Widgets API Reference

WebChatService

6/10/2026

Contents

- Developer

Learn how to use Genesys chat services.

Related documentation:

-

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 widgets. 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 WebChat Service 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.

Limitations

Multiple instances of the same chat session

After starting a chat session, that session can be opened in any number of new tabs on the same site. Each tab runs an independent instance of WebChat connected to the same chat session. Currently, Instances are not synchronized with each other due to Nexus limitation.

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 in configuration options.

Example

```
// When using v2 API

window._genesys.widgets.webchat = {

        apikey: 'n3eNkgxxxxxxxxxxx8VA',
        dataURL: 'https://api.genesyscloud.com/gms-chat/2/chat',
        enableCustomHeader: true,

        userData: {},
        emojis: true,
        actionsMenu: true,

        autoInvite: {

                enabled: false,
                timeToInviteSeconds: 10,
                inviteTimeoutSeconds: 30
        },

        chatButton: {

                enabled: true,
                template: '
CHAT NOW
',
                effect: 'fade',
                openDelay: 1000,
                effectDuration: 300,
                hideDuringInvite: true
        }
};

// When using v3 API

window._genesys.widgets.webchat = {

        emojis: true,
        userData: {},
        transport: {

                type: 'pureengage-v3-rest',
                dataURL: https:///nexus/v3/chat/sessions,
                endpoint: 'xxxxxxxxx',
```

```
                headers: {
                        'x-api-key': 'xxxxxxxx'
                },
                async: {

                        enabled: true,

                        getSessionData: function(sessionData, Cookie, CookieOptions) {

                                // Note: You don't have to use cookies. You can, instead,
store in a secured location like a database.
                                Cookie.set('customer-defined-session-cookie',
JSON.stringify(sessionData), CookieOptions);
                        },

                        setSessionData: function(Open, Cookie, CookieOptions) {

                                // Retrieve from your secured location.
                                return Cookie.get('customer-defined-session-cookie');
                        }
                },

        chatButton: {

                enabled: true,
                template: '
CHAT NOW
',
                effect: 'fade',
                openDelay: 1000,
                effectDuration: 300,
                hideDuringInvite: true
        }
};
```

Options

Version 2 API

Name	Type	Description	Default	Required	Introduced/ Updated
apikey	string	Apigee Proxy secure token. **Important** This option is only supported in GMS REST mode.	n/a	Yes, if using Apigee Proxy	
endpoint	string	Manually select the endpoint on which to initiate chat.	n/a	n/a	
dataURL	string (URL)	URL for GMS REST chat service. If cometD.enabled	n/a	Always	

Name	Type	Description	Default	Required	Introduced/ Updated
		is set to true, this property will be ignored.			
enableCustomHeader	boolean	Enables the use of the custom authorization header defined in _genesys.widgets.main.header static config. Attaches the custom authorization header to all WebChatService request.	false	No	9.0.002.06
userData	object	Arbitrary attached data to include when initiating a chat.	{}	n/a	
ajaxTimeout	number	Number of milliseconds to wait before AJAX timeout.	3000	n/a	
xhrFields	object	Allows you to set the properties for the AJAX xhrFields object (for example, {withCredentials: false}). **Important** This option is only supported in GMS REST mode.	{withCredentials: false}	n/a	
pollExceptionLimit	number	Number of successive poll exceptions (chat server offline) before WebChatService publishes 'chatServerWentOffline'.	5	n/a	
restoreTimeout	number	Number of milliseconds before restore	60000		

Name	Type	Description	Default	Required	Introduced/ Updated
		timeout. Prevents the chat session from restoring after a certain time away from the session (for example, user navigated to a different site during chat and never ended the session).			

Version 3 API

Name	Type	Description	Default	Required	Introduced/ Updated
transport	object	Object containing the transport service configuration options.	n/a	Yes, when using new transport services available with WebChat.	9.0.008.04
transport.type	string	Select the type of transport service that needs to work with WebChat UI plugin. For Pure Engage v3 REST API, the value is 'pureengage-v3-rest'.	n/a	Yes, when using Pure Engage v3 REST API.	9.0.008.04
transport.dataURL	string (URL)	URL for Pure Engage v3 REST API chat service. Please contact your local Genesys customer representative to obtain a valid dataURL.	n/a	Always	9.0.008.04
transport.endpoint	string	The endpoint for Genesys Multicloud CX v3 API.	n/a	Yes	9.0.008.04
transport.headers	object	Object	n/a	Yes	9.0.008.04

Name	Type	Description	Default	Required	Introduced/ Updated
		containing key value pairs of any custom headers.			
transport.headers[x-api-key]	string	The API key provided from Genesys. Please contact your local Genesys customer representative to obtain a valid API key.	n/a	Yes	9.0.008.04
transport.async	object	Object containing Async mode configuration options. **Important** To properly restore a chat session that has ended previously, you'll need to navigate back to the page and open the WebChat Widget. This way, the chat session is restored in the background and is ready. Presently, this is a current limitation in Async WebChat.	{}	No	9.0.008.04
transport.async.enabled	boolean	Enable Asynchronous Chat where a chat session can be active indefinitely. When you close WebChat without ending the chat session, the session will simply go dormant. When you open	false	Yes, when Async WebChat mode is enabled	9.0.008.04

Name	Type	Description	Default	Required	Introduced/ Updated
		WebChat again, the session will restore and continue chatting where left off.			
transport.async.getSessionData	function	A function that you can define to retrieve updated session data from the WebChatService plugin. This function is called back when starting a new Async chat session for the first time, or when the sessionData changes over the course of an active chat session. This function takes the following arguments: sessionData (current active session data), Cookie (Widgets Internal cookie reference), and CookieOptions (a parameter that is needed when using Widgets Cookie). The purpose of this function is to provide you with the active session data so that it can be stored somewhere safe and secure. Later	none	Yes, when Async WebChat mode is enabled	9.0.008.04

Name	Type	Description	Default	Required	Introduced/ Updated
		this needs to be provided in the below **setSessionData** function to restore the chat session. Refer to the example for usage.			
transport.async.setSessionData	function	A function that you can define to return the session data to the WebChatService plugin. During initialization, the WebChatService plugin will call this function to check if any session data is returned. If found, WebChatService tries to restore the chat session using this session data and open the WebChat Widget. WebChatService will also pass the following arguments into this function: Open (WebChat current open state value), Cookie (Widgets Internal cookie reference), and CookieOptions (a parameter that isneeded when using Widgets Cookie). Refer to the example	none	Yes, when Async WebChat mode is enabled	9.0.008.04

Name	Type	Description	Default	Required	Introduced/ Updated
		for usage.			
transport.async.deleteSessionData	function	A function that you can define to delete the session data from your secret storage, it will be called by WebChatService plugin when Async chat session is lost or cannot find anymore due to unknown reasons. This function will enable you write the script for deleting the session data from your secret storage, in this way WebChat will try to start a new chat normally rather than trying to restore a lost chat session. WebChatService will also pass the following arguments into this function - errorData (lost session and error details), Cookie (Widgets Internal cookie reference) and CookieOptions (a parameter that will be needed when using Widgets Cookie).	none	Yes, when Async WebChat mode is enabled	9.0.015.12
userData	object	Arbitrary attached data to include when initiating	{}	n/a	

Name	Type	Description	Default	Required	Introduced/ Updated
		a chat.			
ajaxTimeout	number	Number of milliseconds to wait before AJAX timeout.	3000	n/a	

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 options.

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.

> **Important**
>
> The options data must be under the “form” object when using the "WebChatService.startChat" command in v3 API.

Example

```
// When using v2 API

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
});

// When using v3 API

oMyPlugin.command('WebChatService.startChat', {

  form:{
          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'.
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)

Status	When	Returns
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 Desktop.

> **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	'previous poll has not	

Status	When	Returns	Introduced/Updated
	currently active	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	No active chat session	

Status	When	Returns	Introduced / Updated
	currently active		
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	(JWT string token)	9.0.008.04

Option	Type	Description	Accepted Values	Introduced / Updated
		mode. It is a JWT token string value. Applicable only when using WebChat with Genesys Multicloud CX v3 API. For more information, see the “Genesys Multicloud CX v3” tab in the “Options” table in configuration options.		

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 currently participating 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), 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
field	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 options.	{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	{participantId: (string), sessionId: {string), token: (string), transportId: (string)}	9.0.008.04

Status	When	Returns	Introduced / Updated
	configuration options.		
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. This internal command determines the last received message index and, based on this information, fetches older messages whenever it is called.

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.

```
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
                            }
                }
}).fail(function(e){
                // verifySession not supported for the transport
});
```

Resolutions

Status	When	Returns
resolved	A session exists or not	A boolean *sessionActive*, which holds the session state
rejected	The *verifySession* command is not supported for this transport	This transport doesn't support the *verifySession* 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('WebChatService.ready', function(e){});
```

Name	Description	Data	Introduced/updated
ready	WebChatService is initialized and ready to accept commands.	n/a	
restored	Chat session has been restored after page navigation or refresh. In Asynchronous mode, this event includes data indicating whether a chat session has been restored in Async mode or not.	{async: (boolean)}	9.0.002.06
restoreTimeout	Chat session restoration attempted was denied after user navigated	n/a	

Name	Description	Data	Introduced/updated
	away from originating website for longer than the time limit: default 60 seconds.		
restoreFailed	Could not restore chat session after page navigation or refresh.	n/a	
restoredOffline	Chat session was restored normally but chat server is offline. This means no messages can come through. When chat server is comes back online, 'chatServerBackOnline' is published.	n/a	
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)}	9.0.002.06
error	An error occurred between the client and the server.	(AJAX Response)	
started	Chat session has successfully started.	(AJAX Response containing session data)	
ended	Chat session has successfully ended.	n/a	
agentTypingStarted	Agents has started typing a new message.	(AJAX Response)	
agentTypingStopped	Agent has stopped typing.	(AJAX Response)	
pollingStarted	Chat server automatic polling has started.	n/a	
pollingStopped	Chat server automatic polling has stopped.	n/a	
clientConnected	Indicates the user has been connected to the chat session.	{message: (object), agents: (object), numAgentsConnected: (number)}	
clientDisconnected	Indicates the user has been disconnected form the chat session.	{message: (object), agents: (object), numAgentsConnected: (number)}	
agentConnected	Indicates an agent has connected to the chat.	{message: (object), agents: (object), numAgentsConnected:	

Name	Description	Data	Introduced/updated
		(number)}	
agentDisconnected	Indicates an agent has disconnected from the chat.	{message: (object), agents: (object), numAgentsConnected: (number)}	
supervisorConnected	Indicates a supervisor has connected to the chat.	{message: (object), agents: (object), numAgentsConnected: (number)}	
supervisorDisconnected	Indicates a supervisor has disconnected from the chat.	{message: (object), agents: (object), numAgentsConnected: (number)}	
botConnected	Indicates a bot has connected to the chat. **Important** This event is applicable only when using v2 API.	{message: (object), agents: (object), numAgentsConnected: (number)}	9.0.014.13
botDisconnected	Indicates a bot has disconnected from the chat. **Important** This event is applicable only when using v2 API.	{message: (object), agents: (object), numAgentsConnected: (number)}	9.0.014.13
clientTypingStarted	The user has started typing. Sends an event to the agent.	n/a	
clientTypingStopped	After a user stops typing, a countdown begins. When the countdown completes, the typing notification will clear for the agent.	n/a	
disconnected	Cannot reach servers. No connection. Either the user is offline or the server is offline.	n/a	
reconnected	Connection restored. This event is only published after 'disconnected'.	n/a	
chatServerWentOffline	Chat server has gone offline but chat session has not ended. New messages are temporarily unavailable. This event is published only after the	n/a	

Name	Description	Data	Introduced/updated
	configuration option 'pollExceptionLimit' has been exceeded. Default limit is 5 poll exceptions. 'restoredOffline' is an alternate to this event that is used only when the chat server is down while trying to restore your chat session. The reason for having two events is to allow for separate handling of both scenarios. Important This event is applicable only when using v2 API.		
chatServerBackOnline	Chat server had come back online after going offline. This will only be published after 'chatServerWentOffline'. Important This event is applicable only when using v2 API.	n/a	
connectionPending	If there is a connection problem and WebChatService is trying to reconnect, this event will be published. Published before 'chatServerWentOffline'. Important This event is applicable only when using v2 API.	n/a	
connectionRestored	Is published when the connection has be reestablished. Publishes at the same time as 'chatServerBackOnline'.	n/a	