API Reference

Controlling the Chat Session

4/20/2025

Controlling the Chat Session

Chat session commands

Session commands are used for client-to-server communication: sending commands to chat server. All commands receive their arguments as a single "options" object, similar to startSession and restoreSession. For example,

```
session.getTranscript({fromIndex: 0});

// instead of
session.getTranscript(0);
```

All exceptions to this rule are documented here.

Returned Promises

All of the commands return a "promise" object with two properties: `done` and `fail`.

Some of the "done" callbacks can be used to obtain specific information provided as the result of command execution. If that is the case, the information is specifically documented for particular commands. Most of the time however these callbacks serve as a way to simply acknowledge that the command was successfully executed. In this case, there is no specific documentation for the corresponding command callback.

"fail" callbacks can be used to catch errors that happen during command execution and all receive an object with exact same structure:

event.error.code	Code specifying the particular error
event.error.description	Description of the error (English is default language).

There is no specific documentation for each command's "fail" callback.

Here is an example of general command usage:

```
chat.<ABSTRACT_CHAT_COMMAND>(<COMMAND_OPTIONS>).done(function(<POSSIBLE_RESULTS>) {
  // Command executed successfully.
}).fail(function(event) {
  // Something went wrong. See event.error.code and event.error.description.
});
```

session.getTranscript

Description

A low-level method used to obtain the transcript for the current session.

> **Important**
> No playback is assumed.

Options

Parameter	Type	Default value	Mandatory	Description
`fromIndex`	number	0	No	`0`-based index to retrieve chat transcript from a certain position. `0` means obtaining the entire transcript.

"done" callback

Receives the transcript in serialized JS form.

Parameter	Type	Description

`event.transcript`	Array	Array of JS objects representing transcipt events.
		
		Sample:
		
		`[`
		`  {`

Parameter	Type	Description

		type: 'AgentConnected',
		party: {id: 2 /*ID of this party unique for THIS chat session*/, type: 'Agent', name: <name of an agent>},
		index: <index of THIS message in chat transcript>,
		timestamp: <UTC Timestampt>
		},
		{
		type: 'MessageReceived',
		party: {id: 1, type: <type of party which sent message: 'Client' or 'Agent' or 'External'>, name: <name of party>},
		content: {text: <text which was sent>, type: <type of text: 'text' or 'url'>},
		index: <index of THIS message in chat transcript>,
		timestamp: <UTC Timestampt>
		},
		{
		type: 'AgentDisconnected',
		party: {id: 2 /*ID of this party unique for THIS chat session*/, type: 'Agent', name: <name of an agent>},
		index: <index of THIS message in chat transcript>,
		timestamp: <UTC Timestampt>
		},
		{
		type: 'SessionEnded',
		reason: {code: <code of reason: 1: by leave request, 2 - by an agent, 3 - by error>, description: <default description>},
		timestamp: <UTC Timestampt>
		}
		]

```
session.getTranscript().done(function(event) {
  console.log('Full transcript of current chat session: ', event.transcript);
});
```

session.sendMessage

Description

Send a message to Chat Server.

Options

Parameter	Type	Default value	Mandatory	Description
type	string	'text'	No	Type of message: "text" or "url" If absent, "text" will be used.
message	string	undefined	Yes	Message to be sent

Since sending a "text" message is a much more frequent operation than sending a "url", a shortcut is available; you can pass a string with message contents directly to the method instead of the options object.

```
// This
session.sendMessage({ type: 'text', message: 'foobar' });

// is equivalent to this:
session.sendMessage('foobar');
```

session.sendTyping

Description

Notify Chat Server that client started or stopped typing in chat session.

Options

Parameter	Type	Default value	Mandatory	Description
isTyping	boolean	true	No	Boolean (true or false) which specifies exact meaning of this command • **true** — visitor started typing (or continues typing) in chat session. • **false** — visitor stopped typing

Parameter	Type	Default value	Mandatory	Description
				in chat session (typically a stop or pause in typing for a certain duration, for example — 5 seconds)

session.leave

Description

This command is used to complete chat session in Chat Server by request from the visitor side.

Options

This command does not take parameters.

Chat session events (callbacks)

You can pass callback functions into a chat session that will be called each time a chat context is updated, or whenever other changes take place within the session (for example, agent joins/leaves, Chat Server stops responding, and so on). Another use for session events is in a "playback" scenario, when a chat session is restored in a new browser context (for example, after page reload/navigation).

To add a callback, pass the callback function directly to the corresponding event method. Most of the callbacks receive an event object with properties containing event details. For example,

```
session.onError(function(event) {
    // event.error.code
    // event.error.description
});
```

session.onError

This event will be sent in reaction to an unexpected error occurring during the flow of the chat session.

Additionally, this event is sent when the chat component needs to notify clients about unusual cases. For example:

- Chat Server stops responding and the component tries to restore the session on another server.
- A chat session is restored on another instance of Chat Server.
- A channel to the Chat Server is opened, but the server is not yet ready to send/receive operations.

Another important scenario for this callback — it is triggered if an incorrect set of parameters (or invalid parameter value) is detecing in an incoming "raw" event. For example, if the `content` of a message in a `received message` event is not a string.

Event structure

Parameter	Description
event.error.code	Code specifying the particular error
event.error.description	Default description of error

> **Tip**
> For a list of possible error codes, see Error Codes.

session.onAgentConnected

Executed when an agent joins a chat session.

Event structure

Parameter	Description
`event.party.name`	String that represents name of the agent joined to the chat session
`event.timestamp`	Number
`event.party.id`	Theoretically, the agent name and visitor name could be the same (especially since the visitor might be filling out the the visitor name). To handle this scenario, this id is used to distinguish between agent and visitor names.
`event.index`	Index of this message in chat transcript
`event.restored`	Optional. If present and true, it means that the event was restored during session restoration (in other words, event was already reported to the consumer previously).

session.onAgentDisconnected

Executed when agent leaves chat session, for any of these possible reasons:

- Session closes because of a logout request from the Chat Widget side.
- Agent leaves the conference.
- Agent transfers the session to another agent.
- agent's desktop stops responding.
- Chat server stops responding and session is restored on the new chat session.

Event structure

Parameter	Description
`event.timestamp`	Number
`event.party.name`	String that represents the name of the agent leaving chat session.
`event.party.id`	String with ID of party in chat session.
`event.index`	Index of this message in chat transcript
`event.restored`	Optional. If present and true, it means that the event was restored during session restoration (in other words, was already reported to the consumer previously).

session.onMessageReceived

Executed when a new message appears in the chat transcript (or in the context of the "playback" process during chat session restoration).

> **Important**
>
> Messages sent by `session.sendMessage` are "returned back" via this event as well.

Event structure

Parameter	Description
`event.index`	0-based index of this message in chat session transcript
`event.timestamp`	Number
`event.content.text`	String with message added to the chat context

Parameter	Description

`event.content.type.url`
OR `event.content.type.text`
	One of these is true depending on message type. Usage example:
	`session.onMessageReceived(function(event) {`
	`  if (event.content.type.url) {`
	`  // this is "url" message`
	`  } else if (event.content.type.text) {`
	`  // this is "text" message`
	`  }`
	`});`

`event.party.id`	Theoretically, the agent name and visitor name could be the same (especially since the visitor may be the one filling out the visitor name). To handle this scenario, this ID is used to distinguish between agent and visitor.

`event.party.type.agent`
OR
`event.party.type.client`
OR
`event.party.type.external` OR
`event.party.type.supervisor`
	One of these is set to true, depending on who sends the message.
	• **agent** — message is sent by the agent;
	• **client** — message is sent by the visitor (in other words, the actual user of this API via `session.sendMessage()`)
	• **external** — message is sent by the system (for example, as configured in the routing strategy)
	• **supervisor** — message is sent by the supervisor

`event.party.name`	String with name of party.
`event.restored`	Optional. If present and true, it means that the event was restored during session restoration (in other words, event was already reported to the consumer).

session.onAgentTyping

Executed when agent starts, continues, or stops typing.

Event structure

Parameter	Description
`event.party.id`	ID of party in chat session.
`event.party.name`	Name of agent.
`event.isTyping`	• **true** — when agent starts (or continues) typing;

Parameter	Description
	• **false** — when agent stops typing.

session.onInterrupted

Executed when a connection to the server is interrupted (for example, a network interruption or server is down). If default transport is used, chat tries to automatically reconnect and fires `session.onContinued` when the connection is restored.

Callbacks receive no event object.

> **Important**
>
> This event duplicates the `session.onError` event with the "150" (network interruption) Error code. Genesys recommends that you use `session.onInterrupted/onContinued` if you want to track the connection status. The "150" error code is deprecated and might be removed in future versions.

session.onContinued

Executed only after `session.onInterrupted` when the connection to the server is restored.

Callbacks receive no event object.

session.onSessionEnded

Executed when the client drops out of a chat session due to one of the following reasons:

- Logout request.
- Chat session is finished from the agent's side.
- Chat Server stops responding, resulting in a timeout during which the session is not restored.

Parameters

Parameter	Description

`event.reason.error`
 	
OR
`event.reason.agent`
	One of these is set to true depending on the reason
	
	• **error** — a major error occurs (for example, Chat

Parameter	Description
OR `event.reason.leaveRequest`	Server stops responding) • **agent** — agent ends the session; • **leaveRequest** — visitor sends a session.leave command and it was successful

Miscellaneous methods

session.isAgentConnected

Provides a convenient way to synchronously determine if at least one agent is present in the session.

Returns boolean (true or false).