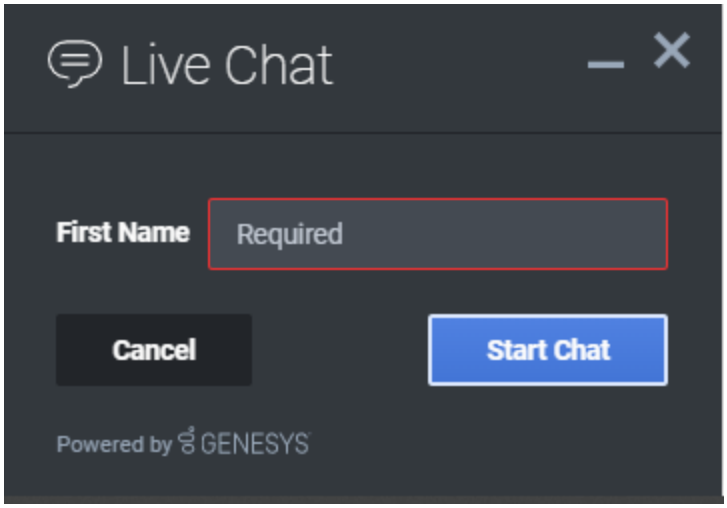
Validation

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.

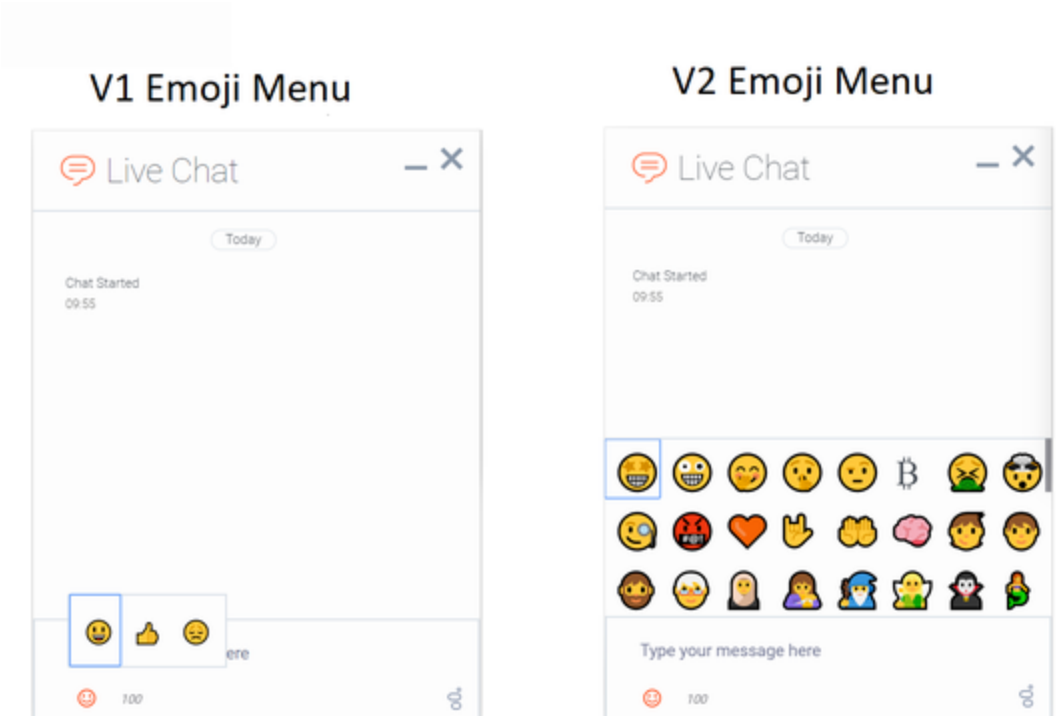
Once the form data object is sent to the WebChatService plugin, it will parse and send in the payload request.

```
 {
   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

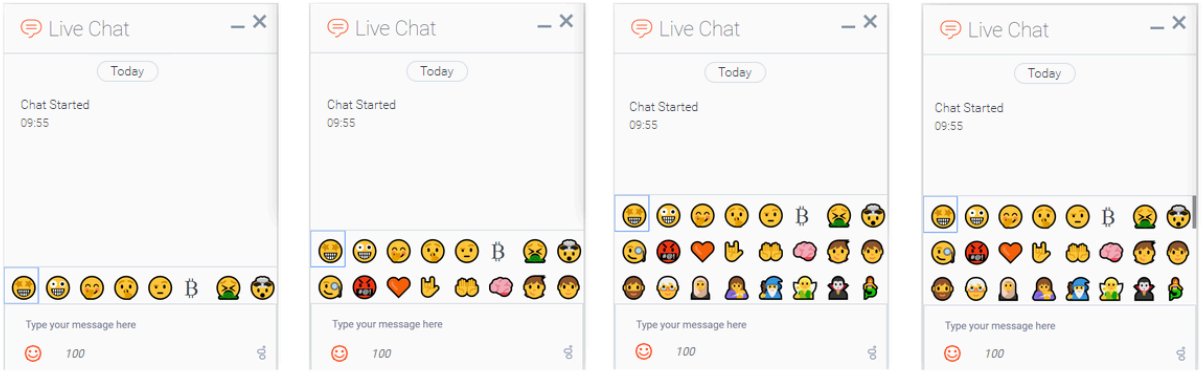
- v1 shows as a tooltip-style overlay; v2 shows as a new block between the transcript and the message input.
- v1 closes when you select an emoji or click outside the menu; v2 lets you choose multiple emojis and only closes if you click the emoji menu button again.
- v1 has three fixed emojis to choose 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 =
"😀😁😂🤣😃😄😅😆😉😊😋😎😍😘🥰😗😙😚☺🙂🤗🤩🤔🤨😐😑😶🙄😏😣😥😮🤐😯😪😫😴😌😛😜😝🤤😒😓😔😕🙃🤑😲☹🙁😖😞😟😤😢😭😦😧😨😩🤯😬😰😱🥵🥶😳🤪😵🥴😠😡🤬😷🤒🤕🤢🤮🤧😇🤠🤡🥳🥺🤥🤫🤭🧐🤓😈👿👹👺💀☠👻👽👾🤖💩😺😸😹😻😼😽🙀😿😾🙈🙉🙊👶👧🧒👦👩🧑👨👵🧓👴👲👳🧕👮👷💂🕵👩⚕👨⚕👩🌾👨🌾👩🍳👨🍳👩🎓👨🎓👩🎤👨🎤👩🏫👨🏫👩🏭👨🏭👩💻👨💻👩💼👨💼👩🔧👨🔧👩🔬👨🔬👩🎨👨🎨👩🚒👨🚒👩✈👨✈👩🚀👨🚀👩⚖👨⚖👰🤵👸🤴🦸🦹🤶🎅🧙🧝🧛🧟🧞🧜🧚👼🤰🤱🙇💁🙅🙆🙋🤦🤷💆💇🚶🏃💃🕺👯🧖🧗🤺🏇⛷🏂🏌🏄🚣🏊⛹🏋🚴🚵🤸🤼🤽🤾🤹🧘🛀🛌👭👫👬💏💑👪🗣👤👥👣🐵🐒🦍🐶🐕🦮🐩🐺🦊🦝🐱🐈🦁🐯🐅🐆🐴🐎🦄🦓🦌🐮🐂🐃🐄🐷🐖🐗🐽🐏🐑🐐🐪🐫🦙🦒🐘🦏🦛🐭🐁🐀🐹🐰🐇🐿🦔🦇🐻🐨🐼🦥🦦🦨🦘🦡🐾🦃🐔🐓🐣🐤🐥🐦🐧🕊🦅🦆🦢🦉🦩🦚🦜🐸🐊🐢🦎🐍🐲🐉🦕🦖🐳🐋🐬🐟🐠🐡🦈🐙🐚🦀🦞🦐🦑🐌🦋🐛🐜🐝🐞🦗🕷🕸🦂🦟🦠💐🌸💮🏵🌹🥀🌺🌻🌼🌷🌱🌲🌳🌴🌵🌾🌿☘🍀🍁🍂🍃🍇🍈🍉🍊🍋🍌🍍🥭🍎🍏🍐🍑🍒🍓🥝🍅🥥🥑🍆🥔🥕🌽🌶🥒🥬🥦🧄🧅🍄🥜🌰🍞🥐🥖🥨🥯🥞🧇🧀🍖🍗🥩🥓🍔🍟🍕🌭🥪🌮🌯🥙🧆🥚🍳🥘🍲🥣🥗🍿🧈🧂🥫🍱🍘🍙🍚🍛🍜🍝🍠🍢🍣🍤🍥🥮🍡🥟🥠🥡🦀🦞🦐🦑🍦🍧🍨🍩🍪🎂🍰🧁🥧🍫🍬🍭🍮🍯🍼🥛☕🍵🍶🍾🍷🍸🍹🍺🍻🥂🥃🥤🧃🧉🧊🥢🍽🍴🥄🔪🏺🌍🌎🌏🌐🗺🗾🧭🏔⛰🌋🗻🏕🏖🏜🏝🏞🏟🏛🏗🧱🏘🏚🏠🏡🏢🏣🏤🏥🏦🏨🏩🏪🏫🏬🏭🏯🏰💒🗼🗽⛪🕌🛕🕍⛩🕋⛲⛺🌁🌃🏙🌄🌅🌆🌇🌉♨🎠🎡🎢💈🎪🚂🚃🚄🚅🚆🚇🚈🚉🚊🚝🚞🚋🚌🚍🚎🚐🚑🚒🚓🚔🚕🚖🚗🚘🚙🚚🚛🚜🏎🏍🛵🦽🦼🛺🚲🛴🛹🚏🛣🛤🛢⛽🚨🚥🚦🛑🚧⚓⛵🛶🚤🛳⛴🛥🚢✈🛩🛫🛬🪂💺🚁🚟🚠🚡🛰🚀🛸🛎🧳⌛⏳⌚⏰⏱⏲🕰🕛🕧🕐🕜🕑🕝🕒🕞🕓🕟🕔🕠🕕🕡🕖🕢🕗🕣🕘🕤🕙🕥🕚🕦🌑🌒🌓🌔🌕🌖🌗🌘🌙🌚🌛🌜🌡☀🌝🌞🪐⭐🌟🌠🌌☁⛅⛈🌤🌥🌦🌧🌨🌩🌪🌫🌬🌀🌈🌂☂☔⛱⚡❄☃⛄☄🔥💧🌊🎃🎄🎆🎇🧨✨🎈🎉🎊🎋🎍🎎🎏🎐🎑🧧🎀🎁🎗🎟🎫🎖🏆🏅🥇🥈🥉⚽⚾🥎🏀🏐🏈🏉🎾🥏🎳🏏🏑🏒🥍🏓🏸🥊🥋🥅⛳⛸🎣🤿🎽🎿🛷🥌🎯🪀🪁🎱🔮🧿🎮🕹🎰🎲🧩🧸♠♥♦♣♟🃏🀄🎴🎭🖼🎨🧵🧶👓🕶🥽🥼🦺👔👕👖🧣🧤🧥🧦👗👘🥻🩱🩲🩳👙👚👛👜👝🛍🎒🩴👞👟🥾🥿👠👡🩰👢👑👒🎩🎓🧢⛑📿💄💍💎🔇🔈🔉🔊📢📣📯🔔🔕🎼🎵🎶🎙🎚🎛🎤🎧📻🎷🎸🎹🎺🎻🥁📱📲☎📞📟📠🔋🔌💻🖥🖨⌨🖱🖲💽💾💿📀🧮🎥🎞📽🎬📺📷📸📹📼🔍🔎🕯💡🔦🏮🪔📔📕📖📗📘📙📚📓📒📃📜📄📰🗞📑🔖🏷💰💴💵💶💷💸💳🧾💹✉📧📨📩📤📥📦📫📪📬📭📮🗳✏✒🖋🖊🖌🖍📝💼📁📂🗂📅📆🗒🗓📇📈📉📊📋📌📍📎🖇📏📐✂🗃🗄🗑🔒🔓🔏🔐🔑🗝🔨🪓⛏⚒🛠🗡⚔🔫🏹🛡🔧🔩⚙🗜⚖🦯🔗⛓🧰🧲⚗🧪🧫🧬🔬🔭📡💉🩸💊🩹🩺🚪🛗🪞🪟🛏🛋🪑🚽🪠🚿🛁🪤🪒🧴🧷🧹🧺🧻🪣🧼🪥🧽🧯🛒🚬⚰🪦⚱🗿🏧🚮🚰♿🚹🚺🚻🚼🚾🛂🛃🛄🛅⚠🚸⛔🚫🚳🚭🚯🚱🚷📵🔞☢☣⬆↗➡↘⬇↙⬅↖↕↔↩↪⤴⤵🔃🔄🔙🔚🔛🔜🔝🛐⚛🕉✡☸☯✝☦☪☮🕎🔯♈♉♊♋♌♍♎♏♐♑♒♓⛎🔀🔁🔂▶⏩⏭⏯◀⏪⏮🔼⏫🔽⏬⏸⏹⏺⏏🎦🔅🔆📶📳📴♀♂⚧✖➕➖➗♾‼⁉❓❔❕❗〰💱💲⚕♻⚜🔱📛🔰⭕✅☑✔❌❎➰➿〽✳✴❇©®™#⃣*⃣0⃣1⃣2⃣3⃣4⃣5⃣6⃣7⃣8⃣9⃣🔟🔠🔡🔢🔣🔤🅰🆎🅱🆑🆒🆓ℹ🆔Ⓜ🆕🆖🅾🆗🅿🆘🆙🆚🈁🈂🈷🈶🈯🉐🈹🈚🈲🉑🈸🈴🈳㊗㊙🈺🈵🔴🟠🟡🟢🔵🟣🟤⚫⚪🟥🟧🟨🟩🟦🟪🟫⬛⬜◼◻◾◽▪▫🔶🔷🔸🔹🔺🔻💠🔘🔳🔲🏁🚩🎌🏴🏳🏳🌈🏴☠🇦🇨🇦🇩🇦🇪🇦🇫🇦🇬🇦🇮🇦🇱🇦🇲🇦🇴🇦🇶🇦🇷🇦🇸🇦🇹🇦🇺🇦🇼🇦🇽🇦🇿🇧🇦🇧🇧🇧🇩🇧🇪🇧🇫🇧🇬🇧🇭🇧🇮🇧🇯🇧🇱🇧🇲🇧🇳🇧🇴🇧🇶🇧🇷🇧🇸🇧🇹🇧🇻🇧🇼🇧🇾🇧🇿🇨🇦🇨🇨🇨🇩🇨🇫🇨🇬🇨🇭🇨🇮🇨🇰🇨🇱🇨🇲🇨🇳🇨🇴🇨🇵🇨🇷🇨🇺🇨🇻🇨🇼🇨🇽🇨🇾🇨🇿🇩🇪🇩🇬🇩🇯🇩🇰🇩🇲🇩🇴🇩🇿🇪🇦🇪🇨🇪🇪🇪🇬🇪🇭🇪🇷🇪🇸🇪🇹🇪🇺🇫🇮🇫🇯🇫🇰🇫🇲🇫🇴🇫🇷🇬🇦🇬🇧🇬🇩🇬🇪🇬🇫🇬🇬🇬🇭🇬🇮🇬🇱🇬🇲🇬🇳🇬🇵🇬🇶🇬🇷🇬🇸🇬🇹🇬🇺🇬🇼🇬🇾🇭🇰🇭🇲🇭🇳🇭🇷🇭🇹🇭🇺🇮🇨🇮🇩🇮🇪🇮🇱🇮🇲🇮🇳🇮🇴🇮🇶🇮🇷🇮🇸🇮🇹🇯🇪🇯🇲🇯🇴🇯🇵🇰🇪🇰🇬🇰🇭🇰🇮🇰🇲🇰🇳🇰🇵🇰🇷🇰🇼🇰🇾🇰🇿🇱🇦🇱🇧🇱🇨🇱🇮🇱🇰🇱🇷🇱🇸🇱🇹🇱🇺🇱🇻🇱🇾🇲🇦🇲🇨🇲🇩🇲🇪🇲🇫🇲🇬🇲🇭🇲🇰🇲🇱🇲🇲🇲🇳🇲🇴🇲🇵🇲🇶🇲🇷🇲🇸🇲🇹🇲🇺🇲🇻🇲🇼🇲🇽🇲🇾🇲🇿🇳🇦🇳🇨🇳🇪🇳🇫🇳🇬🇳🇮🇳🇱🇳🇴🇳🇵🇳🇷🇳🇺🇳🇿🇴🇲🇵🇦🇵🇪🇵🇫🇵🇬🇵🇭🇵🇰🇵🇱🇵🇲🇵🇳🇵🇷🇵🇸🇵🇹🇵🇼🇵🇾🇶🇦🇷🇪🇷🇴🇷🇸🇷🇺🇷🇼🇸🇦🇸🇧🇸🇨🇸🇩🇸🇪🇸🇬🇸🇭🇸🇮🇸🇯🇸🇰🇸🇱🇸🇲🇸🇳🇸🇴🇸🇷🇸🇸🇸🇹🇸🇻🇸🇽🇸🇾🇸🇿🇹🇦🇹🇨🇹🇩🇹🇫🇹🇬🇹🇭🇹🇯🇹🇰🇹🇱🇹🇲🇹🇳🇹🇴🇹🇷🇹🇹🇹🇻🇹🇼🇹🇿🇺🇦🇺🇬🇺🇲🇺🇳🇺🇸🇺🇾🇺🇿🇻🇦🇻🇨🇻🇪🇻🇬🇻🇮🇻🇳🇻🇺🇼🇫🇼🇸🇽🇰🇾🇪🇾🇹🇿🇦🇿🇲🇿🇼🏴🏴🏴";
```

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 = "🥥🥦🥨🥩🇦🇺🇫🇷🎂🛍️🇨🇦🎪🧧🎄
🇲🇽🪔🐉🐰🎬🍂👨🎉🧨🎃🕎💕🎨🇺🇸👩🎆🏅🏳️‍🌈👑☪️🌷☘️☀️🏈🦃💘💒❄️⛷️🌍🌎;🎨: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;"
        }
    }
}
```