Service Client API Reference

System namespace

8/1/2025

Contents

Learn about the System namespace methods in the Service Client API.

> **Important**
> Depending on your environment, you might need to contact your Genesys representative to complete the configuration described on this page.

Methods

The System namespace includes the following methods:

- amIVisible
- closeDialog
- closeToast
- closeViewInApplicationMenuBar
- getAllowedServices
- isFrameLeading
- isFrameFollowing
- isFrameNegotiating
- isFrameLeadingOrNegotiating
- isLastActiveFrame
- popupToast
- openDialog
- triggerActivity
- updateViewInApplicationMenuBar
- updateToast

amIVisible

Signature	amIVisible(succeeded, failed) → {boolean}
Description	Get the current visibility state of the frame.

Signature	amIVisible(succeeded, failed) → {boolean}
Parameters	**Name** / **Type** / **Description** succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.
Returns	`true` if the frame is visible.

Sample request

```
setTimeout(function() {
    genesys.wwe.service.system.amIVisible(succeeded, failed);
}, 3000); // This gives 3 seconds to switch the panel to test.
```

Sample response

The asynchronous answer is included in the `data` attribute:

```
{
  "request": "system.amIVisible",
  "data": true,
  "userAgent": "WWE Server",
  "protocolVersion": 2
}
```

closeDialog

Signature	closeDialog(*dialogId*, *succeeded*, *failed*) → {boolean}
Description	Close a previously opened dialog.
Parameters	**Name** / **Type** / **Description** dialogId / string / The dialog identifier (returned in the response of openDialog). succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.

Signature	closeDialog(*dialogId*, *succeeded*, *failed*) → {boolean}
Returns	`true` if the dialog is closed; `false` if the dialog is not found.

closeToast

Signature	closeToast(*id*, succeeded, failed) → {boolean}
Description	Closes the specified toast.
Parameters	**Name** / **Type** / **Description** id / string / The identifier of the toast to close. The identifier is returned by the popupToast method. succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.
Returns	`true` if the toast has been updated; `false` if the toast identifier has not been found.

closeViewInApplicationMenuBar

Signature	closeViewInApplicationMenuBar(parameters, succeeded, failed) → {boolean}
Description	Removes the given view from the **Application Menu** bar region.
Parameters	**Name** / **Type** / **Description** name / string / The name of the custom view to be removed. succeeded / function / A function called when the operation succeeds. failed / function / A function

Signature	closeViewInApplicationMenuBar(parameters, succeeded, failed) → {boolean}
	Name / **Type** / **Description** called when the operation fails.
Returns	`true` if the view is removed; `false` if the view name is not found.

Sample request

```
genesys.wwe.service.system.closeViewInApplicationMenuBar("view1", succeeded, failed)
```

Sample response

```
{
   "request": "system.closeViewInApplicationMenuBar",
    "data": true,
    "userAgent": "WWE Server",
    "protocolVersion": 2
}
```

Sample request

```
genesys.wwe.service.system.closeDialog("wweCustomDialog1", succeeded, failed)
```

Sample response

The asynchronous answer is included in the `data` attribute:

```
{
    "request": "system.closeDialog",
    "data": true,
    "userAgent": "WWE Server",
    "protocolVersion": 2
}
```

getAllowedServices

Signature	getAllowedServices(succeeded, failed) → {Array.}
Description	Gets the list of allowed services, as determined by the Security configuration. If the domain of the web application that calls this method isn't listed in the service-client-api.accepted-web-content-origins option, then this method fails.
Parameters	**Name** / **Type** / **Description** succeeded / function / A function called when the operation succeeds.

Signature	getAllowedServices(succeeded, failed) → {Array.}
	Name / **Type** / **Description** failed / function / A function called when the operation fails.
Returns	Array.

isFrameLeading

Signature	isFrameLeading(*succeeded*, *failed*) → {boolean}
Description	Find out if the browser tab is leading.
Parameters	**Name** / **Type** / **Description** succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.
Returns	`true` if the browser tab is the leader.

isFrameFollowing

Signature	isFrameFollowing(*succeeded*, *failed*) → {boolean}
Description	Find out if the browser tab is following.
Parameters	**Name** / **Type** / **Description** succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.
Returns	`true` if this browser tab is following.

isFrameNegotiating

Signature	isFrameNegotiating(*succeeded*, *failed*) → {boolean}
Description	Find out if there is an election in progress and the browser tab state is not yet set to leading or following (the tab is "negotiating.")
Parameters	**Name** / **Type** / **Description** succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.
Returns	`true` if the tab is negotiating.

isFrameLeadingOrNegotiating

Signature	isFrameLeadingOrNegotiating(*succeeded*, *failed*) → {boolean}
Description	Find out if the browser tab is leading or there is an election in progress and the tab state is not yet set to leading or following (the tab is "negotiating.").
Parameters	**Name** / **Type** / **Description** succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.
Returns	`true` if the browser tab is leading or negotiating.

isLastActiveFrame

Signature	isLastActiveFrame(*succeeded*, *failed*) → {boolean}
Description	Find out if this is the last active browser tab.
Parameters	**Name** / **Type** / **Description** succeeded / function / A function called when

Signature	isLastActiveFrame(*succeeded*, *failed*) → {boolean}
	Name / **Type** / **Description** (unnamed) / (unnamed) / the operation succeeds. failed / function / A function called when the operation fails.
Returns	`true` if this is the last active browser tab.

openDialog

Signature	openDialog(*url*, *options*, *succeeded*, *failed*) → {string}
Description	Open an iframe in a dialog, based on the configured parameters.
Parameters	**Name** / **Type** / **Description** url / string / The URL of the iframe to load in the dialog. options / object / Optional parameters to configure the dialog. This value can't be null, so you must pass {} if there are no specific options. You can include any of the following options: • label - Set a custom value for the aria-label attribute on the dialog. When the dialog pops up, this value identifies it

<table>
<tr><th>Signature</th><td>openDialog(url, options, succeeded, failed) → {string}</td></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>to accessibility tools like screen readers.
width - The initial width of the dialog. Valid formats are px or %.
height - The initial height of the dialog. Valid formats are px or %.</td></tr>
<tr><td>succeeded</td><td>function</td><td>A function called when the operation succeeds.</td></tr>
<tr><td>failed</td><td>function</td><td>A function called when the operation fails.</td></tr>
</table>
</td></tr>
<tr><th>Returns</th><td>The dialog identifier or null if the <code>url</code> parameter is not defined.</td></tr>
</table>

Sample request

```
genesys.wwe.service.system.openDialog("", {
  label: "Dialog $Agent.FullName$",
  width: "430px",
  height: "325px"
}, succeeded, failed)
```

Sample response

The asynchronous answer is included in the `data` attribute:

```
{
    "request": "system.openDialog",
    "data": "wweCustomDialog1",
```

```
    "userAgent": "WWE Server",
    "protocolVersion": 2
}
```

popupToast

Signature	popupToast(*parameters*, succeeded, failed) → {string}
Description	Pops up a new custom toast.
Parameters	**Name** / **Type** / **Description** parameters / object / **Name** / **Type** / **Description** title / string / The title iconUrl / string / The URL of the icon you want to display in the title bar of the custom toast popup. subject / string / Optional. The subject message / string / Optional. The message keyValues / string / Optional. JSON object used to fill the key value pair list. For example:

Signature	popupToast(*parameters*, succeeded, failed) → {string}

Name	Type	Description

Name	Type	Description
		{"key1" ; "value one","key2" ; "value two","key3" ; "value three"}.
buttons	Array	Optional. Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1].
buttonSheet	boolean	Optional. If set to true, displays the **Show** and **Dismiss**

Signature	popupToast(*parameters*, succeeded, failed) → {string}

Name	Type	Description

Name	Type	Description
		buttons and pops up the current iframe if the **Show** button is pushed. If set to `false`, displays '"OK"' or custom buttons based on the parameter's buttons.
autoClose	Object	Optional. If set to greater than `0`, the popup is automatically closed after the specified milliseconds.
sendToMe	Object	Optional. If set to `true`, the message

<table>
<tr><th>Signature</th><td>popupToast(parameters, succeeded, failed) → {string}</td></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td><code>true</code>, sends the subject, iconUrl, title, keyValues, and message parameters to the MyMessage panel.</td></tr>
<tr><td>width</td><td>number</td><td>Optional. The width of the custom toast popup, in pixels. This values takes precedence over the service-client-api.toast.width configuration option.</td></tr>
</table>
</td></tr>
<tr><td>succeeded</td><td>function</td><td>A function called when the operation succeeds.</td></tr>
<tr><td>failed</td><td>function</td><td>A function called when the operation fails.</td></tr>
</table>
</td></tr>
<tr><th>Returns</th><td>A unique identifier</td></tr>
</table>

triggerActivity

Signature	triggerActivity(succeeded, failed)
Description	Triggers a fake activity to prevent the inactivity timer from closing the agent session.
Parameters	

Name	Type	Description
succeeded	function	A function called when the operation succeeds.
failed	function	A function called when the operation fails.

updateViewInApplicationMenuBar

Signature	updateViewInApplicationMenuBar(parameters, succeeded, failed) → {string}
Description	Creates a custom view in the **Application Menu bar** region.
Parameters	

Name	Type	Description
name	string	A unique name for the custom view of the **Application Menu bar** that is to be created or updated. If a view with the given name already exists, it will be updated, otherwise, a new view will be created.
iconUrl	string	The URL of the icon you want to display in the custom view of the **Application Menu bar** region. This parameter is

Signature	updateViewInApplicationMenuBar(parameters, succeeded, failed) → {string}

Name	Type	Description
		mandatory if **label** is not provided.
label	string	The main textual content to be displayed for the custom view. This parameter is mandatory if **iconUrl** is not provided.
shortLabel	string	Optional. A shorter version of the **label** that will be used in the shortened mode if **iconUrl** is not available.
tooltip	string	Optional. The tooltip content to be shown when the mouse is hovered on the custom view.
labelColor	string	Optional. The color of the label text in case-insensitive hex color code format, for example, #FFFFF.
backgroundColor	string	Optional. The background color of the region where icon and title are displayed. The format of the background color is case-

Signature	updateViewInApplicationMenuBar(parameters, succeeded, failed) → {string}
	Name / **Type** / **Description** insensitive hex color code, for example, #FFFFF. By default, it is usually the same color as the navigation bar. succeeded / function / A function called when the operation succeeds. failed / function / A function called when the operation fails.
Returns	View name if successful.

Sample request

```
genesys.wwe.service.system.updateViewInApplicationMenuBar({
    name: "view1",
    iconUrl: "https://cdn1.iconfinder.com/data/icons/free-social-media-12/32/
RSS_social_media-128.png",
    label: "Main content text",
    shortLabel: "Short text"
    tooltip: "Tooltip text",
    labelColor: "#FFFFF",
    backgroundColor: "#00000"
}, succeeded, failed)'
```

Sample response

```
{
    "request": "system.updateViewInApplicationMenuBar",
    "data": "view1",
    "userAgent": "WWE Server",
    "protocolVersion": 2
}
```

updateToast

Signature	updateToast(*id*, *parameters*, succeeded, failed) → {boolean}
Description	Updates the specified toast.

<table>
<tr><th>Signature</th><td>updateToast(id, parameters, succeeded, failed) → {boolean}</td></tr>
<tr><th>Parameters</th><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>id</td><td>string</td><td>The identifier of the toast to update. The identifier is returned by the popupToast method.</td></tr>
<tr><td>parameters</td><td>object</td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>title</td><td>string</td><td>The title</td></tr>
<tr><td>iconURL</td><td>URL string</td><td>The URL of the icon you want to display in the title bar of the custom toast popup.</td></tr>
<tr><td>subject</td><td>string</td><td>Optional. The subject.</td></tr>
<tr><td>message</td><td>string</td><td>Optional. The message.</td></tr>
<tr><td>keyValues</td><td>object</td><td>Optional. JSON object used to fill the key value pair list. For</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Signature	updateToast(*id*, *parameters*, succeeded, failed) → {boolean}

Name	Type	Description
		Name **Type** **Description** example: {"key1" : "value one","key2" : "value two","key3" : "value three"}. buttons Array Each character string in this array becomes a button. All buttons are displayed as buttons, not hyperlinks, in the following order: [Button 2] [Button 3] ... [Button N] [Button 1]. buttonsShown boolean If set to `true`, displays **Show** and **Dismiss** buttons and

<table>
<tr><th>Signature</th><td>updateToast(id, parameters, succeeded, failed) → {boolean}</td></tr>
<tr><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>pops up the current iframe if the Show button is pushed. If set to <code>false</code>, displays '"OK"' or custom buttons based on the parameter's buttons.</td></tr>
</table>
</td></tr>
<tr><td>succeeded</td><td>function</td><td>A function called when the operation succeeds.</td></tr>
<tr><td>failed</td><td>function</td><td>A function called when the operation fails.</td></tr>
</table>
</td></tr>
<tr><th>Returns</th><td><code>true</code> if the toast has been updated; <code>false</code> if the toast identifier has not been found.</td></tr>
</table>