# Salesforce Console Integration Toolkit Developer's Guide

Version 35.0, Winter '16

@salesforcedocs

Last updated: December 17, 2015

# CONTENTS

# CHAPTER 1 Introducing the Salesforce Console Integration Toolkit

The Salesforce console is designed for users in fast-paced environments who need to find, update, and create records in Salesforce quickly. The Salesforce Console Integration Toolkit provides you with programmatic access to the Salesforce console so that you can extend it to meet your business needs. With the Salesforce Console Integration Toolkit, you can open and close tabs in the console to streamline a business process. For example, the toolkit lets you integrate third-party systems with the console, opening up an external application in the same window, in a tab.

To use this guide, it helps if you have a basic familiarity with:

- JavaScript
- Visualforce
- Web services
- Software development
- The Salesforce console

This guide explains how to use the Salesforce Console Integration Toolkit in JavaScript to embed API calls and processes. The toolkit is available for use with third-party domains, such as `www.yourdomain.com`; however, the examples in this guide are in Visualforce. The functionality it describes is available to your organization if you have:

- Enterprise, Unlimited, Performance, or Developer Edition with the Service Cloud
- The Salesforce console

The Salesforce Console Integration Toolkit is a browser-based JavaScript API. It uses browsers as clients to display pages as tabs in the console. The Salesforce Console Integration Toolkit:

- Matches the API version for any given release. For example, if the current version of SOAP API is 20.0, then there's also a version 20.0 of the Salesforce Console Integration Toolkit.
- Supports any browser that the Salesforce console supports. For more information, see “Salesforce Console” in the Salesforce online help.

Note: To enable the toolkit for third-party domains, add the domains to the whitelist of the Salesforce console. See “Whitelist Domains for a Salesforce Console” in the Salesforce online help.

IN THIS SECTION:

Sample Visualforce Page Using the Salesforce Console Integration Toolkit
This example shows how to change the Salesforce console user interface using the Salesforce Console Integration Toolkit.

## When to Use the Salesforce Console Integration Toolkit

The Salesforce Console Integration Toolkit helps advanced administrators and developers implement custom functionality for the Salesforce console. For example, you can use the Salesforce Console Integration Toolkit to display Visualforce pages or third-party content as tabs in the Salesforce console. The Salesforce Console Integration Toolkit is an API that uses browsers as clients to display pages in the console.

The following table lists additional features that developers can use to implement custom functionality for Salesforce organizations.

| Feature | Description |
|---|---|
| SOAP API | Use standard SOAP API calls if you want to add functionality to a composite application that processes only one type of record at a time and does not require any transactional control (such as setting a Savepoint or rolling back changes).<br>For more information, see the *SOAP API Developer's Guide*. |
| Visualforce | Visualforce consists of a tag-based markup language that gives developers a more powerful way of building applications and customizing the Salesforce user interface. With Visualforce you can:<br>• Build wizards and other multistep processes.<br>• Create your own custom flow control through an application.<br>• Define navigation patterns and data-specific rules for optimal, efficient application interaction.<br>For more information, see the *Visualforce Developer's Guide*. |
| Apex | Use Apex if you want to:<br>• Create Web services.<br>• Create email services.<br>• Perform complex validation over multiple objects.<br>• Create complex business processes that are not supported by workflow.<br>• Create custom transactional logic (logic that occurs over the entire transaction, not just with a single record or object).<br>• Attach custom logic to another operation, such as saving a record, so that it occurs whenever the operation is executed, regardless of whether it originates in the user interface, a Visualforce page, or from SOAP API.<br>For more information, see the *Force.com Apex Code Developer's Guide*. |

## Salesforce Console Integration Toolkit Support Policy

The current release of the Salesforce Console Integration Toolkit is the only version that receives enhancements. Previous versions may or may not receive fixes. When a new version is released, the previous version remains available.

IN THIS SECTION:

## Backward Compatibility

Salesforce strives to make backward compatibility easy when using the Salesforce Console Integration Toolkit.

Each new Salesforce release consists of two components:

- A new release of platform software that resides on Salesforce systems
- A new version of the API

For example, the Summer '10 release included API version 19.0 and the Winter '11 release included API version 20.0.

The version of the Salesforce Console Integration Toolkit matches the API version for any given release. So if the current version of the API is 20.0, there's also a version 20.0 of the Salesforce Console Integration Toolkit.

We maintain support for each Salesforce Console Integration Toolkit version across releases of the platform. The Salesforce Console Integration Toolkit is backward compatible in that an application created to work with a given Salesforce Console Integration Toolkit version will continue to work with that same Salesforce Console Integration Toolkit version in future platform releases.

Salesforce doesn't guarantee that an application written against one Salesforce Console Integration Toolkit version will work with future Salesforce Console Integration Toolkit versions: Changes in method signatures and data representations are often required as we continue to enhance the Salesforce Console Integration Toolkit. However, we strive to keep the Salesforce Console Integration Toolkit consistent from version to version with minimal changes required to port applications to newer Salesforce Console Integration Toolkit versions.

For example, an application written using Salesforce Console Integration Toolkit version 20.0, which shipped with the Winter '11 release, will continue to work with Salesforce Console Integration Toolkit version 20.0 on the Summer '11 release and on future releases. However, that same application may not work with Salesforce Console Integration Toolkit version 21.0 without modifications to the application.

## End-of-Life

Salesforce is committed to supporting each Salesforce Console Integration Toolkit version for a minimum of three years from the date of its first release. To improve the quality and performance of the Salesforce Console Integration Toolkit, versions that are more than three years old may not be supported.

When a Salesforce Console Integration Toolkit version is scheduled to be unsupported, an advance end-of-life notice will be given at least one year before support for the version ends. Salesforce will directly notify customers using Salesforce Console Integration Toolkit versions scheduled for end of life.

## Other Resources

In addition to this guide, there are other resources available for you as you learn how to use the Salesforce Console Integration Toolkit.

- Online help: See *Salesforce Console*
- Developer website: `https://developer.salesforce.com/gettingstarted`
- Firebug extension to Firefox: Firebug for Firefox

- Eclipse plug-in: Force.com IDE

Note: Salesforce Education Services offers a suite of training courses to enable developers to design, create, integrate, and extend applications built on the Force.com platform. Be sure to visit http://www.salesforce.com/training to learn more.

## Salesforce Console Integration Toolkit Typographical Conventions

The Salesforce Console Integration Toolkit guide uses a few typographical conventions.

| Convention | Description |
|---|---|
| `Courier font` | In descriptions of syntax, monospace font indicates items that you should type as shown, except for brackets. For example:<br>`Public class HelloWorld` |
| *`Italics`* | In descriptions of syntax, italics represent variables. You supply the actual value. In the following example, three values need to be supplied: *datatype variable_name* [= *value*];<br>If the syntax is bold and italic, the text represents a code element that needs a value supplied by you, such as a class name or variable value:<br>`public static class` ***YourClassHere*** `{ ... }` |
| **`Bold Courier font`** | In code samples and syntax descriptions, bold courier font emphasizes a portion of the code or syntax. |
| < > | In descriptions of syntax, less-than and greater-than symbols (< >) are typed exactly as shown.<br>`<apex:pageBlockTable value="{!account.Contacts}" var="contact">`<br><br>`   <apex:column value="{!contact.Name}"/>`<br>`   <apex:column value="{!contact.MailingCity}"/>`<br>`   <apex:column value="{!contact.Phone}"/>`<br>`</apex:pageBlockTable>` |
| { } | In descriptions of syntax, braces ({ }) are typed exactly as shown.<br>`<apex:page>`<br>`    Hello {!$User.FirstName}!`<br>`</apex:page>` |
| [ ] | In descriptions of syntax, anything included in brackets is optional. In the following example, specifying ***value*** is optional:<br>***data_type variable_name*** `[ =` ***value***`];` |

| Convention | Description |
|---|---|
| \| | In descriptions of syntax, the pipe sign means "or". You can do one of the following (not all). In the following example, you can create a new unpopulated set in one of two ways, or you can populate the set: `Set<data_type> set_name [= new Set<data_type>();] \| [= new Set<data_type{value [, value2. . .] };] \| ;` |

## Sample Visualforce Page Using the Salesforce Console Integration Toolkit

This example shows how to change the Salesforce console user interface using the Salesforce Console Integration Toolkit.

1. Create a Visualforce page. See the *Visualforce Developer's Guide*.
2. Cut and paste the following sample code into your Visualforce page.

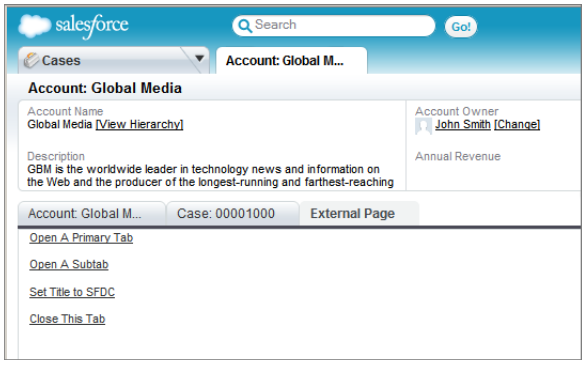
   This code demonstrates various functions of the Salesforce Console Integration Toolkit:

```
<apex:page standardController="Case">

   <apex:includeScript value="/support/console/20.0/integration.js"/>
   <script type="text/javascript">
        function openPrimaryTab() {
            sforce.console.openPrimaryTab(undefined,
               'http://www.salesforce.com', true, 'salesforce');
        }

        //The callback function that openSubtab will call once it's got the ID for its
 primary tab
        var callOpenSubtab=function callOpenSubtab(result) {
            sforce.console.openSubtab(result.id,
               'http://www.yahoo.com', true, 'yahoo');
        };

        function openSubtab() {
            sforce.console.getEnclosingPrimaryTabId(callOpenSubtab);
        }

        //Sets the title of the current tab to "SFDC"
        function setTitle() {
            sforce.console.setTabTitle('SFDC');
        }

        //The callback function that closeTab will call once it's got the ID for its
tab
        var callCloseTab= function callCloseTab(result) {
            sforce.console.closeTab(result.id);
        }
        function closeTab() {
            sforce.console.getEnclosingTabId(callCloseTab);
        }
  </script>

   <A HREF="#" onClick="openPrimaryTab();return false">Open A Primary Tab</A>
   <p/><A HREF="#" onClick="openSubtab();return false">Open A Subtab</A>
   <p/><A HREF="#" onClick="setTitle();return false">Set Title to SFDC</A>
   <p/><A HREF="#" onClick="closeTab();return false">Close This Tab</A>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

After you create the above Visualforce page and add it as a custom link on cases, this page displays after you navigate to a case and click the link:

**Output of Sample Visualforce Page**



Account: Global Media

Account Name
Global Media [View Hierarchy]

Account Owner
John Smith [Change]

Description
GBM is the worldwide leader in technology news and information on the Web and the producer of the longest-running and farthest-reaching

Annual Revenue

Open A Primary Tab

Open A Subtab

Set Title to SFDC

Close This Tab

# CHAPTER 2 Working with the Salesforce Console Integration Toolkit

Use the Salesforce Console Integration Toolkit to do the following in the Salesforce console.

- Open a new primary tab or subtab that displays a specified URL
- Set the title of a primary tab or a subtab
- Return the ID of a primary tab or subtab
- Close a specified primary tab or subtab

IN THIS SECTION:

## Connecting to the Toolkit

The first portion of any JavaScript code that uses the Salesforce Console Integration Toolkit must make the toolkit available to the JavaScript code. The syntax for this is different depending on whether you are embedding JavaScript in a Visualforce page, or a third-party domain.

- For Visualforce pages or any source other than a custom `onclick` JavaScript button, specify a `<script>` tag that points to the toolkit file:

```
<apex:page>
          <script src="/support/console/35.0/integration.js"
type="text/javascript"></script>
     ...
</apex:page>
```

  For Visualforce, a relative path is sufficient to include `integration.js`, and is recommended.

- For a third-party domain:

  ```
  <script src="https://c.na1.visual.force.com/support/console/35.0/integration.js"
  type="text/javascript"></script>
  ```

  For third-party domains, it is necessary to specify an absolute URL to `integration.js` to use the toolkit. The default instance at which you can access the toolkit library is: `c.na1.visual.force.com/support/console/35.0/integration.js`. We recommend that you use the default instance when the organization's instance cannot be determined.

The version of the Salesforce Console Integration Toolkit is in the URL.

## Asynchronous Calls with the Salesforce Console Integration Toolkit

The Salesforce Console Integration Toolkit lets you issue asynchronous calls. Asynchronous calls allow the client-side process to continue instead of waiting for a callback from the server. To issue an asynchronous call, you must include an additional parameter with the API call, which is referred to as a callback function. Once the result is ready, the server invokes the callback method with the result.

Asynchronous syntax:

```
method('arg1','arg2', ..., callback_method);
```

For example:

```
//Open a new primary tab with the Salesforce home page in it
   sforce.console.openPrimaryTab(null, 'http://www.salesforce.com',
      false, 'Salesforce', callback);
```

## Working with Force.com Canvas

To integrate the Salesforce Console with external applications that require authentication methods, such as signed requests or OAuth 2.0 protocols, Salesforce recommends you use Force.com Canvas.

Force.com Canvas and the Salesforce Console Integration Toolkit are similar—they're a set of tools and JavaScript APIs that developers can use to add third-party systems to Salesforce. However, one of the benefits of Force.com Canvas, is the ability to choose authentication methods. For more information, see the *Force.com Canvas Developer's Guide*.

Note: For a canvas app to appear in a console, you must add it to the console as a custom console component. See Add Console Components to Apps.

When developing a canvas app, and you want to include functionality from the Salesforce Console Integration Toolkit, do the following:

1. Include the console integration toolkit API in `index.jsp`.
2. If your console has a whitelist for domains, add the domain of your canvas app to the whitelist. See "Whitelist Domains for a Salesforce Console" in the Salesforce Help.
3. Call `Sfdc.canvas.client.signedrequest()` to store the signed request needed by the console integration toolkit API. For example, if the Force.com Canvas method of authentication is a signed request, do the following:

   ```
   Sfdc.canvas.client.signedrequest('<%=signedRequest%>')
   ```

If the Force.com Canvas method of authentication is OAuth, do the following in the callback function used to get the context as shown in "Getting Context in Your Canvas App" in the *Force.com Canvas Developer's Guide*:

```
Sfdc.canvas.client.signedrequest(msg)
```

Consider the following when working with the Salesforce Console Integration Toolkit and canvas apps:

- The console integration toolkit API script depends on the signed request and should be added after the call to `Sfdc.canvas.client.signedrequest()` has executed. We recommend that you load the scripts dynamically.
- To retrieve the entity ID of the record that is associated with the canvas sidebar component, do the following:

```
// Get signedRequest
var signedRequest = Sfdc.canvas.client.signedrequest();
var parsedRequest = JSON.parse(signedRequest);
// get the entity Id that is associated with this canvas sidebar component.
var entityId = parsedRequest.context.environment.parameters.entityId;
```

- To retrieve the `entityId` for OAuth, do the following:

```
var entityId = msg.payload.environment.parameters.entityId;
```

  To see an example on how to retrieve `msg.payload`, see "Getting Context in Your Canvas App" in the *Force.com Canvas Developer's Guide*.

## Best Practices

Salesforce recommends that you adhere to a few best practices as you use the Salesforce Console Integration Toolkit.

- Many of the methods in the Salesforce Console Integration Toolkit are asynchronous and return their results using a callback method. We recommend that you refer to the documentation for each method to understand the information for each response.
- Errors generated by the Salesforce Console Integration Toolkit are typically emitted in a way that doesn't halt JavaScript processing. Therefore, we recommend that you use a tool such as Firebug for Firefox to monitor the JavaScript console and to help you debug your code.
- To display Visualforce pages properly in the Salesforce Console, we recommend you:
  - Accept the default setting `showHeader="true"` and set `sidebar="false"` on the `apex:page` tag.
  - Set `Behavior` on custom buttons and links that include methods from the toolkit to display in an existing window without a sidebar or header. For more information, see Define Custom Buttons and Links" in the Salesforce online help.
- When using Firefox, we recommend that you don't call `closeTab()` on a tab with an active alert box because the browser may not load properly.
- Duplicate tabs might open when users initiate methods with invalid URLs. We recommend that you check URLs for validity before you include them in methods.
- To prevent `External Page` from displaying as a tab name, we recommend that you specify the `tabLabel` argument on methods such as `openPrimaryTab()` and `openSubtab()`.
- For information on how you can customize, extend, or integrate the sidebars of the Salesforce console using Visualforce, see "Console Components" in the Salesforce online help.
- To enable the toolkit for third-party domains, add the domains to the whitelist of the Salesforce console. See "Whitelist Domains for a Salesforce Console" in the Salesforce online help.
- When working with the Salesforce Console Integration Toolkit, we recommend that you keep in mind that it doesn't support nested iframes.

# CHAPTER 3 Methods for Primary Tabs and Subtabs

A Salesforce console displays Salesforce pages as primary tabs or subtabs. A primary tab displays the main item to work on, such as an account. A subtab displays related items, such as an account's contacts or opportunities.

IN THIS SECTION:

getFocusedPrimaryTabId()
Returns the ID of the primary tab on which the browser is focused. This method is only available in API version 25.0 or later.

getFocusedPrimaryTabObjectId()
Returns the object ID of the primary tab on which the browser is focused. This method is only available in API version 25.0 or later.

getFocusedSubtabId()
Returns the ID of the subtab on which the browser is focused. For example, a case ID or account ID. This method is only available in API version 25.0 or later.

getFocusedSubtabObjectId()
Returns the object ID of the subtab on which the browser is focused. For example, a case ID or account ID. This method is only available in API version 24.0 or later.

getPageInfo()
Returns page information for the specified tab after its content has loaded. If the tab ID is null, it returns page information for the enclosing primary tab or subtab. Note that to get the page information from a custom console component, a `tabId` must be passed as the first parameter to this method.This method is only available in API version 26.0 or later.

getPrimaryTabIds()
Returns all of the IDs of open primary tabs. This method is only available in API version 26.0 or later.

getSubtabIds()
Returns all of the IDs of the subtabs on the primary tab specified by a primary tab ID. If the primary tab ID is null, it returns the IDs of the subtabs on the current primary tab. This method can only be called from a custom console component or a detail page overwritten by a Visualforce page. This method is only available in API version 26.0 or later.

getTabLink()
Retrieves the URL to a tab, or group of related tabs, from the Salesforce console. This method is only available in API version 28.0 or later.

isInConsole()
Determines if the page is in the Salesforce console. This method is only available in API version 22.0 or later.

onEnclosingTabRefresh()
Registers a function to call when the enclosing tab refreshes. This method is only available in API version 24.0 or later.

onFocusedSubtab()
Registers a function to call when the focus of the browser changes to a different subtab. This method is only available in API version 24.0 or later.

onTabSave()
Registers and calls a callback method when a user clicks **Save** in a subtab's Unsaved Changes dialog box. When using this method, you must call `setTabUnsavedChanges()` in the callback method. This notifies the console that the custom save operation completed. In the call to `setTabUnsavedChanges()`, pass the first parameter as `false` to indicate a successful save or `true` to indicate an unsuccessful save. This method is only available in API version 28.0 or later.

openConsoleUrl()
Opens a URL created by the `generateConsoleUrl()` method (a URL to a tab, or group of related tabs, in the Salesforce console). This method is only available in API version 28.0 or later.

openPrimaryTab()
Opens a new primary tab to display the content of the specified URL, which can be relative or absolute. You can also override an existing tab. This method is only available in API version 20.0 or later.

openSubtab()
Opens a new subtab (within a primary tab) that displays the content of a specified URL, which can be relative or absolute. You can also override an existing subtab. Use to open a new subtab on a primary tab via the primary tab's ID. This method is only available in API version 20.0 or later.

openSubtabByPrimaryTabName()
Opens a new subtab (within a primary tab) that displays the content of a specified URL, which can be relative or absolute. You can also override an existing subtab. Use to open a new subtab on a primary tab via the primary tab's name. This method is only available in API version 22.0 or later.

refreshPrimaryTabById()
Refreshes a primary tab specified by ID, including its subtabs. This method can't refresh subtabs with URLs to external pages or Visualforce pages. This method is only available in API version 22.0 or later.

refreshPrimaryTabByName()
Refreshes a primary tab specified by name, including its subtabs. This method can't refresh subtabs with URLs to external pages or Visualforce pages. This method is only available in API version 22.0 or later.

refreshSubtabById()
Refreshes a subtab with the last known URL with a specified ID. This method can't refresh a subtab If the last known URL is an external page or a Visualforce page. This method is only available in API version 22.0 or later.

refreshSubtabByNameAndPrimaryTabId()
Refreshes a subtab with the last known URL with the specified name and primary tab ID. This method can't refresh a subtab If the last known URL is an external page or a Visualforce page. This method is only available in API version 22.0 or later.

refreshSubtabByNameAndPrimaryTabName()
Refreshes a subtab with the last known URL with the specified name and primary tab name. This method can't refresh a subtab If the last known URL is an external page or a Visualforce page. This method is only available in API version 22.0 or later.

reopenLastClosedTab()
Reopens the last closed primary tab, and any of its subtabs that were open, the moment it was closed. This method is only available in API version 35.0 or later.

resetSessionTimeOut()
Resets a session timeout on a Visualforce page so that users can continue working without being logged out. This method is only available in API version 24.0 or later.

setTabUnsavedChanges()
Sets the unsaved changes icon ( * ) on subtabs to indicate unsaved data. This method is only available in API version 23.0 or later.

setTabIcon()
Sets an icon on the specified tab. If a tab is not specified, the icon is set on the enclosing tab. Use this method to customize a tab’s icon. This method is only available in API version 28.0 or later.

setTabLink()
Sets a console tab’s URL attribute to the location of the tab’s content. Use this method to generate secure console URLs when users navigate to tabs displaying content outside of the Salesforce domain.This method is only available in API version 28.0 or later.

setTabStyle()
Sets a cascading style sheet (CSS) on the specified tab. If a tab is not specified, the CSS is set on the enclosing tab. Use this method to customize a tab’s look and feel. This method is only available in API version 28.0 or later.

setTabTextStyle()
Sets a cascading style sheet (CSS) on a specified tab’s text. If a tab is not specified, the CSS is set on the enclosing tab’s text. Use this method to customize a tab’s text style. This method is only available in API version 28.0 or later.

setTabTitle()
Sets the title of a primary tab or subtab. This method is only available in API version 20.0 or later.

## closeTab()

Closes a specified primary tab or subtab. Note that closing the first tab in a primary tab closes the primary tab itself. This method is only available in API version 20.0 or later.

### Syntax

```
sforce.console.closeTab(id:String, (optional) callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the primary tab or subtab to close. |
| `callback` | function | For API version 35.0 or later, the JavaScript method that's called upon completion of the method. |

### Sample Code API 20.0 or Later–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testCloseTab();return false">
           Click here to close this tab</A>

    <apex:includeScript value="/support/console/20.0/integration.js"/>
    <script type="text/javascript">
        function testCloseTab() {
            //First find the ID of the current tab to close it
            sforce.console.getEnclosingTabId(closeSubtab);
        }

        var closeSubtab = function closeSubtab(result) {
            //Now that we have the tab ID, we can close it
            var tabId = result.id;
            sforce.console.closeTab(tabId);
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

### Response

None

## Sample Code API Version 35.0 or Later–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testCloseTab();return false">
           Click here to close this tab</A>

    <apex:includeScript value="/support/console/35.0/integration.js"/>
    <script type="text/javascript">
         var callback = function () {
             if (result.error) {
          alert("Error message is " + result.error);
             }
         };
        function testCloseTab() {
            //First find the ID of the current tab to close it
            sforce.console.getEnclosingTabId(closeSubtab);
         }

        var closeSubtab = function closeSubtab(result) {
            //Now that we have the tab ID, we can close it
            var tabId = result.id;
            sforce.console.closeTab(tabId, callback);
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the tab was re-opened, `false` otherwise. |
| `error` | string | Error message if the tab couldn't be closed. |

Tip: When using Firefox, we recommend that you don't call `closeTab()` on a tab with an active alert box because the browser may not load properly.

## `focusPrimaryTabById()`

Focuses the browser on a primary tab that is already open with the specified ID. This method is only available in API version 22.0 or later.

## Syntax

```
sforce.console.focusPrimaryTabById(id:String, (optional)callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| id | string | ID of the primary tab to go to. |
| callback | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testFocusPrimaryTabById();return false">
         Click here to go to an open primary tab by id</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testFocusPrimaryTabById() {
            //Get the value for 'scc-pt-0' from the openPrimaryTab method
            //This value is for example purposes only
            var primaryTabId = 'scc-pt-0';
            sforce.console.focusPrimaryTabById(primaryTabId, focusSuccess);
        }

        var focusSuccess = function focusSuccess(result) {
            //Report whether going to the open primary tab was successful
            if (result.success == true) {
                alert('Going to the primary tab was successful');
            } else {
                alert('Going to the primary tab was not successful');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if going to the primary tab was successful; `false` if going to the primary tab wasn't successful. |

## `focusPrimaryTabByName()`

Focuses the browser on a primary tab that is already open with the specified name. This method is only available in API version 22.0 or later.

### Syntax

```
sforce.console.focusPrimaryTabByName(name:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `name` | string | Name of the primary tab to go to. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testFocusPrimaryTabByName();return false">
         Click here to go to a primary tab by name</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testFocusPrimaryTabByName() {
            //Get the value for 'myPrimaryTab' from the openPrimaryTab method
            //This value is for example purposes only
            var primaryTabName = 'myPrimaryTab';
            sforce.console.focusPrimaryTabByName(primaryTabName, focusSuccess);
        }

        var focusSuccess = function focusSuccess(result) {
            //Report whether going to the primary tab was successful
            if (result.success == true) {
                alert('Going to the primary tab was successful');
            } else {
                alert('Going to the Primary tab was not successful');
            }
        };

  </script>
```

```
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if going to the primary tab was successful; `false` if going to the primary tab wasn't successful. |

## `focusSidebarComponent()`

Focuses the browser on a sidebar component. Use this method to focus on a component with the tab or accordion sidebar style. For more information, see “Sidebar Styles for Console Components” in the Salesforce Help. This method is only available in API version 34.0 or later.

### Syntax

```
sforce.console.focusSidebarComponent(componentInfo:string (optional)tabId:string,
callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `componentInfo` | string | The JSON object that represents the component to focus on. This argument must include one of the following forms:<br>Unambiguous types:<br>• `{componentType: 'CASE_EXPERT_WIDGET' }`<br>• `{componentType: 'FILES_WIDGET' }`<br>• `{componentType: 'HIGHLIGHTS_PANEL' }`<br>• `{componentType: 'KNOWLEDGE_ONE'}`<br>• `{componentType: 'MILESTONE_WIDGET' }`<br>• `{componentType: 'TOPICS_WIDGET' }`<br>• `{componentType: 'VISUALFORCE' }`<br>Types that require additional parameters: |

| Name | Type | Description |
|---|---|---|
| | | • `{componentType: 'CANVAS', canvasAppId: '09Hxx0000000001'}` • `{componentType: 'RELATED_LIST', listName: 'Solution'}` • `{componentType: 'LOOKUP', fieldName: 'Account'}` • `{componentType: 'VISUALFORCE', pageName: 'VF1'}` |
| `tabId` | string | The ID of the tab on which to focus the browser. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/34.0/integration.js"/>
    <script type="text/javascript">

        var callback = function (result) {}
            if(result.success){
            alert('Congratulations!');
        }else{
            alert('Something is wrong!');
        }
    };
        function focusKnowledgeComponent() {
            sforce.console.focusSidebarComponent(JSON.stringify({componentType:
            'KNOWLEDGE_ONE'}),"scc-st-2", callback);
        }
    </script>
   <A HREF="#" onClick="focusSidebarComponent(); return false">Focus Knowledge Component</A>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if focusing the sidebar component was successful; `false` otherwise. |

## `focusSubtabById()`

Focuses the browser on a subtab that is already open with the specified ID. This method is only available in API version 22.0 or later.

## Syntax

```
sforce.console.focusSubtabById(id:String, (optional)callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the subtab to go to. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testFocusSubtabById();return false">
         Click here to go to a subtab by id</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testFocusSubtabById() {
            //Get the value for 'scc-st-0' from the openSubtab method
            //This value is for example purposes only
            var subtabId = 'scc-st-0';
            sforce.console.focusSubtabById(subtabId, focusSuccess);
        }

        var focusSuccess = function focusSuccess(result) {
            //Report whether going to the subtab was successful
            if (result.success == true) {
                alert('Going to the subtab was successful');
            } else {
                alert('Going to the subtab was not successful');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if going to the subtab was successful; `false` if going to the subtab wasn't successful. |

## `focusSubtabByNameAndPrimaryTabId()`

Focuses the browser on a subtab that is already open with the specified name and primary tab ID. This method is only available in API version 22.0 or later.

### Syntax

```
sforce.console.focusSubtabByNameAndPrimaryTabId(name:String,
primaryTabId:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `name` | string | Name of the subtab to go to. |
| `primaryTabId` | string | ID of the primary tab in which the subtab opened. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testFocusSubtabByNameAndPrimaryTabId();return false">
         Click here to go to a subtab by name and primary tab ID</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testFocusSubtabByNameAndPrimaryTabId() {
            //Get the values for 'mySubtab' and 'scc-pt-0' from the openSubtab method
            //These values are for example purposes only
            var subtabName = 'mySubtab';
            var primaryTabId = 'scc-pt-0';
            sforce.console.focusSubtabByNameAndPrimaryTabId(subtabName, primaryTabId,
focusSuccess);
        }

        var focusSuccess = function focusSuccess(result) {
            //Report whether going to the subtab was successful
            if (result.success == true) {
                alert('Going to the subtab was successful');
            } else {
                alert('Going to the subtab was not successful');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if going to the subtab was successful; `false` if going to the subtab wasn't successful. |

## `focusSubtabByNameAndPrimaryTabName()`

Focuses the browser on a subtab that is already open with the specified name and primary tab name. This method is only available in API version 22.0 or later.

### Syntax

```
sforce.console.focusSubtabByNameAndPrimaryTabName(name:String,
primaryTabName:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `name` | string | Name of the subtab to go to. |
| `primaryTabName` | string | Name of the primary tab in which the subtab opened. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testFocusSubtabByNameAndPrimaryTabName();return false">
         Click here to go to a subtab by name and primary tab name</A>
```

```
    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testFocusSubtabByNameAndPrimaryTabName() {
            //Get the value for 'mySubtab' and 'myPrimaryTab' from the openSubtab method
            //These values are for example purposes only
            var subtabName = 'mySubtab';
            var primaryTabName = 'myPrimaryTab';
            sforce.console.focusSubtabByNameAndPrimaryTabName(subtabName, primaryTabName,
 focusSuccess);
        }

        var focusSuccess = function focusSuccess(result) {
            //Report whether going to the subtab was successful
            if (result.success == true) {
                alert('Going to the subtab was successful');
            } else {
                alert('Going to the subtab was not successful');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if going to the subtab was successful; `false` if going to the subtab wasn't successful. |

## `generateConsoleUrl()`

Generates a URL to a tab, or group of related tabs, in the Salesforce console. If any tabs include external URLs, then add the external URLs to the console's whitelist so that they can display correctly. For more information, see "Whitelist Domains for a Salesforce Console" in the online help. This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console.generateConsoleUrl(urls:String, (optional)callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `urls` | string | An array of URLs. The first URL is a primary tab and subsequent URLs are subtabs. Note that the last URL is the subtab on which the console is focused. These URLs can be standard Salesforce URLs or relative URLs. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <A HREF="#" onClick="testGenerateConsoleURL();return false">
        Click here to generate a console URL</A>

    <script type="text/javascript">
        function showConsoleUrl(result) {
            alert(result.consoleUrl);
         }
        function testGenerateConsoleURL() {
            sforce.console.generateConsoleUrl([/apex/pagename, /entityId,
www.externalUrl.com, Standard Salesforce Url/entityId], showConsoleUrl);        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `consoleUrl` | string | Console URL that represents the array of URLs passed into Salesforce. |
| `success` | boolean | `true` if the URL was generated successfully, `false` if otherwise. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## getEnclosingPrimaryTabId()

Returns the ID of the current primary tab. This method works within a primary tab or subtab, not within the navigation tab or custom console components. This method is only available in API version 20.0 or later.

## Syntax

```
sforce.console.getEnclosingPrimaryTabId((optional)callback:function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testCloseTab();return false">
         Click here to close this primary tab</A>

    <apex:includeScript value="/support/console/20.0/integration.js"/>
    <script type="text/javascript">
        function testCloseTab() {
            //First find the ID of the current primary tab to close it
            sforce.console.getEnclosingPrimaryTabId(closeSubtab);
        }

        var closeSubtab = function closeSubtab(result) {
            //Now that we have the primary tab ID, we can close it
            var tabId = result.id;
            sforce.console.closeTab(tabId);
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the current primary tab that contains this tab. |

## getEnclosingPrimaryTabObjectId()

Returns the object ID of the current primary tab, which contains a subtab. For example, a case ID or account ID. This method works within a primary tab or subtab. This method is only available in API version 24.0 or later.

### Syntax

```
sforce.console.getEnclosingPrimaryTabObjectId((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testGetEnclosingPrimaryTabObjectId();">
           Click here to get enclosing primary tab object ID</A>

    <apex:includeScript value="/support/console/24.0/integration.js"/>
    <script type="text/javascript">
        function testGetEnclosingPrimaryTabObjectId() {
            sforce.console.getEnclosingPrimaryTabObjectId(showObjectId);
        }
            var showObjectId = function showObjectId(result) {
                // Display the object ID
            alert ('Object ID: ' + result.id);
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the current primary tab that contains this subtab. |
| `success` | boolean | `true` if returning the enclosing primary tab was successful; `false` if returning the enclosing primary tab wasn't successful. |

## getEnclosingTabId()

Returns the ID of the tab that contains the current Visualforce page, which may be a primary tab or subtab. This method will work if the call is made within a component enclosed within a subtab. This method is only available in API version 20.0 or later.

### Syntax

```
sforce.console.getEnclosingTabId()
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testCloseTab();return false">
        Click here to close this tab</A>

    <apex:includeScript value="/support/console/20.0/integration.js"/>
    <script type="text/javascript">
        function testCloseTab() {
            //First find the ID of the current tab to close it
            sforce.console.getEnclosingTabId(closeSubtab);
        }

        var closeSubtab = function closeSubtab(result) {
            //Now that we have the tab ID, we can close it
            var tabId = result.id;
            sforce.console.closeTab(tabId);
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the current primary tab or subtab. |

## getFocusedPrimaryTabId()

Returns the ID of the primary tab on which the browser is focused. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.getFocusedPrimaryTabId((optional) callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>

     <A HREF="#" onClick="testGetFocusedPrimaryTabId();return false">
         Click here to get the focused primary tab ID</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testGetFocusedPrimaryTabId() {
            sforce.console.getFocusedPrimaryTabId(showTabId);
        }
        var showTabId = function showTabId(result) {
            //Display the tab ID
            alert('Tab ID: ' + result.id);
        };

  </script>

</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the primary tab on which the browser is focused. If no primary tab is open, the ID is null. |
| `success` | boolean | `true` if returning the primary tab ID on which the browser is focused was successful; `false` if returning the primary tab ID on which the browser is focused wasn't successful. |

## `getFocusedPrimaryTabObjectId()`

Returns the object ID of the primary tab on which the browser is focused. This method is only available in API version 25.0 or later.

## Syntax

```
sforce.console.getFocusedPrimaryTabObjectId((optional) callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>

     <A HREF="#" onClick="testGetFocusedPrimaryTabObjectId();return false">
         Click here to get the focused primary tab object ID</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testGetFocusedPrimaryTabObjectId() {
            sforce.console.getFocusedPrimaryTabObjectId(showObjectId);
        }
        var showObjectId = function showObjectId(result) {
            //Display the object ID
            alert('Object ID: ' + result.id);
        };

  </script>

</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `id` | string | The object ID of the primary tab on which the browser is focused. If there is no primary tab open, the ID is null. |
| `success` | boolean | `true` if returning the primary tab object ID on which the browser is focused was successful; `false` if returning the primary tab object ID on which the browser is focused wasn't successful. |

## `getFocusedSubtabId()`

Returns the ID of the subtab on which the browser is focused. For example, a case ID or account ID. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.getFocusedSubtabId((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that’s called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testGetFocusedSubtabId();">
           Click here to get the ID of the focused subtab</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testGetFocusedSubtabId() {
            sforce.console.getFocusedSubtabId(showTabId);
        }
        var showTabId = function showTabId(result) {
                // Display the tab ID
            alert ('Tab ID: ' + result.id);
        };
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the subtab on which the browser is focused. If no subtab is open, the ID is null. |
| `success` | boolean | `true` if returning the ID of the focused subtab was successful; `false` if returning the ID of the focused subtab wasn't successful. |

## getFocusedSubtabObjectId()

Returns the object ID of the subtab on which the browser is focused. For example, a case ID or account ID. This method is only available in API version 24.0 or later.

### Syntax

```
sforce.console.getFocusedSubtabObjectId((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testGetFocusedSubtabObjectId();">
           Click here to get the object ID of the focused subtab</A>

    <apex:includeScript value="/support/console/24.0/integration.js"/>
    <script type="text/javascript">
        function testGetFocusedSubtabObjectId() {
            sforce.console.getFocusedSubtabObjectId(showObjectId);
        }
            var showObjectId = function showObjectId(result) {
                // Display the object ID
            alert ('Object ID: ' + result.id);
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The object ID of the subtab on which the browser is focused. If no subtab is open, the ID is null. |
| `success` | boolean | `true` if returning the object ID of the focused subtab was successful; `false` if returning the object ID of the focused subtab wasn't successful. |

## getPageInfo()

Returns page information for the specified tab after its content has loaded. If the tab ID is null, it returns page information for the enclosing primary tab or subtab. Note that to get the page information from a custom console component, a `tabId` must be passed as the first parameter to this method.This method is only available in API version 26.0 or later.

### Syntax

```
sforce.console.getPageInfo(tabId:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `tabId` | string | ID of the tab from which page information is returned. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testGetPageInfo();return false">
        Click here to get page information</A>

    <apex:includeScript value="/support/console/26.0/integration.js"/>
    <script type="text/javascript">
        function testGetPageInfo() {
            //Get the page information of 'scc-pt-1'
           //This value is for example purposes only
             var tabId = 'scc-pt-1';
              sforce.console.getPageInfo(tabId , showPageInfo);
        }

        var showPageInfo = function showPageInfo(result) {
            alert('Page Info: ' + result.pageInfo);
            };
  </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `pageInfo` | string | Returns the URL of the current page as a JSON string, and includes any applicable object ID, object name, object type, and for API version 33.0 or later, the object tab name. For example:<br>`{"url":"http://na.salesforce.com/001x0000003DGQR",`<br>`"objectId":"001x0000003DGQR","objectName":"Acme","object":"Account","displayName":"Company"`<br>For API version 31.0 and later, invoking this API method on a PersonAccount object returns the following additional information.<br>• accountId or contactId, the associated account or contact ID<br>• personAccount, which is `true` if the object is a PersonAccount and `false` otherwise<br>For example:<br>`{"url":"http://na1.salesforce.com/001x0000003DGQR",`<br>`"objectId":"001x0000003DGQR","objectName":"Acme Person Account",`<br>`"object":"Account", "contactId":"003D000000QOMqg",`<br>`"personAccount":true}` |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## getPrimaryTabIds()

Returns all of the IDs of open primary tabs. This method is only available in API version 26.0 or later.

### Syntax

```
sforce.console.getPrimaryTabIds((optional) callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testGetPrimaryTabIds();return false">
        Click here to get the primary tab IDs</A>

    <apex:includeScript value="/support/console/26.0/integration.js"/>
    <script type="text/javascript">
        function testGetPrimaryTabIds() {
              sforce.console.getPrimaryTabIds(showTabId);
        }
        var showTabId = function showTabId(result) {
            //Display the primary tab IDs
            alert('Primary Tab IDs: ' + result.ids);
            };
  </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `ids` | string | An array of open primary tab IDs , in order of appearance. |
| `success` | boolean | `true` if returning the IDs of open primary tabs was successful; `false` if returning the IDs of open primary tabs wasn't successful. |

## getSubtabIds()

Returns all of the IDs of the subtabs on the primary tab specified by a primary tab ID. If the primary tab ID is null, it returns the IDs of the subtabs on the current primary tab. This method can only be called from a custom console component or a detail page overwritten by a Visualforce page. This method is only available in API version 26.0 or later.

### Syntax

```
sforce.console.getSubtabIds( (optional) primaryTabId:String, (optional) callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `primaryTabId` | string | ID of the primary tab from which the subtab IDs are returned. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testGetSubtabIds();return false">
        Click here to get the subtab IDs</A>

    <apex:includeScript value="/support/console/26.0/integration.js"/>
    <script type="text/javascript">
        function testGetSubtabIds() {
            //Get the subtabs of the primary tab 'scc-pt-0'
           //This value is for example purposes only
             var primaryTabId = 'scc-pt-0';
              sforce.console.getSubtabIds(primaryTabId , showTabId);
        }

        var showTabId = function showTabId(result) {
            //Display the subtab IDs
            alert('Subtab IDs: ' + result.ids);
            };
  </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `ids` | string | An array of open subtab IDs. |
| `success` | boolean | `true` if firing the event was successful; `false` if firing the event wasn't successful. |

## `getTabLink()`

Retrieves the URL to a tab, or group of related tabs, from the Salesforce console. This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console.getTabLink(level:String, (optional)tabId:String,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `level` | string | Level that matches one of the Link to Share options in the Salesforce console user interface. The options are:<br>• All primary tabs and subtabs — `sforce.console.TabLink.PARENT_AND_CHILDREN`.<br>• Only the specified tab — `sforce.console.TabLink.TAB_ONLY`<br>• A standard Salesforce URL — `sforce.console.TabLink.SALESFORCE_URL`<br>For more information, see “Salesforce Console Tabs” in the online help. |

| Name | Type | Description |
|---|---|---|
| `tabId` | string | Optional tab ID of the tab from which you're retrieving the URL. If you do not pass a tab ID, the URL to the current tab is returned. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <A HREF="#" onClick="getEnclosingPrimaryTabId();return false">
        Click here to get tab link</A>

    <script type="text/javascript">
        var getEnclosingPrimaryTabId = function getEnclosingPrimaryTabId() {
            sforce.console.getEnclosingPrimaryTabId(getTabLink);
        }
        var getTabLink = function getTabLink(result) {
          sforce.console.getTabLink(sforce.console.TabLink.PARENT_AND_CHILDREN, result.id,
 showTabLink);
        }
        var showTabLink = function showTabLink(result) {
            var link = result.tabLink;
            };
  </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `tabLink` | string | The retrieved URL. |
| `success` | boolean | `true` if the link was retrieved successfully, `false` if retrieving was unsuccessful. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## `isInConsole()`

Determines if the page is in the Salesforce console. This method is only available in API version 22.0 or later.

## Syntax

```
sforce.console.isInConsole()
```

## Arguments

None

## Sample Code–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testIsInConsole();return false">
         Click here to check if the page is in the Service Cloud console</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testIsInConsole() {
            if (sforce.console.isInConsole()) {
                  alert('in console');
               } else {
                  alert('not in console');
            }
        }
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

## Response

Returns `true` if the page is in the Salesforce console; `false` if the page is not in the Salesforce console.

## onEnclosingTabRefresh()

Registers a function to call when the enclosing tab refreshes. This method is only available in API version 24.0 or later.

## Syntax

```
sforce.console.onEnclosingTabRefresh(eventHandler:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when the enclosing tab refreshes. |

### Sample Code–Visualforce

```
<apex:page>

    <apex:includeScript value="/support/console/24.0/integration.js"/>

    <script type="text/javascript">
        var eventHandler = function eventHandler(result) {
            alert('Enclosing tab has refreshed:' + result.id
          + 'and the object Id is:' + result.objectId);
        };
            sforce.console.onEnclosingTabRefresh(eventHandler);
    </script>
</apex:page>
```

### Event Handler Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the refreshed tab. |
| `objectId` | string | The object ID of the refreshed tab or null if no object exists. |

## onFocusedSubtab()

Registers a function to call when the focus of the browser changes to a different subtab. This method is only available in API version 24.0 or later.

### Syntax

```
sforce.console.onFocusedSubtab(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when the focus of the browser changes to a different subtab. |

### Sample Code–Visualforce

```
<apex:page>
```

```
    <apex:includeScript value="/support/console/24.0/integration.js"/>

    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('Focus changed to a different subtab. The subtab Id is:'
        + result.id + 'and the object Id is:' + result.objectId);
        };
        sforce.console.onFocusedSubtab(eventHandler);
    </script>
</apex:page>
```

## Event Handler Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the subtab on which the browser is focused. |
| `objectId` | string | The object ID of the subtab on which the browser is focused or null if no object exists. |

## `onTabSave()`

Registers and calls a callback method when a user clicks **Save** in a subtab's Unsaved Changes dialog box. When using this method, you must call `setTabUnsavedChanges()` in the callback method. This notifies the console that the custom save operation completed. In the call to `setTabUnsavedChanges()`, pass the first parameter as `false` to indicate a successful save or `true` to indicate an unsuccessful save. This method is only available in API version 28.0 or later.

Registering a callback method affects the user interface. When no save handler is registered, the user is presented with two options when closing a tab with unsaved changes: **Continue** or **Cancel**. When a save handler is registered, the user is presented with three options when closing the tab: **Save**, **Don't Save**, or **Cancel**. In this scenario, the callback method registered is called when the user chooses **Save**.

Important: When using `onTabSave()` with `setTabUnsavedChanges()`:

- Note that calling `sforce.console.setTabUnsavedChanges(false, ...)` closes the specified tab. We recommend placing the call to `sforce.console.setTabUnsavedChanges()` at the end of the callback method, as any subsequent save logic might not execute.
- *Not* calling `sforce.console.setTabUnsavedChanges()` will have a severe affect on the user interface. For example, closing a primary tab with a subtab for which `sforce.console.setTabUnsavedChanges()` has not been called prevents a `Saving...` modal dialog box from closing.
- Any callback passed to `sforce.console.setTabUnsavedChanges()` will not execute if the specified tab saves successfully and closes.

Note: Calling `onTabSave()` from a custom console component prevents that component from refreshing when saving the tab. For more information on custom console components, see "Console Components" in the Salesforce online help.

### Syntax

```
sforce.console.onTabSave(callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called to handle the save operation. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testOnTabSave();return false">
         Click here to register save handler</A>

    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <script type="text/javascript">
        function testOnTabSave() {
            sforce.console.onTabSave(handleSave);
        }
    var handleSave = function handleSave(result) {
        alert('save handler called from tab with id ' + result.id +
             ' and objectId ' + result.objectId);
        //Perform save logic here

        //Mark tab as 'clean'
        sforce.console.setTabUnsavedChanges(false, undefined, result.id);
    };
  </script>
</apex:page>
```

### Response

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the tab being saved. |
| `objectId` | string | Object ID of the tab being saved, if applicable; null otherwise. |

## `openConsoleUrl()`

Opens a URL created by the `generateConsoleUrl()` method (a URL to a tab, or group of related tabs, in the Salesforce console). This method is only available in API version 28.0 or later.

## Syntax

```
sforce.console.openConsoleUrl(id:String, consoleUrl:URL, active:Boolean,
(optional)tabLabels:String, (optional)tabNames:String, (optional)callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the console tab to override. If the ID corresponds to an existing primary tab, then the existing primary tab is redirected to the given URL because the console prevents duplicate tabs. Use null to create a new primary tab. |
| `consoleUrl` | string | Console URL that represents the array of URLs passed into Salesforce. |
| `active` | boolean | If `true`, the opened primary tab displays immediately. If `false`, the opened primary tab displays in the background and the current tab maintains focus. |
| `tabLabels` | string | Optional array of labels of the opened primary tab or subtabs. The order in which the tabs appear in the console URL should match the order of the labels that appear in the array. If you do not want to set the labels of tabs, use an empty string (`''`). |
| `tabNames` | string | Optional array of names of the opened primary and subtabs. The order in which the tabs appear in the console URL should match the order of the names that appear in the array. If you do not want to set the names of tabs, use an empty string (`''`). |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <A HREF="#" onClick="testGenerateConsoleURL();return false">
        Click here to open a console URL</A>

    <script type="text/javascript">
        var generateConsoleUrl = function testGenerateConsoleURL() {
            sforce.console.generateConsoleUrl([/apex/pagename, /entityId,
www.externalUrl.com, Standard Salesforce Url/entityId], showConsoleUrl);
         }
        var openConsoleUrl = function showConsoleUrl(result) {
            sforce.console.openConsoleUrl(null, result.consoleUrl, true, ['Apex', '',
'Salesforce', ''], ['', '', 'externalUrl', ''])
         }
  </script>
</apex:page>
```

Note: This example shows that if you want to set a label or name, you must set the other values to empty string (`''`).

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the console URL was opened successfully, `false` otherwise. |

## `openPrimaryTab()`

Opens a new primary tab to display the content of the specified URL, which can be relative or absolute. You can also override an existing tab. This method is only available in API version 20.0 or later.

- If the ID corresponds to an existing primary tab, the existing tab is redirected to the given URL because the Salesforce console prevents duplicate tabs.
- If the URL is to a Salesforce object, that object displays as specified in the Salesforce console app settings. For example, if cases are set to open as a subtab of their parent accounts, and `openPrimaryTab()` is called on a case, the case opens as subtab on its parent account's primary tab.

If there's an error opening the tab, the error code is reported in the JavaScript console.

### Syntax

```
sforce.console.openPrimaryTab(id:String, url:URL, active:Boolean,
(optional)tabLabel:String, (optional)callback:Function, (optional)name)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the primary tab to override.<br>Use `null` to create a new primary tab.<br>If the ID corresponds to an existing primary tab, the existing tab is redirected to the given URL because the Salesforce console prevents duplicate tabs. |
| `url` | URL | URL of the opened primary tab.<br>If the URL is to a Salesforce object, that object displays as specified in the Salesforce console app settings. For example, if cases are set to open as a subtab of their parent accounts, and `openPrimaryTab()` is called on a case, the case opens as subtab on its parent account's primary tab.<br>Note that users can open an external URL if it has been added to the console's whitelist. For more information, see "Whitelist Domains for a Salesforce Console" in the online help. |

| Name | Type | Description |
|---|---|---|
| `active` | boolean | If `true`, the opened primary tab displays immediately. If `false`, the opened primary tab displays in the background and the current tab maintains focus. |
| `tabLabel` | string | Optional label of the opened primary tab. If a label isn't specified, `External Page` displays.<br>Add labels as text; HTML isn't supported. |
| `callback` | function | JavaScript method called upon completion of the method. |
| `name` | string | Optional name of the opened primary tab.<br>This argument is only available in API version 22.0 and later. |

## Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testOpenPrimaryTab();return false">
         Click here to open a new primary tab</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testOpenPrimaryTab() {
            //Open a new primary tab with the salesforce.com home page in it
            sforce.console.openPrimaryTab(null, 'http://www.salesforce.com', false,
                'salesforce', openSuccess, 'salesforceTab');
        }

        var openSuccess = function openSuccess(result) {
            //Report whether opening the new tab was successful
            if (result.success == true) {
                alert('Primary tab successfully opened');
            } else {
                alert('Primary tab cannot be opened');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the tab successfully opened; `false` if the tab didn't open. |
| `id` | string | ID of the primary tab. IDs are only valid during a user session; IDs become invalid when a user leaves the Salesforce console. |

## `openSubtab()`

Opens a new subtab (within a primary tab) that displays the content of a specified URL, which can be relative or absolute. You can also override an existing subtab. Use to open a new subtab on a primary tab via the primary tab's ID. This method is only available in API version 20.0 or later.

If there's an error opening the tab, the error code is reported in the JavaScript console.

### Syntax

```
sforce.console.openSubtab(primaryTabId:String, url:URL, active:Boolean, tabLabel:String,
 id:String, (optional)callback:Function, (optional)name:String)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `primaryTabId` | string | ID of the primary tab in which the subtab opened. |
| `url` | URL | URL of the opened subtab.<br>If the URL is to a Salesforce object, that object displays as specified in the Salesforce console app settings. For example, if cases are set to open as a primary tab, and `openSubtab()` is called on a case, the case opens as a primary tab.<br>Note that users can open an external URL if it has been added to the console's whitelist. For more information, see "Whitelist Domains for a Salesforce Console" in the online help. |
| `active` | boolean | If `true`, the opened subtab displays immediately. If `false`, the opened subtab displays in the background and the current tab maintains focus. |
| `tabLabel` | string | Optional label of the opened subtab. If a label isn't specified, `External Page` displays.<br>Add labels as text; HTML isn't supported. |
| `id` | string | ID of the subtab to override.<br>Use `null` to create a new subtab. |
| `callback` | function | JavaScript method called upon completion of the method. |

| Name | Type | Description |
|---|---|---|
| `name` | string | Optional name of the opened subtab.<br>This argument is only available in API version 22.0 and later. |

## Sample Code–Visualforce

```
<apex:page standardController="Case">

    <A HREF="#" onClick="testOpenSubtab();return false">
        Click here to open a new subtab</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testOpenSubtab() {
            //First find the ID of the primary tab to put the new subtab in
            sforce.console.getEnclosingPrimaryTabId(openSubtab);
        }

        var openSubtab = function openSubtab(result) {
            //Now that we have the primary tab ID, we can open a new subtab in it
            var primaryTabId = result.id;
            sforce.console.openSubtab(primaryTabId , 'http://www.salesforce.com', false,

                'salesforce', null, openSuccess, 'salesforceSubtab');
        };

        var openSuccess = function openSuccess(result) {
            //Report whether we succeeded in opening the subtab
            if (result.success == true) {
                alert('subtab successfully opened');
            } else {
                alert('subtab cannot be opened');
            }
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the subtab successfully opened; `false` if the subtab didn't open. |

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the subtab. IDs are only valid during a user session; IDs become invalid when the user leaves the Salesforce console. |

## openSubtabByPrimaryTabName()

Opens a new subtab (within a primary tab) that displays the content of a specified URL, which can be relative or absolute. You can also override an existing subtab. Use to open a new subtab on a primary tab via the primary tab's name. This method is only available in API version 22.0 or later.

If there's an error opening the tab, the error code is reported in the JavaScript console.

### Syntax

```
sforce.console.openSubtabByPrimaryTabName(primaryTabName:String, url:URL, active:Boolean,
 tabLabel:String, id:String, (optional)callback:Function, (optional)name:String)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `primaryTabName` | string | Name of the primary tab in which the subtab opened. |
| `url` | URL | URL of the opened subtab.<br>If the URL is to a Salesforce object, that object displays as specified in the Salesforce console app settings. For example, if cases are set to open as a primary tab, and `openSubtab()` is called on a case, the case opens as a primary tab.<br>Note that users can open an external URL if it has been added to the console's whitelist. For more information, see "Whitelist Domains for a Salesforce Console" in the online help. |
| `active` | boolean | If `true`, the opened subtab displays immediately. If `false`, the opened subtab displays in the background and the current tab maintains focus. |
| `tabLabel` | string | Optional label of the opened subtab. If a label isn't specified, `External Page` displays.<br>Add labels as text; HTML isn't supported. |
| `id` | string | ID of the subtab to override.<br>Use `null` to create a new subtab. |
| `callback` | function | JavaScript method called upon completion of the method. |
| `name` | string | Optional name of the opened subtab.<br>This argument is only available in API version 22.0 and later. |

## Sample Code–Visualforce

```
<apex:page standardController="Case">

    <A HREF="#" onClick="testOpenSubtab();return false">
        Click here to open a new subtab by primary tab name</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testOpenSubtabByPrimaryTabName() {
            //First open a primary tab by name
            sforce.console.openPrimaryTab(null, 'http://www.yahoo.com', true, 'Yahoo',
openSubtab, 'yahoo');
        }

        var openSubtab = function openSubtab(result) {
            //Open the subtab by the name specified in function
testOpenSubtabByPrimaryTabName()
          sforce.console.openSubtabByPrimaryTabName('yahoo', 'http://www.salesforce.com',
 true,
                'salesforce', null, openSuccess);
        };

        var openSuccess = function openSuccess(result) {
            //Report whether we succeeded in opening the subtab
            if (result.success == true) {
                alert('subtab successfully opened');
            } else {
                alert('subtab cannot be opened');
            }
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the subtab successfully opened; `false` if the subtab didn't open. |
| `id` | string | ID of the subtab. IDs are only valid during a user session; IDs become invalid when the user leaves the Salesforce console. |

## `refreshPrimaryTabById()`

Refreshes a primary tab specified by ID, including its subtabs. This method can't refresh subtabs with URLs to external pages or Visualforce pages. This method is only available in API version 22.0 or later.

### Syntax

```
sforce.console.refreshPrimaryTabById(id:String, active:Boolean,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the primary tab to refresh. |
| `active` | boolean | If `true`, the refreshed primary tab displays immediately. If `false`, the refreshed primary tab displays in the background. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testRefreshPrimaryTabById();return false">
         Click here to refresh a primary tab by id</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testRefreshPrimaryTabById() {
            //Get the value for 'scc-pt-0' from the openPrimaryTab method
            //This value is for example purposes only
            var primaryTabId = 'scc-pt-0';
            sforce.console.refreshPrimaryTabById(primaryTabId, true, refreshSuccess);
        }

        var refreshSuccess = function refreshSuccess(result) {
            //Report whether refreshing the primary tab was successful
            if (result.success == true) {
                alert('Primary tab refreshed successfully');
            } else {
                alert('Primary did not refresh');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the primary tab refreshed successfully; `false` if the primary tab didn't refresh. |

## `refreshPrimaryTabByName()`

Refreshes a primary tab specified by name, including its subtabs. This method can't refresh subtabs with URLs to external pages or Visualforce pages. This method is only available in API version 22.0 or later.

### Syntax

```
sforce.console.refreshPrimaryTabByName(name:String, active:Boolean,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `name` | string | Name of the primary tab to refresh. |
| `active` | boolean | If `true`, the refreshed primary tab displays immediately. If `false`, the refreshed primary tab displays in the background. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testRefreshPrimaryTabByName();return false">
         Click here to refresh a primary tab by name</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testRefreshPrimaryTabByName() {
            //Set the name of the tab by using the openPrimaryTab method
            //This value is for example purposes only
            var primaryTabName = 'myPrimaryTab';
            sforce.console.refreshPrimaryTabByName(primaryTabName, true, refreshSuccess);

        }

        var refreshSuccess = function refreshSuccess(result) {
            //Report whether refreshing the primary tab was successful
            if (result.success == true) {
                alert('Primary tab refreshed successfully');
            } else {
                alert('Primary tab did not refresh');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the primary tab refreshed successfully; `false` if the primary tab didn't refresh. |

## `refreshSubtabById()`

Refreshes a subtab with the last known URL with a specified ID. This method can't refresh a subtab If the last known URL is an external page or a Visualforce page. This method is only available in API version 22.0 or later.

### Syntax

```
sforce.console.refreshSubtabById(id:String, active:Boolean, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `id` | string | ID of the subtab to refresh. |
| `active` | boolean | If `true`, the refreshed subtab displays immediately. If `false`, the refreshed subtab displays in the background. |

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testRefreshSubtabById();return false">
         Click here to refresh a subtab by id</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testRefreshSubtabById() {
            //Set the name of the tab by using the openSubtab method
            //This value is for example purposes only
            var subtabId = 'scc-st-0';
            sforce.console.refreshSubtabById(subtabId, true, refreshSuccess);
        }

        var refreshSuccess = function refreshSuccess(result) {
            //Report whether refreshing the subtab was successful
            if (result.success == true) {
                alert('Subtab refreshed successfully');
            } else {
                alert('Subtab did not refresh');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the subtab refreshed successfully; `false` if the subtab didn't refresh. |

## `refreshSubtabByNameAndPrimaryTabId()`

Refreshes a subtab with the last known URL with the specified name and primary tab ID. This method can't refresh a subtab If the last known URL is an external page or a Visualforce page. This method is only available in API version 22.0 or later.

## Syntax

```
sforce.console.refreshSubtabByNameAndPrimaryTabId(name:String, primaryTabId:String,
active:Boolean, (optional)callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `name` | string | Name of the subtab to refresh. |
| `primaryTabId` | string | ID of the primary tab in which the subtab opened. |
| `active` | boolean | If `true`, the refreshed subtab displays immediately. If `false`, the refreshed subtab displays in the background. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testRefreshSubtabByNameAndPrimaryTabId();return false">
         Click here to refresh a subtab by name and primary tab ID</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testRefreshSubtabByNameAndPrimaryTabId() {
            //Get the value for 'mySubtab' and 'scc-pt-0' from the openSubtab method
            //These values are for example purposes only
            var subtabName = 'mySubtab';
            var primaryTabId = 'scc-pt-0';
            sforce.console.refreshSubtabByNameAndPrimaryTabId(subtabName, primaryTabId,
true, refreshSuccess);
        }

        var refreshSuccess = function refreshSuccess(result) {
            //Report whether refreshing the subtab was successful
            if (result.success == true) {
                alert('Subtab refreshed successfully');
            } else {
                alert('Subtab did not refresh');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the subtab refreshed successfully; `false` if the subtab didn't refresh. |

## `refreshSubtabByNameAndPrimaryTabName()`

Refreshes a subtab with the last known URL with the specified name and primary tab name. This method can't refresh a subtab If the last known URL is an external page or a Visualforce page. This method is only available in API version 22.0 or later.

### Syntax

```
sforce.console.refreshSubtabByNameAndPrimaryTabName(name:String, primaryTabName:String,
 active:Boolean, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `name` | string | Name of the subtab to refresh. |
| `primaryTabName` | string | Name of the primary tab in which the subtab opened. |
| `active` | boolean | If `true`, the refreshed subtab displays immediately. If `false`, the refreshed subtab displays in the background. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testRefreshSubtabByNameAndPrimaryTabName();return false">
         Click here to refresh a subtab by name and primary tab name</A>

    <apex:includeScript value="/support/console/22.0/integration.js"/>
    <script type="text/javascript">
        function testRefreshSubtabByNameAndPrimaryTabName() {
            //Get the value for 'mySubtab' and 'myPrimaryTab' from the openSubtab method
            //These values are for example purposes only
            var subtabName = 'mySubtab';
            var primaryTabName = 'myPrimaryTab';
          sforce.console.refreshSubtabByNameAndPrimaryTabName(subtabName, primaryTabName,
 true, refreshSuccess);
```

```
        }

        var refreshSuccess = function refreshSuccess(result) {
            //Report whether refreshing the subtab was successful
            if (result.success == true) {
                alert('Subtab successfully refreshed');
            } else {
                alert('Subtab did not refresh');
            }
        };

  </script>

</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the subtab refreshed successfully; `false` if the subtab didn't refresh. |

## `reopenLastClosedTab()`

Reopens the last closed primary tab, and any of its subtabs that were open, the moment it was closed. This method is only available in API version 35.0 or later.

### Syntax

```
sforce.console.reopenLastClosedTab()
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/35.0/integration.js"/>
    <script type="text/javascript">

        var  = callback = function (result) {
            if (result.success) {
                alert('Last tab was re-opened!');
            } else {
                alert('No tab was re-opened.');
            }
        };

        function reopenLastClosedTabTest() {
               sforce.console.reopenLastClosedTab(callback);
        }

  </script>
  <A HREF="#" onClick="reopenLastClosedTabTest(); return false">Re-open Last Closed Tab</A>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the tab was reopened, `false` otherwise. |

## `resetSessionTimeOut()`

Resets a session timeout on a Visualforce page so that users can continue working without being logged out. This method is only available in API version 24.0 or later.

For more information, see "Modify Session Security Settings" in the Salesforce Help.

### Syntax

```
sforce.console.resetSessionTimeOut()
```

### Arguments

None

### Sample Code–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testResetSessionTimeOut();">
           Click here to reset session timeout</A>
    <apex:includeScript value="/support/console/24.0/integration.js"/>
    <script type="text/javascript">
        function testResetSessionTimeOut() {
            sforce.console.resetSessionTimeOut();
        };
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see "Define Custom Buttons and Links" in the Salesforce online help.

## Response

None

## setTabUnsavedChanges()

Sets the unsaved changes icon ( * ) on subtabs to indicate unsaved data. This method is only available in API version 23.0 or later.

### Syntax

```
sforce.console.setTabUnsavedChanges(unsaved:Boolean, callback:Function,
(optional)subtabId:String)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `unsaved` | boolean | If `true`, the tab is marked as having unsaved changes. |
| `callback` | function | JavaScript method that's called upon completion of the method. |
| `subtabId` | string | The ID of the subtab that is marked as having unsaved changes.<br>This argument is only available in API version 25.0 or later. |

### Sample Code API Version 23.0 or Later–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testSetTabUnsavedChanges();return false">
          Click here to indicate this tab has unsaved changes</A>

    <apex:includeScript value="/support/console/23.0/integration.js"/>
    <script type="text/javascript">
        function testSetTabUnsavedChanges() {
            sforce.console.setTabUnsavedChanges(true, displayResult);
        };
        function displayResult(result) {
             if (result.success) {
                 alert('Tab status has been successfully updated');
             } else {
                 alert('Tab status couldn’t be updated');
             }
        }

    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

## Response

This method returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if update was successful; `false` if update wasn't successful. |

## Sample Code API Version 25.0 or Later–Visualforce

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testSetTabUnsavedChanges();return false">
           Click here to indicate this tab has unsaved changes</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
         function testSetTabUnsavedChanges() {
            sforce.console.getFocusedSubtabId(setTabDirty);
        };
         function setTabDirty(result) {
            sforce.console.setTabUnsavedChanges(true, displayResult, result.id);
        };
         function displayResult(result) {
            if (result.success) {
                alert('Tab status has been successfully updated');
             } else {
                alert('Tab status couldn’t be updated');
             }
        };
    </script>
</apex:page>
```

Note: This example is only set to run if the Visualforce page is inside an application-level custom component. For more information, see Methods for Application-Level Custom Console Components on page 84.

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if returning the focused subtab ID was successful; `false` if if returning the focused subtab ID wasn't successful. |

## `setTabIcon()`

Sets an icon on the specified tab. If a tab is not specified, the icon is set on the enclosing tab. Use this method to customize a tab’s icon. This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console.setTabIcon(iconUrl:String, tabID:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `iconUrl` | string | A URL pointing to an image, which is used as the tab’s icon. If null or undefined, the tab’s default icon is used. |
| `tabId` | string | The ID of the tab on which to set the icon. If null or undefined, the enclosing tab’s ID is used. |
| `callback` | function | JavaScript method that’s called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testSetTabIcon();return false">
         Click here to change the enclosing tab’s icon</A> <BR/>
    <A HREF="#" onClick="testResetTabIcon(); return false;">
         Click here to reset the enclosing tab’s icon</A>

    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <script type="text/javascript">
        function checkResult(result) {
            if (result.success) {
               alert('Tab icon set successfully!');
            } else {
               alert('Tab icon cannot be set!');
            }
    }
```

```
        function testSetTabIcon() {
            sforce.console.setTabIcon('http://host/path/to/your/icon.png', null,
checkResult);
    }
        function testResetTabIcon() {
            sforce.console.setTabIcon(null, null, checkResult);
    }
    </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the tab's icon was successful, `false` if setting the tab's icon wasn't successful. |

Note: If this method is called without passing in a tab ID, the tab in which the Visualforce page is enclosed is used. If there isn't an enclosing tab, the error message `Cannot get a workspace or view tab from the given ID` displays in the browser's developer console.

## setTabLink()

Sets a console tab's URL attribute to the location of the tab's content. Use this method to generate secure console URLs when users navigate to tabs displaying content outside of the Salesforce domain.This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console.setTabLink((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page standardController="Account">
    <apex: detail />
        <apex:includeScript value="/support/console/28.0/integration.js"/>
    <script type="text/javascript">
```

```
        window.onload = function() {
            sforce.console.setTabLink();
             };
  </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the link was set successfully, `false` if setting was unsuccessful. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## setTabStyle()

Sets a cascading style sheet (CSS) on the specified tab. If a tab is not specified, the CSS is set on the enclosing tab. Use this method to customize a tab's look and feel. This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console.setTabStyle(style:String, tabId:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `style` | string | A CSS specification string used to style the tab. If null or undefined, the tab's default style is used. |
| `tabId` | string | The ID of the tab on which to set the style. If null or undefined, the enclosing tab's ID is used. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testSetTabStyle();return false">
         Click here to change the enclosing tab’s background color to red</A> <BR/>
    <A HREF="#" onClick="testResetTabStyle(); return false;">
         Click here to reset the enclosing tab’s style</A>

    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <script type="text/javascript">
        function checkResult(result) {
            if (result.success) {
               alert('Tab style set successfully!');
            } else {
               alert('Tab style cannot be set!');
            }
    }
        function testSetTabStyle() {
            sforce.console.setTabStyle('background:red;', null, checkResult);
    }
        function testResetTabStyle() {
            sforce.console.setTabStyle(null, null, checkResult);
    }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the tab's style was successful, `false` if setting the tab's style wasn't successful. |

Note: If this method is called without passing in a tab ID, the tab in which the Visualforce page is enclosed is used. If there isn't an enclosing tab, the error message `Cannot get a workspace or view tab from the given ID` displays in the browser's developer console.

## `setTabTextStyle()`

Sets a cascading style sheet (CSS) on a specified tab's text. If a tab is not specified, the CSS is set on the enclosing tab's text. Use this method to customize a tab's text style. This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console. setTabTextStyle(style:String, tabID:String, (optional)callback:Function))
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `style` | string | A CSS specification string used to set a tab's text style. If null or undefined, the tab's default text style is used. |

| Name | Type | Description |
|---|---|---|
| `tabId` | string | The ID of the tab on which to set the text style. If null or undefined, the enclosing tab's ID is used. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testSetTabTextStyle();return false">
         Click here to change the enclosing tab’s text style</A> <BR/>
    <A HREF="#" onClick="testResetTabTextStyle(); return false;">
         Click here to reset the enclosing tab’s text style</A>

    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <script type="text/javascript">
        function checkResult(result) {
            if (result.success) {
               alert('Tab text style set successfully!');
            } else {
               alert('Tab text style cannot be set!');
            }
    }
        function testSetTabTextStyle() {
            sforce.console.setTabTextStyle('color:blue;font-style:italic;', null,
checkResult);
    }
        function testResetTabTextStyle() {
            sforce.console.setTabTextStyle(null, null, checkResult);
    }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the tab's text style was successful, `false` if setting the tab's text style wasn't successful. |

Note: If this method is called without passing in a tab ID, the tab in which the Visualforce page is enclosed is used. If there isn't an enclosing tab, the error message `Cannot get a workspace or view tab from the given ID` displays in the browser's developer console.

## setTabTitle()

Sets the title of a primary tab or subtab. This method is only available in API version 20.0 or later.

### Syntax

```
sforce.console.setTabTitle(tabTitle:String, (optional)tabID:String)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `tabTitle` | string | Title of a primary tab or subtab. |
| `tabId` | string | The ID of the tab in which to set the title.<br>This argument is only available in API version 25.0 or later. |

### Sample Code–Visualforce API Version 20.0 or Later

```
<apex:page standardController="Case">
    <A HREF="#" onClick="testSetTabTitle();return false">
         Click here to change this tab's title</A>

    <apex:includeScript value="/support/console/20.0/integration.js"/>
    <script type="text/javascript">
        function testSetTabTitle() {
            //Set the current tab's title
            sforce.console.setTabTitle('My New Title');
        }
    </script>
</apex:page>
```

Note: This example is set to run by clicking a custom link on a case. For more information, see “Define Custom Buttons and Links” in the Salesforce online help.

### Response

None

### Sample Code–Visualforce API Version 25.0 or Later

```
<apex:page>
    <A HREF="#" onClick="testSetTabTitle();return false">
         Click here to change the title of the focused primary tab</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        sforce.console.getFocusedPrimaryTabId(function(result){
               sforce.console.setTabTitle('My New Title', result.id);
        });
    </script>
</apex:page>
```

Note: This example is only set to run if the Visualforce page is inside an application-level custom component. For more information, see Methods for Application-Level Custom Console Components on page 84.

## Response

None

# CHAPTER 4 Methods for Navigation Tabs

A Salesforce console displays a navigation tab from which users can select objects to view lists or home pages. Administrators choose the objects that users can access from a navigation tab. For more information, see "Salesforce Console Navigation Tab" and "Create a Salesforce Console App" in the online help.

IN THIS SECTION:

## focusNavigationTab()

Focuses the browser on the navigation tab. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.focusNavigationTab((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        var callback = function (result) {}
            if(result.success){
```

```
            alert('success');
        }
        else{
        alert('Something is wrong.');
        }
    };
        sforce.console.focusNavigationTab(callback);
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if returning the object IDs was successful; `false` otherwise. |

## `getNavigationTabs()`

Returns all of the objects in the navigation tab. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.getNavigationTabs((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        var callback = function (result) {
           var id;
              if (result.success) {
                var tempItem = JSON.parse(result.items);
                for (var i = 0, len = tempItem.length; i < len; i++) {

alert('Label:'+tempItem[i].label+'listViewURl:'+tempItem[i].listViewUrl+'navTabid:'
           +tempItem[i].navigationTabId+'Selected ' +tempItem[i].selected);
           }
              } else {
                alert('something is wrong!');
                }
        };
        sforce.console.getNavigationTabs(callback);
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `menuItems` | object | The IDs of objects in the navigation tab. |
| `success` | boolean | `true` if returning the IDs of objects in the navigation tab was successful, `false` otherwise. |

## getSelectedNavigationTab()

Returns the selected object in the navigation tab. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.getSelectedNavigationTab((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        var callback = function (result) {}
            if (result.success) {
             alert('the navigation tab id is ' + result.navigationTabId + ' and navigation
 url is ' + result.listViewUrl);
```

```
                        } else {
                alert('something is wrong!');
            }
        };
        sforce.console.getSelectedNavigationTab(callback);
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `navigationTabId` | string | The object ID of the selected object. |
| `listViewUrl` | object | The list view URL of the selected object. |
| `label` | object | The label of the selected object. |
| `selected` | boolean | `true` if returning the selected field of the object was successful, `false` otherwise. |
| `success` | boolean | `true` if returning the object IDs was successful, `false` otherwise. |

## `refreshNavigationTab()`

Refreshes the selected navigation tab. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.refreshNavigationTab((optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        var callback = function (result) {}
            if(result.success){
              alert('success');
            }
            else{
              alert('Something is wrong.');
            }
        };
        sforce.console.refreshNavigationTab(callback);
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if refreshing the navigation tab was successful, `false` otherwise. |

## `setSelectedNavigationTab()`

Sets the navigation tab with a specific ID or URL. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.setSelectedNavigationTab((optional)callback, navigatorTabId:(optional)string,
 url:(optional)string)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |
| `navigatorTabId` | string | The ID of the navigation tab to be selected. |
| `url` | string | The URL of the navigation tab to be selected. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">
        var callback = function (result) {}
            if (result.success) {
              alert('Successful');
            } else {
                alert('something is wrong!');
            }
        };
        sforce.console.setSelectedNavigationTab(callback,'nav-tab-4');
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the navigation tab with a specific ID or URL was successful, `false` otherwise. |

# CHAPTER 5 Methods for Computer-Telephony Integration (CTI)

Salesforce Call Center seamlessly integrates Salesforce with Computer-Telephony Integration systems. Whether developers create a CTI system with Open CTI or the CTI Toolkit, console users access telephony features through a SoftPhone, which is a call-control tool that appears in the footer of a console. For more information, see “Salesforce Open CTI Overview” and “Call Center Overview” in the Salesforce Help.

IN THIS SECTION:

setCallAttachedData()
Sets the call data associated with a call object ID. Use to get information or send information between an interaction log and a custom console component.This method is only available in API version 31.0 or later.

setCallObjectIds()
Sets call object IDs, in ascending order of arrival. This method is only available in API version 31.0 or later.

## `fireOnCallBegin()`

Fires an event that notifies a call has begun. Use to get information or send information between an interaction log and a custom console component. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.cti.fireOnCallBegin( callObjectId:String, callType:String, callLabel:String,
 (optional)callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callObjectId` | string | The object ID of the call. |
| `callType` | string | String that specifies the call type, which must be `internal`, `inbound` or `outbound`. |
| `callLabel` | string | String that specifies a call as it appears in the **Attach Call** drop-down button. For example, `Call Label — Inbound Call 12:52:31 PM`. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testFireOnCallBegin();return false">
             Click here to start a call</A>

    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        function testFireOnCallBegin() {
            sforce.console.cti.fireOnCallBegin('call.794937' , 'outbound' , 'label 1');
        }

    </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if firing the event is successful, `false` otherwise. |

## fireOnCallEnd()

Fires an event that notifies a call has ended. Use to get information or send information between an interaction log and a custom console component. This method executes when `fireOnCallBegin()` is called first. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.cti.fireOnCallEnd( callObjectId:String, callDuration:Number,
callDisposition:String, (optional)callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callObjectId` | string | The object ID of the call. |
| `callDuration` | number | Number specifying the duration of the call. |
| `callDisposition` | string | String representing the call's disposition, such as call successful, left voicemail, or disconnected. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testFireOnCallEnd();return false">
             Click here to end a call</A>

    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        function testFireOnCallEnd() {
            //Here, 'call.1' refers to a call that is in progress.
            sforce.console.cti.fireOnCallEnd('call.1', 60, 'Set Appointment');
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if firing the event is successful, `false` otherwise. |

## fireOnCallLogSaved()

Calls the `eventHandler` function registered with `onCallLogSaved()`. Use to get information or send information between an interaction log and a custom console component.. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.cti.fireOnCallLogSaved( id:String, (optional)callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `id` | string | The object ID of the saved call log. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        var MyCallback = function (result) {
             alert('fireOnCallLogSaved was thrown: ' + result.success);
        };

        function testFireOnCallLogSaved() {
            // Simulates that a call log was saved by passing the task object Id as input.

             sforce.console.cti.fireOnCallLogSaved('00Txx000003qf8u', myCallback);
        }

        var callback = function (result) {
            alert('Call Log was saved! Object Id saved is : ' + result.id);
        };

        sforce.console.cti.onCallLogSaved(callback);
    </script>
    <a href="#" onClick="testFireOnCallLogSaved();return false">
        Test fireOnCallLogSaved API!</a>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if firing the event is successful, `false` otherwise. |

## getCallAttachedData()

Returns the attached data of a call represented by the call object ID or null if there isn't an active call. The data is returned in JSON format. This method is for computer-telephony integration (CTI); it's only available in API version 24.0 or later.

### Syntax

```
sforce.console.cti.getCallAttachedData( callObjectId, getCallType, (optional)
callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callObjectId` | string | The call object ID of the call that retrieves the attached data. |
| `getCallType` | boolean | `true` if the type of call is returned as either 'INTERNAL,' 'INBOUND,' or 'OUTBOUND'; `false` otherwise. This field is only available in API version 29.0 or later. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
     <script type="text/javascript">

     /* Note: Open CTI needs to set call type to before getting it, and call type is
INTERNAL/INBOUND/OUTBOUND.
```

```
      */

           var callback2 = function (result) {
              alert('Call attached data  is ' + result.data + '\n Call Type is ' +
result.type);

           };

           /* Retrieving call ID of first call that came in and
            * calling getCallAttachedData() to retrieve call data.
            */
           var callback1 = function (result) {
                if (result.ids && result.ids.length > 0) {
                  sforce.console.cti.getCallAttachedData(result.ids[0], callback2,
{getCallType:true});
                }
           };

          //Note that we are using the CTI submodule here
          function testGetCallAttachedData() {
            sforce.console.cti.getCallObjectIds(callback1);
        };

     </script>
  <h1>CTI</h1>
   <button onclick="testGetCallAttachedData()">getAttachedData</button>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `data` | string | The attached data of a call in JSON format. |
| `success` | boolean | `true` if returning the attached data was successful; `false` if returning the attached data wasn't successful. |
| `type` | string | The type of call. Possible values are 'INTERNAL', 'INBOUND', and 'OUTBOUND'. |

## `getCallObjectIds()`

Returns any active call object IDs in the order in which they arrived or null if there aren't any active calls. This method is for computer-telephony integration (CTI); it's only available in API version 24.0 or later.

## Syntax

```
sforce.console.cti.getCallObjectIds( (optional) callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/24.0/integration.js"/>
     <script type="text/javascript">

          var callback = function (result) {
              alert('Active call object ids: ' + result.ids);
          };

          //Note that we are using the CTI submodule here
          sforce.console.cti.getCallObjectIds(callback);
     </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `ids` | string | The call object IDs of active calls or null if no call is active. |
| `success` | boolean | `true` if returning the active call object IDs was successful; `false` if returning the active call object IDs wasn't successful. |

## onCallBegin()

Registers a function that is called when a call begins (comes in). This method is for computer-telephony integration (CTI); it's only available in API version 24.0 or later.

### Syntax

```
sforce.console.cti.onCallBegin( eventHandler:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when a call begins. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/24.0/integration.js"/>
     <script type="text/javascript">

           var callback = function (result) {
              alert('Call ' + result.id + 'Just came in!');
           };

          //Note that we are using the CTI submodule here
           sforce.console.cti.onCallBegin(callback);
     </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `id` | string | The call object ID of the call which has begun. |

## `onCallEnd()`

Registers a function that is called when a call ends. This method is for computer-telephony integration (CTI); it's only available in API version 24.0 or later.

### Syntax

```
sforce.console.cti.onCallEnd( eventHandler:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when a call ends. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/24.0/integration.js"/>
     <script type="text/javascript">

           var callback = function (result) {
              var str = 'Call ' + result.id + ' ended! ';
              str += 'Call duration is ' + result.duration + '. ';
              str += 'Call disposition is ' + result.disposition;
              alert(str);
           };

          //Note that we are using the CTI submodule here
           sforce.console.cti.onCallEnd(callback);
     </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The call object ID of the call which has ended. |
| `duration` | string | The duration of the call. |
| `disposition` | string | The disposition of the call. |

# onCallLogSaved()

Registers a function that is fired when an interaction log saves a call log. Use to get information or send information between an interaction log and a custom console component. This method is only available in API version 31.0 or later.

## Syntax

```
sforce.console.cti.onCallLogSaved( eventHandler:Function )
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | For a standard interaction log, `eventHandler` is a function that is executed when a call log is saved. For a custom interaction log, `eventHandler` is a function that is executed when the `fireOnCallLogSaved` API is called in your Visualforce page. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <script type="text/javascript">

        var callback = function (result) {
            alert('Call Log was saved! Object Id saved is : ' + result.id);
        };

        sforce.console.cti.onCallLogSaved(callback);

  </script>
    <p>Registered onCallLogSaved listener...</p>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `id` | string | Call log object ID that was saved. |

## onSendCTIMessage()

Registers a function that is fired when a message is sent with the `sendCTIMessage()`. Use to get information or send information between an interaction log and a custom console component. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.cti.onSendCTIMessage( eventHandler:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when a message is sent with the `sendCTIMessage()` method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
     <script type="text/javascript">
```

```
            var callback = function (result) {
                alert('sendCTIMessage API sent the following message: ' + result.message);

            };

            sforce.console.cti.onSendCTIMessage(callback);

            function sendCTIMessage() {
                sforce.console.cti.sendCTIMessage('sending a message to CTI');
            }
      </script>
      <a href="#" onClick="sendCTIMessage();return false">
                  Send a message to see your listener receiving it!</a>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `message` | string | The message that was sent with the `sendCTIMessage()` method. |

## `sendCTIMessage()`

Sends a message to the CTI adapter or Open CTI. This method is for computer-telephony integration (CTI); it's only available in API version 24.0 or later.

### Syntax

```
sforce.console.cti.sendCTIMessage( msg, (optional) callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `msg` | string | Message to send to the adapter. |
| `callback` | function | JavaScript method called when the message is sent. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/24.0/integration.js"/>
     <script type="text/javascript">

           var callback = function (result) {
              if (result.success) {
                 alert('CTI message was sent successfully!');
             } else {
                 alert('CTI message was not sent successfully.');
               }
           };

          //Note that we are using the CTI submodule here
           sforce.console.cti.sendCTIMessage('/ANSWER?LINE_NUMBER=1', callback);
     </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if sending the message was successful; `false` if sending the message wasn't successful. |

## setCallAttachedData()

Sets the call data associated with a call object ID. Use to get information or send information between an interaction log and a custom console component.This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.cti.setCallAttachedData( callObjectId:String, callData:JSON string
callType:String, (optional)callback:Functional)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callObjectId` | string | The object ID of the call. |
| `callData` | string | JSON string that specifies the data to attach to the call. |
| `callType` | string | String that specifies the call type, such as `internal`, `inbound`, or `outbound`. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testSetCallAttachedData();return false">
          Click here to set call attached data </A>

    <apex:includeScript value="/support/console/31.0/integration.js"/>
     <script type="text/javascript">

           function testSetCallAttachedData() {
              //callData must be a JSON string. We assume that your browser has
              //access to a JSON library.
              var callData = JSON.stringify({"ANI":"4155551212", "DNIS":"8005551212"});

              //Set the call attached data associated to call id 'call.1'
              sforce.console.cti.setCallAttachedData('call.1', callData, 'outbound');
           }
     </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the event firing was successful; `false` otherwise. |

## `setCallObjectIds()`

Sets call object IDs, in ascending order of arrival. This method is only available in API version 31.0 or later.

### Syntax

```
sforce.console.cti.setCallObjectIds( callObjectIds:Array, callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callObjectId` | array | An array of string IDs specifying the calls to set. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>
    <A HREF="#" onClick="testSetCallObjectIds();return false">
             Click here to set call object Ids</A>

     <apex:includeScript value="/support/console/31.0/integration.js"/>
     <script type="text/javascript">

          function checkResult(result) {
              if (result.success) {
                    alert('Call object ids set successfully!');
              } else {
                    alert('Call object ids cannot be set!');
              }
          }

          function testSetCallObjectIds() {
              sforce.console.cti.setCallObjectIds(['call.1', 'call.2', 'call.3'],
checkResult);
          }
     </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the call IDs was successful; `false` otherwise. |

# CHAPTER 6 Methods for Application-Level Custom Console Components

Custom console components let you customize, extend, or integrate the footer, sidebars, highlights panels, and interaction logs of a Salesforce console using Visualforce, canvas apps, lookup fields, or related lists. Administrators can add components to either:

- Page layouts to display content on specific pages
- Salesforce console apps to display content across all pages and tabs

For more information, see "Console Components" in the Salesforce Help.

IN THIS SECTION:

setCustomConsoleComponentButtonIconUrl()
Sets the button icon URL of an application-level custom console component that's on a page. This method is only available in API version 25.0 or later.

setCustomConsoleComponentButtonStyle()
Sets the style of a button used to launch an application-level custom console component that's on a page. This method is only available in API version 25.0 or later.

setCustomConsoleComponentButtonText()
Sets the text on a button used to launch an application-level custom console component that's on a page. This method is only available in API version 25.0 or later.

setCustomConsoleComponentHeight()
Sets the window height of an application-level custom console component that's on a page. This method is available in API version 32.0 or later.

setCustomConsoleComponentVisible()
Sets the window visibility of an application-level custom console component that's on a page. This method is available in API version 32.0 and later. In API version 31.0 and earlier, this method was called `setCustomConsoleComponentWindowVisible()`.

setCustomConsoleComponentWidth()
Sets the window width of an application-level custom console component that's on a page. This method is available in API version 32.0 or later.

setCustomConsoleComponentPopoutable()
Sets a custom console component to be popped out or popped into a browser. To use this method, multi-monitor components must be turned on. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the online help. This method is only available in API version 30.0 or later.

setCustomConsoleComponentWindowVisible()
Sets the window visibility of an application-level custom console component that's on a page. This method is available in API versions 25.0 through 31.0.

setSidebarVisible()
Shows or hides a console sidebar based on `tabId` and region. This method is available in API version 33.0 or later.

## addToBrowserTitleQueue()

Adds a browser tab title to a list of titles, which rotates every three seconds. This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console.addToBrowserTitleQueue( title:String, callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `title` | string | Browser tab title that is displayed. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <A HREF="#" onClick="testAddToBrowserTitleQueue();return false">
            Click here to enqueue a browser title</A>

    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <script type="text/javascript">
                 function testAddToBrowserTitleQueue() {
             var title = 'TestTitle';
          sforce.console.addToBrowserTitleQueue(title);
          }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | If `true`, the title was successfully added to the browser title queue. If `false`, the title wasn't added to the browser title queue. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## blinkCustomConsoleComponentButtonText()

Blinks a button's text on an application-level custom console component that's on a page. This method is only available in API version 25.0 or later.

## Syntax

```
sforce.console.blinkCustomConsoleComponentButtonText(alternateText:String, interval:number,
 (optional)callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `alternateText` | string | The alternate text to display when the button text blinks. |
| `interval` | number | Controls how often the text blinks in milliseconds. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testBlinkCustomConsoleComponentButtonText();return false">
         Click here to blink the button text on a custom console component</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testBlinkCustomConsoleComponentButtonText() {
        //Blink the custom console component button text
            sforce.console.blinkCustomConsoleComponentButtonText('Hello World', 10,
function(result){
              if (result.success) {
               alert('The text blinking starts!');
              } else {
               alert('Could not initiate the text blinking!');
              }
            });
           }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if blinking, the button text was successful; `false` if blinking the button text wasn't successful. |

## `isCustomConsoleComponentPoppedOut()`

Determines if a custom console component is popped out from a browser. To use this method, multi-monitor components must be turned on. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the online help. This method is only available in API version 30.0 or later.

## Syntax

```
sforce.console.isCustomConsoleComponentPoppedOut (callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="checkPoppedOut(); return false;">
         Is this component popped out?</A> <BR/>

    <apex:includeScript value="/support/console/30.0/integration.js"/>
    <script type="text/javascript">
        function checkResult(result) {
          if (result.success) {
            alert('Is this component popped out: ' + result.poppedOut);
          } else {
            alert('Error invoking this method. Check the browser developer console for
more information.');
          }
        }
        function checkPoppedOut() {
          sforce.console.isCustomConsoleComponentPoppedOut(checkResult);
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if returning the component's pop out status was successful; `false` otherwise. |
| `poppedOut` | boolean | `true` if the component is popped out; `false` otherwise. |

## `isCustomConsoleComponentWindowHidden()`

Determines if the application-level custom console component window is hidden. This method is available in API versions 25.0 through 31.0.

Note: If this method is called from a popped out component in a Salesforce console where multi-montior components is turned on, nothing will happen. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the online help. Starting in API version 32.0, this method is no longer available and has been replaced by `isCustomConsoleComponentHidden()`. For more information, see "`isCustomConsoleComponentHidden()`."

## Syntax

```
sforce.console.isCustomConsoleComponentWindowHidden((optional) callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>

     <A HREF="#" onClick="testIsCustomConsoleComponentWindowHidden();return false">
         Click here to check if the custom console component window is hidden</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testIsCustomConsoleComponentWindowHidden() {
            sforce.console.isCustomConsoleComponentWindowHidden(checkWindowVisibility);
        }

        var checkWindowVisibility = function checkWindowVisibility(result) {
            //Display the window visibility
            if (result.success) {
              alert('Is window hidden: ' + result.hidden);
            } else {
              alert('Error!');
            }
        }
  </script>

</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `hidden` | boolean | `true` if the custom console component window is hidden; `false` if the custom console component window is visible. |
| `success` | boolean | `true` if returning the custom console component window visibility was successful; `false` if returning the custom console component window visibility wasn't successful. |

## `isCustomConsoleComponentHidden()`

Determines if the application-level custom console component window is hidden. This method is available in API version 32.0 and later. In API version 31.0 and earlier, this method was called `isCustomConsoleComponentWindowHidden()`.

## Syntax

```
sforce.console.isCustomConsoleComponentHidden((optional) callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>

     <A HREF="#" onClick="testIsCustomConsoleComponentHidden();return false">
         Click here to check if the custom console component window is hidden</A>

    <apex:includeScript value="/support/console/32.0/integration.js"/>
    <script type="text/javascript">
        function testIsCustomConsoleComponentHidden() {
            sforce.console.isCustomConsoleComponentHidden(checkWindowVisibility);
        }

        var checkWindowVisibility = function checkWindowVisibility(result) {
            //Display the window visibility
            if (result.success) {
              alert('Is window hidden: ' + result.hidden);
            } else {
              alert('Error!');
            }
        }
  </script>

</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `hidden` | boolean | `true` if the custom console component window is hidden; `false` if the custom console component window is visible. |
| `success` | boolean | `true` if the `isCustomConsoleComponentHidden()` call was successful; `false` if the `isCustomConsoleComponentHidden()` call wasn't successful. |

## isInCustomConsoleComponent()

Determines if the page is in an application-level custom console component. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.isInCustomConsoleComponent((optional) callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

     <A HREF="#" onClick="testIsInCustomConsoleComponent();return false">
         Click here to check if the page is in an app-level custom console component</A>


    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testIsInCustomConsoleComponent() {
            sforce.console.isInCustomConsoleComponent(checkInComponent);
        }

        var checkInComponent = function checkInComponent(result) {
            //Check if in component
            alert('Is in custom console component: ' + result.inCustomConsoleComponent);
        };

  </script>

</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `inCustomConsoleComponent` | boolean | `true` if the page is in a custom console component; `false` if the page isn't in a custom console component. |

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if returning the page status was successful; `false` if returning the page status wasn't successful. |

## `onCustomConsoleComponentButtonClicked()`

Registers a function to call when a button is clicked on an application-level custom console component. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.onCustomConsoleComponentButtonClicked(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called when a button is clicked on a custom console component. |

### Sample Code–Visualforce

```
<apex:page>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
             var eventHandler = function (result) {
                 alert('The Custom Console Component button is clicked. The component ID
is: ' + result.id +
                 ' and the component window is: ' + (result.windowHidden ? 'hidden' :
'visible'));
        };

        sforce.console.onCustomConsoleComponentButtonClicked(eventHandler);
    </script>

</apex:page>
```

### Event Handler Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the custom console component which includes the page. |

| Name | Type | Description |
|---|---|---|
| `windowHidden` | boolean | `true` if the custom console component window is hidden after the button is clicked; `false` if the custom console component window is visible after the button is clicked. |

## `onFocusedPrimaryTab()`

Registers a function to call when the focus of the browser changes to a different primary tab. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.onFocusedPrimaryTab(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when the focus of the browser changes to a different primary tab. |

### Sample Code–Visualforce

```
<apex:page>

    <apex:includeScript value="/support/console/25.0/integration.js"/>

    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('Focus changed to a different primary tab. The primary tab ID is:'
        + result.id + 'and the object Id is:' + result.objectId);
        };
        sforce.console.onFocusedPrimaryTab(eventHandler);
    </script>
</apex:page>
```

### Event Handler Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `id` | string | The ID of the primary tab on which the browser is focused. |

| Name | Type | Description |
|---|---|---|
| `objectId` | string | The object ID of the primary tab on which the browser is focused or null if no object exists. |

## removeFromBrowserTitleQueue()

Removes a browser tab title from the list of titles, which rotates every three seconds. This method is only available in API version 28.0 or later.

### Syntax

```
sforce.console.removeFromBrowserTitleQueue( title:String, callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `title` | string | Browser tab title to remove. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/28.0/integration.js"/>
    <script type="text/javascript">

        <A HREF="#" onClick="testAddToBrowserTitleQueue();return false"> {
            Click here to enqueue a browser title</A>

        <A HREF="#" onClick="testRemoveFromBrowserTitleQueue();return false">
            Click here to remove browser title</A>

        var title = 'TestTitle';
        function testAddToBrowserTitleQueue() {
            sforce.console.addToBrowserTitleQueue(title);
        }
        function testRemoveFromBrowserTitleQueue() {
            sforce.console.removeFromBrowserTitleQueue(title);
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | If `true`, the title was successfully removed from the browser title queue. If `false`, the title wasn't removed from the browser title queue. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## scrollCustomConsoleComponentButtonText()

Scrolls a button's text on an application-level custom console component that's on a page. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.scrollCustomConsoleComponentButtonText(interval:number, pixelsToScroll:number,
 isLeftScrolling:boolean,  (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `interval` | number | Controls how often the button text is scrolled in milliseconds. |
| `pixelsToScroll` | number | Controls how many pixels the button text scrolls. |
| `isLeftScrolling` | boolean | If `true`, the text scrolls left. If `false`, the text scrolls right. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testScrollCustomConsoleComponentButtonText();return false">
         Click here to scroll the button text on a custom console component</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testScrollCustomConsoleComponentButtonText() {
        //Scroll the custom console component button text from right to left
            sforce.console.scrollCustomConsoleComponentButtonText(500, 10, true,
function(result){
              if (result.success) {
                    alert('The text scrolling starts!');
                  } else {
                    alert('Could not initiate the text scrolling!');
                  }
              });
            }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if scrolling the button text was successful; `false` if scrolling the button text wasn't successful. |

## **`setCustomConsoleComponentButtonIconUrl()`**

Sets the button icon URL of an application-level custom console component that's on a page. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.setCustomConsoleComponentButtonIconUrl(iconURL:String,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `iconUrl` | string | A URL that points to an image that's used as a button to launch a custom console component. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testSetCustomConsoleComponentButtonIconUrl();return false">
         Click here to set the custom console component button icon</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testSetCustomConsoleComponentButtonIconUrl() {

sforce.console.setCustomConsoleComponentButtonIconUrl('http://imageserver/img.png');
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the button icon URL was successful; `false` if setting the button icon URL wasn't successful. |

## setCustomConsoleComponentButtonStyle()

Sets the style of a button used to launch an application-level custom console component that's on a page. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.setCustomConsoleComponentButtonStyle(style:String, (optional)callback:
Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `style` | string | The style of a button used to launch a custom console component. The styles supported include font, font color, and background color. Font and font color isn't available for Internet Explorer® 7. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testSetCustomConsoleComponentButtonStyle();return false">
         Click here to set the style of a button used to launch a custom console
component</A>
```

```
    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testSetCustomConsoleComponentButtonStyle() {
            sforce.console.setCustomConsoleComponentButtonStyle('background:red;');
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the button style was successful; `false` if setting the button style wasn't successful. |

## setCustomConsoleComponentButtonText()

Sets the text on a button used to launch an application-level custom console component that’s on a page. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.setCustomConsoleComponentButtonText(text:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `text` | string | Text that’s displayed on a button used to launch a custom console component. |
| `callback` | function | JavaScript method that’s called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testSetCustomConsoleComponentButtonText();return false">
       Click here to set the text on a button used to launch a custom console component</A>


    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testSetCustomConsoleComponentButtonText() {
```

```
        //Change the custom console component button text to 'Hello World'
            sforce.console.setCustomConsoleComponentButtonText('Hello World');
        }
    </script>
</apex:page>
```

## Response

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the button text was successful; `false` if setting the button text wasn't successful. |

## `setCustomConsoleComponentHeight()`

Sets the window height of an application-level custom console component that's on a page. This method is available in API version 32.0 or later.

Note: If this method is called from a popped out component in a Salesforce console where multi-monitor components is turned on, nothing will happen. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the Salesforce Help.

### Syntax

```
sforce.console.setCustomConsoleComponentHeight( height:number, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `height` | number | The new height in pixels. |
| `callback` | function | Javascript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testSetCustomConsoleComponentHeight();return false">
         Click here to set the custom console component height to 100px</A>

    <apex:includeScript value="/support/console/32.0/integration.js"/>
    <script type="text/javascript">
        function testSetCustomConsoleComponentHeight() {
        // Set the custom console component height
            sforce.console.setCustomConsoleComponentHeight(100);
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the method call was successful; `false` otherwise. |

## `setCustomConsoleComponentVisible()`

Sets the window visibility of an application-level custom console component that's on a page. This method is available in API version 32.0 and later. In API version 31.0 and earlier, this method was called `setCustomConsoleComponentWindowVisible()`.

### Syntax

```
sforce.console.setCustomConsoleComponentVisible(visible:Boolean,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `visible` | boolean | `true` to make the custom console component window visible, `false` to hide the custom console component window. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testSetCustomConsoleComponentVisible();return false">
         Click here to make the custom console component window visible</A>

    <apex:includeScript value="/support/console/32.0/integration.js"/>
    <script type="text/javascript">
        function testSetCustomConsoleComponentVisible() {
        // Make the custom console component window visible
            sforce.console.setCustomConsoleComponentVisible(true);
        }
    </script>
</apex:page>
```

### Response

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the button window visibility was successful; `false` if setting the button window visibility wasn't successful. |

## `setCustomConsoleComponentWidth()`

Sets the window width of an application-level custom console component that's on a page. This method is available in API version 32.0 or later.

Note: If this method is called from a popped out component in a Salesforce console where multi-monitor components is turned on, nothing will happen. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the Salesforce Help.

### Syntax

```
sforce.console.setCustomConsoleComponentWidth( width:number, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `width` | number | The new width in pixels. |
| `callback` | function | Javascript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testSetCustomConsoleComponentWidth();return false">
         Click here to set the custom console component width to 100px</A>

    <apex:includeScript value="/support/console/32.0/integration.js"/>
    <script type="text/javascript">
        function testSetCustomConsoleComponentWidth() {
        // Set the custom console component width
            sforce.console.setCustomConsoleComponentWidth(100);
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if the method call was successful; `false` otherwise. |

## `setCustomConsoleComponentPopoutable()`

Sets a custom console component to be popped out or popped into a browser. To use this method, multi-monitor components must be turned on. For more information, see “Turn on Multi-Monitor Components for a Salesforce Console” in the online help. This method is only available in API version 30.0 or later.

### Syntax

```
sforce.console.setCustomConsoleComponentPopoutable(popoutable:Boolean,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `popoutable` | boolean | If `true`, the component can be popped out or popped into a browser. If `false`, the component cannot be popped out or popped into a browser. |
| `callback` | function | JavaScript method that’s called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="enablePopout(); return false;">
         Click here to enable pop out or pop in functionality</A> <BR/>
    <A HREF="#" onClick="disablePopout(); return false;">
         Click here to disable pop out or pop in functionality</A>

    <apex:includeScript value="/support/console/30.0/integration.js"/>
    <script type="text/javascript">
        function checkResult(result) {
          if (result.success) {
            alert('The method was successfully invoked.');
          } else {
            alert('Error while invoking this method. Check the browser developer console
for more information.');
          }
        }

        function enablePopout() {
          sforce.console.setCustomConsoleComponentPopoutable(true, checkResult);
        }

        function disablePopout() {
          sforce.console.setCustomConsoleComponentPopoutable(false, checkResult);
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if enabling pop out or pop in functionality for the component was successful; `false` otherwise. |

## `setCustomConsoleComponentWindowVisible()`

Sets the window visibility of an application-level custom console component that's on a page. This method is available in API versions 25.0 through 31.0.

Note: If this method is called from a popped out component in a Salesforce console where multi-montior components is turned on, nothing will happen. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the Salesforce Help. Starting in API version 32.0, this method is no longer available and has been replaced by `setCustomConsoleComponentVisible()`. For more information, see `setCustomConsoleComponentVisible()`.

### Syntax

```
sforce.console.setCustomConsoleComponentWindowVisible(visible:Boolean,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `visible` | boolean | `true` to make the custom console component window visible, `false` to hide the custom console component window. |
| `callback` | function | JavaScript method that's called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>

    <A HREF="#" onClick="testSetCustomConsoleComponentWindowVisible();return false">
         Click here to make the custom console component window visible</A>

    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">
        function testSetCustomConsoleComponentWindowVisible() {
        //Make the custom console component window visible
            sforce.console.setCustomConsoleComponentWindowVisible(true);
        }
    </script>
</apex:page>
```

## Response

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if setting the button window visibility was successful; `false` if setting the button window visibility wasn't successful. |

## `setSidebarVisible()`

Shows or hides a console sidebar based on `tabId` and region. This method is available in API version 33.0 or later.

### Syntax

```
sforce.console.setSidebarVisible( visible:Boolean, (optional)tabId:String,
(optional)region:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `visible` | boolean | `true` to show the sidebar or `false` to hide the sidebar. |
| `tabId` | string | The ID of the tab on which to show or hide the sidebar. |
| `region` | string | The region on the console where the sidebar is located, such as left or right, top or bottom. Regions are represented as: • `sforce.console.Region.LEFT` • `sforce.console.Region.RIGHT` • `sforce.console.Region.TOP` • `sforce.console.Region.BOTTOM` |

| Name | Type | Description |
|---|---|---|
| callback | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page>
   <apex:includeScript value="/support/console/33.0/integration.js"/>
   <script type="text/javascript">

      var callback = function (result) {
         if (result.success) {
      alert('Congratulations!');
         }else {
      alert('something is wrong!');
         }
      };
         function setSidebarVisible() {

sforce.console.setSidebarVisible(true,'scc-st-1',sforce.console.Region.LEFT,callback);
      }

   </script>
    <A HREF="#" onClick="setSidebarVisible(); return false">SetSidebarToExpand</A>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| success | boolean | true if the method call was successful; false otherwise. |

# CHAPTER 7 Methods for Push Notifications

Push notifications are visual indicators on lists and detail pages in a console that show when a record or field has changed during a user's session. For example, if two support agents are working on the same case, and one agent changes the `Priority`, a push notification appears to the other agent so he or she spots the change and doesn't duplicate the effort.

When administrators set up a Salesforce console, they choose when push notifications display, and which objects and fields trigger push notifications. Developers can use push notification methods to customize push notifications beyond the default visual indicators supplied by Salesforce. For example, developers can use the methods below to create personalized notifications about objects accessible to specific console users, thereby eliminating the need for email notifications. For more information, see "Configure Push Notifications for a Salesforce Console" in the Salesforce Help.

Consider the following when using push notification methods:

- Push notification listener response is only available for the objects and fields selected to trigger push notifications for a console.
- When a Visualforce page includes a listener added by the `addPushNotificationListener()` method, the page receives notifications. The listener receives notifications when there is an update by any user to the objects selected for triggering console push notifications and the current user has access to the modified record. This functionality is slightly different from push notifications set up in the Salesforce user interface in that:
  - Listeners receive update notifications for changes made by all users.
  - Listeners receive notifications when an object's fields are updated or created, even if those fields aren't selected to trigger push notifications; and the notifications don't include details about what changed. For example, if `Status` on the Case object is set to trigger a push notification, but `Priority` on the Case object changes, a listener receives a notification that the case changed without specifying details.
  - Listeners don't obey the `Choose How Lists Refresh` and `Choose How Detail Pages Refresh` push notifications settings in a Salesforce console.
  - The only way to stop receiving notifications is to remove listeners using the `removePushNotificationListener()` method.

IN THIS SECTION:

## `addPushNotificationListener()`

Adds a listener for a push notification. A user can only register a listener once until he or she removes the listener, or the listener is removed by another user. This method is only available in API version 26.0 or later.

For more information on push notifications, see Methods for Push Notifications on page 106.

## Syntax

```
sforce.console.addPushNotificationListener( objects: array, eventHandler:Function )
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `objects` | array | Objects set to receive notifications. |
| `eventHandler` | function | JavaScript method called when there is a push notification. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/26.0/integration.js"/>
    <script type="text/javascript">

        var eventHandler = function (result) {
            alert('There is a push notification of object: ' + result.Id);
         };
         //Add a push notification listener for Case and Account
         sforce.console.addPushNotificationListener(['Case', 'Account'], eventHandler);
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method.

| Name | Type | Description |
|---|---|---|
| `id` | string | The object ID of the push notification. |
| `entityType` | string | The type of object included in the push notification. For example, Account or Contact.<br><br>Objects available for push notifications are determined by the administrator who set up a Salesforce console. For more information, see "Configure Push Notifications for a Salesforce Console" in the Salesforce online help. |
| `Type` | string | The field of the object included in the push notification. For example, the `Account Name` field on Account. Notifications occur when the field is either updated or created.<br><br>Fields on objects available for push notifications are determined by the administrator who set up a Salesforce console. For more information, see "Configure Push Notifications for a Salesforce Console" in the Salesforce online help. |
| `LastModifiedById` | string | The user ID of the user who last modified the object in the push notification. |

## `removePushNotificationListener()`

Removes a listener that gets added for a push notification. This method is only available in API version 26.0 or later.

For more information on push notifications, see Methods for Push Notifications on page 106.

### Syntax

```
sforce.console.removePushNotificationListener((optional) callback:Function )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | A function called when the removal of the push notification listener completes. |

### Sample Code–Visualforce

```
<apex:page standardController="Case">

     <A HREF="#" onClick="testRemovePushNotification();return false">
         Click here to remove push notification</A>

    <apex:includeScript value="/support/console/26.0/integration.js"/>
    <script type="text/javascript">

        function testRemovePushNotification() {
            sforce.console.removePushNotificationListener(removeSuccess);
         }
        var removeSuccess = function removeSuccess(result) {
            //Report whether removing the push notification listener is successful
            if (result.success == true) {
                alert('Removing push notification was successful');
            } else {
                alert('Removing push notification wasn't successful');
            }
         };
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method.

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if removing the push notification listener was successful; `false` if removing the push notification listener wasn't successful. |

# CHAPTER 8 Methods for Console Events

JavaScript can be executed when certain types of events occur in a console, such as when a user closes a tab. The following standard events are supported:

| Event | Description | Payload |
|---|---|---|
| `sforce.console.ConsoleEvent.OPEN_TAB` | Fired when a primary tab or subtab is opened. Available in API version 30.0 or later. | • `id` — The ID of the opened tab.<br>• `objectId` — The object ID of the opened tab, if available. |
| `sforce.console.ConsoleEvent.CLOSE_TAB` | Fired when a primary tab or subtab with a specified ID in the `additionalParams` argument is closed. Or, fired when a primary tab or subtab with no specified ID is closed. Available in API version 30.0 or later. | • `id` — The ID of the closed tab.<br>• `objectID` — The object ID of the closed tab, if available. |
| `sforce.console.ConsoleEvent.CONSOLE_LOGOUT` | Delays the execution of logging out of a console when a user clicks **Logout**. When **Logout** is clicked:<br>**1.** An overlay appears, which tells a user that logout is in progress.<br>**2.** Callbacks are executed that have been registered by using `sforce.console.ConsoleEvent.CONSOLE_LOGOUT`<br>**3.** Console logout logic is executed.<br>If the callback contains synchronous blocking code, the console logout code isn't executed until the blocking code is executed. As a best practice, avoid synchronous blocking code or long code execution during logout.<br>Available in API version 31.0 or later. | None |

IN THIS SECTION:

## addEventListener()

Adds a listener for a custom event type or a standard event type when the event is fired. This method adds a listener for custom event types in API version 25.0 or later; it adds a listener for standard event types in API version 30.0 or later.

For the list of standard events, see Methods for Console Events on page 109.

### Syntax

```
sforce.console.addEventListener( eventType: String, eventListener:Function,
(optional)additionalParams:Object )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventType` | string | Custom event type for which eventListener listens. |
| `eventListener` | function | JavaScript method called when an eventType is fired. |
| `additionalParams` | object | Optional parameters accepted by this method. The supported properties on this object are `tabId`: The ID of the tab to listen for the specified event.<br>This argument is only available in API version 30.0 or later. |

### Sample Code API Version 25.0 or Later–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">

        var listener = function (result) {
            alert('Message received from event: ' + result.message);
        };
        //Add a listener for the 'SampleEvent' event type
        sforce.console.addEventListener('SampleEvent', listener);
    </script>
</apex:page>
```

### Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `message` | string | The message which is sent with the fired event.<br>If the response is from a custom keyboard shortcut, the `message` includes the following information on which the browser is focused, in this order:<br>**1.** Object ID of the primary tab<br>**2.** ID of the primary tab<br>**3.** Object ID of the subtab<br>**4.** ID of the subtab<br>For more information, see "Customize Keyboard Shortcuts for a Salesforce Console" in the online help. |

## Sample Code API Version 30.0 or Later–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/30.0/integration.js"/>
    <script type="text/javascript">

        var onEnclosingPrimaryTabClose = function (result) {
            alert('The enclosing primary tab is about to be closed. Tab ID: ' + result.id
 + ', Object ID: ' + (result.objectId ? result.objectId : 'not available'));
        };

        //Add a listener to handle the closing of the enclosing primary tab
        sforce.console.getEnclosingPrimaryTabId(function (result) {
            if (result.id) {
                sforce.console.addEventListener(sforce.console.ConsoleEvent.CLOSE_TAB,
                onEnclosingPrimaryTabClose, { tabId : result.id });
            } else {
                alert('Could not find an enclosing primary TAB!');
            }
        });
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `message` | string | The message which is sent with the fired event.<br>If the response is from a console event, the `message` includes payload details as described in Methods for Console Events on page 109.<br>If the response is from a custom keyboard shortcut, the `message` includes the following information on which the browser is focused, in this order: |

| Name | Type | Description |
|---|---|---|
| | | 1. Object ID of the primary tab<br>2. ID of the primary tab<br>3. Object ID of the subtab<br>4. ID of the subtab<br>For more information, see "Customize Keyboard Shortcuts for a Salesforce Console" in the online help. |

## fireEvent()

Fires a custom event. This method is only available in API version 25.0 or later.

### Syntax

```
sforce.console.fireEvent( eventType:String, message:String, (optional)callback:Function
)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventType` | string | The type of custom event to fire. |
| `message` | string | The message which is sent with the fired event. |
| `callback` | function | JavaScript method called when the custom event is fired. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/25.0/integration.js"/>
    <script type="text/javascript">

        <A HREF="#" onClick="testFireEvent(); return false;">
            Click here to fire an event of type 'SampleEvent'</A>

        var callback = function(result) {
            if (result.success) {
                   alert('The custom event is fired!');
            } else {
                   alert('The custom event could not be fired!');
            }
         };

        function testFireEvent() {
```

```
        //Fire an event of type 'SampleEvent'
        sforce.console.fireEvent('SampleEvent', 'EventMessage', callback);
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous, so it returns its response in an object in a callback method. The response object contains the following field:

| Name | Type | Description |
|---|---|---|
| `success` | boolean | `true` if firing the event is successful, `false` if firing the event wasn't successful. |

## **`removeEventListener()`**

Removes a listener for a custom event type or a standard event type. This method removes a listener for custom event types in API version 25.0 or later; it removes a listener for standard event types in API version 30.0 or later.

For the list of standard events, see Methods for Console Events on page 109.

### Syntax

```
sforce.console.removeEventListener( eventType: String, eventListener:Function,
(optional)additionalParams:Object )
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventType` | string | Event type for which eventListener is removed. |
| `eventListener` | function | Event listener to remove. |
| `additionalParams` | object | Optional parameters accepted by this method. The supported properties on this object are `tabId`: The ID of the tab to remove the listener for the specified event. This argument is only available in API version 30.0 or later. |

### Sample Code API Version 25.0 or Later–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/25.0/integration.js"/>
        <A HREF="#" onClick="testRemoveEventListener(); return false;">
            Click here to remove an event listener for the 'SampleEvent' event type</A>
    <script type="text/javascript">
        var listener = function (result) {
            alert('Message received from event: ' + result.message);
        };
        //Add a listener for the 'SampleEvent' event type
        sforce.console.addEventListener('SampleEvent', listener);

        function testRemoveEventListener() {
            sforce.console.removeEventListener('SampleEvent', listener);
        }
    </script>
</apex:page>
```

## Response

None

## Sample Code API Version 30.0 or Later–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/30.0/integration.js"/>
         <A HREF="#" onClick="testRemoveEventListener(); return false;">
              Click here to remove an event listener for the console 'CLOSE_TAB' event
type</A>

    <script type="text/javascript">
        var tabId;

        var onEnclosingPrimaryTabClose = function (result) {
            alert('The enclosing primary tab is about to be closed. Tab ID: ' + result.id
 + ',
                   Object ID: ' + (result.objectId ? result.objectId : 'not available'));

        };

        //Add a listener to handle the closing of the enclosing primary tab
        sforce.console.getEnclosingPrimaryTabId(function (result) {
            if (result.id) {
                tabId = result.id;
                sforce.console.addEventListener(sforce.console.ConsoleEvent.CLOSE_TAB,
    onEnclosingPrimaryTabClose, { tabId : tabId });
            } else {
                alert('Could not find an enclosing primary TAB!');
            }
        });

        function testRemoveEventListener() {
            sforce.console.removeEventListener(sforce.console.ConsoleEvent.CLOSE_TAB,
                               onEnclosingPrimaryTabClose, { tabId : tabId });
        }
    </script>
</apex:page>
```

## Response

None

# CHAPTER 9 Methods for Live Agent

Live Agent lets you connect with customers or website visitors in real time through Web-based chat. For more information, see "Add Live Agent to the Salesforce Console" in the Salesforce Help.

IN THIS SECTION:

onAgentStateChanged()
Registers a function to call when agents change their Live Agent status, such as from Online to Away. Available in API version 29.0 or later.

onChatCanceled()
Registers a function to call when a chat visitor cancels a chat request. Available in API version 29.0 or later.

onChatCriticalWaitState()
Registers a function to call when a chat becomes critical to answer or a waiting chat is answered. Available in API version 29.0 or later.

onChatDeclined()
Registers a function to call when an agent declines a chat request. Available in API version 29.0 or later.

onChatEnded()
Registers a function to call when an engaged chat ends. Available in API version 29.0 or later.

onChatRequested()
Registers a function to call when an agent receives a chat request. Available in API version 29.0 or later.

onChatStarted()
Registers a function to call when an agent starts a new chat with a customer. Available in API version 29.0 or later.

onChatTransferredOut()
Registers a function to call when an engaged chat is transferred out to another agent. Available in API version 29.0 or later.

onCurrentCapacityChanged()
Registers a function to call when an agent's capacity for accepting chats changes—for example, if an agent accepts a new chat, ends a currently engaged chat, or otherwise changes the number of chats to which they are assigned, or if a chat request is pushed to their chat queue. Available in API version 29.0 or later.

onCustomEvent()
Registers a function to call when a custom event takes place during a chat. Available in API version 29.0 or later.

onFileTransferCompleted()
Registers a function to call when a file is transferred from a customer to an agent. Available in API version 31.0 or later.

onNewMessage()
Registers a function to call when a new message is sent from a customer, agent, or supervisor. Available in API version 29.0 or later.

onTypingUpdate()
Registers a function to call when the customer's text in the chat window changes. If Sneak Peek is enabled, this function is called whenever the customer edits the text in the chat window. If Sneak Peek is not enabled, this function is called whenever a customer starts or stops typing in the chat window. Available in API version 29.0 or later.

sendCustomEvent()
Sends a custom event to the client-side chat window for a chat with a specific chat key. Available in API version 29.0 or later.

sendMessage()
Sends a new chat message from the agent to a chat with a specific chat key. Available in API version 29.0 or later.

setAgentInput()
Sets the string of text in the agent's text input area in the chat log of a chat with a specific chat key.Available in API version 29.0 or later.

setAgentState()
Sets an agent's Live Agent status, such as Online, Away, or Offline. Available in API version 29.0 or later.

Methods for Live Agent Chat Visitors

There are a few methods available that you can use to customize the chat visitor experience for Live Agent in a custom Visualforce chat window.

## acceptChat()

Accepts a chat request. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.acceptChat(chatKey:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the chat request you wish to accept. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testAcceptChat();return false;">Accept Chat</a>

    <script type="text/javascript">
        function testAcceptChat() {
            //Get the value for 'myChatKey'from the getChatRequests() or onChatRequested()
 methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            sforce.console.chat.acceptChat(chatKey, acceptSuccess);
        }

        function acceptSuccess(result) {
            //Report whether accepting the chat was succesful
            if (result.success == true) {
                alert('Accepting the chat was successful');
            } else {
                alert('Accepting the chat was not successful');
            }
        };
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if accepting the chat was successful; `false` if accepting the chat wasn't successful. |

## `cancelFileTransferByAgent()`

Indicates that a file transfer request has been canceled by an agent. Available in API version 31.0 or later.

### Syntax

```
sforce.console.chat.cancelFileTransferByAgent(chatKey:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the chat for which the agent canceled the file transfer request. |
| `callback` | function | JavaScript method that is called when the method is completed. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <a href="#" onClick="testCancelFileTransfer();return false;">Cancel file transfer</a>


    <script type="text/javascript">
        function testCancelFileTransfer() {
           //Gets the value for 'myChatKey'from the getChatRequests() or onChatRequested()

              methods.
            //These values are for example purposes only.
            var chatKey = 'myChatKey';
            sforce.console.chat.cancelFileTransferByAgent(chatKey, fileSuccess);
        }

        function fileSuccess(result) {
            //Report whether canceling was successful
            if (result.success == true) {
                alert('Canceling file transfer was successful.');
            } else {
                alert('Canceling file transfer was not successful.');
            }
        };
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if canceling the file transfer request was successful; `false` if canceling the file transfer request wasn't successful. |

## `declineChat()`

Declines a chat request. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.declineChat(chatKey:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the request you wish to decline. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testDeclineChat();return false;">Decline Chat</a>

    <script type="text/javascript">
        function testDeclineChat() {
            //Get the value for 'myChatKey'from the getChatRequests() or onChatRequested()
 methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            sforce.console.chat.declineChat(chatKey, declineSuccess);
        }
        function declineSuccess(result) {
            //Report whether declining the chat was succesful
            if (result.success == true) {
                alert('Declining the chat was successful');
            } else {
                alert('Declining the chat was not successful');
            }
        };
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if declining the event was successful; `false` if declining the event wasn't successful. |

## `endChat()`

Ends a chat in which an agent is currently engaged. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.endChat(chatKey:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the engaged chat you wish to end. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testEndChat();return false;">End Chat</a>

    <script type="text/javascript">
```

```
        function testEndChat() {
            //Get the value for 'myChatKey'from the getEngagedChats() or onChatStarted()
methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            sforce.console.chat.endChat(chatKey, endSuccess);
        }

        function endSuccess(result) {
            //Report whether ending the chat was succesful
            if (result.success == true) {
                alert('Ending the chat was successful');
            } else {
                alert('Ending the chat was not successful');
            }
        };
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if ending the chat was successful; `false` if ending the chat wasn't successful. |

## getAgentInput()

Returns the string of text which is currently in the agent's text input area in the chat log of a chat with a specific chat key. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.getAgentInput(chatKey:String, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which to retrieve the agent's input text. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetAgentInput();">Get Agent Input</a>

    <script type="text/javascript">

        function testGetAgentInput() {
            //Get the value for 'myChatKey'from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            sforce.console.chat.getAgentInput(chatKey, getAgentInputSuccess);
        }

        function getAgentInputSuccess(result) {
            //Report whether getting the agent's input was successful
            if (result.success == true) {
                agentInput = result.text;
                alert('The text in the agent input is: ' + agentInput);
            } else {
                alert('Getting the agent input was not successful');
            }
        };


    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `text` | String | The text that is currently in an agent’s text input area. |
| `success` | Boolean | `true` if getting the agent’s input was successful; `false` if getting the agent’s input wasn’t successful. |

## `getAgentState()`

Returns the agent's current Live Agent status, such as Online, Away, or Offline. Available in API version 29.0 or later.

## Syntax

```
sforce.console.chat.getAgentState(callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetAgentState();return false;">Get Agent State</a>

    <script type="text/javascript">
        function testGetAgentState() {
            sforce.console.chat.getAgentState(function(result) {
                if (result.success) {
                    alert('Agent State:' + result.state);
                } else {
                    alert('getAgentState has failed');
                }
            });
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `state` | String | String representing the current agent state—for example, Online, Away, or Offline. |
| `success` | Boolean | `true` if getting the agent's Live Agent status was successful; `false` if getting the agent's Live Agent status wasn't successful. |

## getChatLog()

Returns the chat log of a chat associated with a specific chat key. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.getChatLog(chatKey:String, callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which to retrieve the chat log. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetChatLog();">Get Chat Log</a>

    <script type="text/javascript">

        function testGetChatLog() {
            //Get the value for 'myChatKey'from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            sforce.console.chat.getChatLog(chatKey, getChatLogSuccess);
        }

        function getChatLogSuccess(result) {
            //Report whether getting the chat log was succesful
            if (result.success == true) {
                chatLogMessage = result.messages[0].content;
                alert('The first message in this chatLog is: ' + chatLogMessage);
            } else {
                alert('Getting the chat log was not successful');
            }
        };


    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following fields:

| Name | Type | Description |
|---|---|---|
| `customEvents` | Array of `customEvent` objects | An array of custom event objects representing the custom events that occurred during a chat. |

| Name | Type | Description |
|---|---|---|
| `messages` | Array of `message` objects | An array of chat message objects containing all of the chat messages from the chat log. |
| `success` | Boolean | `true` if getting the chat log was successful; `false` if getting the chat log wasn't successful. |

### `customEvent`

The `customEvent` object contains a single event from the chat log and the following properties:

| Property | Type | Description |
|---|---|---|
| `source` | String | The person who initiated the custom event, either the chat visitor or the agent. |
| `type` | String | The type of custom event that occurred. |
| `data` | String | The data of the custom event that was sent to the chat; corresponds to the `data` argument of the `liveagent.chasitor.sendCustomEvent()` method used to send this event from the chat window. |
| `timestamp` | Date/Time | The date and time a custom event was received. |

### `message`

The `message` object contains a single chat message from the chat log and the following properties:

| Property | Type | Description |
|---|---|---|
| `content` | String | The text content of a message in the chat log. |
| `name` | String | The name of the user who sent the message in the chat log. This appears exactly as it is displayed in the chat log. |
| `type` | String | The type of message that was received, such as Agent or Visitor. |
| `timestamp` | Date/Time | The date and time the chat message was received. |

## `getChatRequests()`

Returns the chat keys of the chat requests that have been assigned to an agent. Available in API version 29.0 or later.

## Syntax

```
sforce.console.chat.getChatRequests(callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetChatRequests();return false;">Get Chat Requests</a>

    <script type="text/javascript">
        function testGetChatRequests() {
            sforce.console.chat.getChatRequests(function(result) {
                if (result.success) {
                    alert('Number of Chat Requests ' + result.chatKey.length);
                } else {
                    alert('getChatRequests has failed');
                }
            });
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | Array | Array of `chatKey` values, one for each of the current chat requests. |
| `success` | Boolean | `true` if getting chat requests was successful; `false` if getting chat requests wasn't successful. |

## getDetailsByChatKey()

Returns the details of the chat associated with a specific chat key. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.getDetailsByChatKey(chatKey:String, callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which to retrieve details. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetDetailsByChatKey();">Get Chat Details</a>

    <script type="text/javascript">

        function testGetDetailsByChatKey() {
            //Get the value for 'myChatKey' from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            sforce.console.chat.getDetailsByChatKey(chatKey, getDetailsSuccess);
        }

        function getDetailsSuccess(result) {
            //Report whether accepting the chat was succesful
            if (result.success == true) {
                ipAddress = result.details.ipAddress;
                alert('The Visitor IP Address for this chat is: ' + ipAddress);
            } else {
                alert('Getting the details was not successful');
            }
        };


    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `primaryTabId` | String | The ID of the primary tab associated with the chat. |
| `details` | Object | An object that contains all the details for a chat associated with a particular primary tab. |
| `success` | Boolean | `true` if retrieving the details was successful; `false` if retrieving the details wasn't successful. |

## `details`

The `details` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| `acceptTime` | Date/Time | The date and time an agent accepted the chat request. |
| `breadcrumbs` | Array of `breadcrumb` objects | An array of `breadcrumb` objects representing a list of Web pages visited by the visitor before and during the chat. |
| `chatKey` | String | The chat key associated with the chat. |
| `customDetails` | Array of `customDetail` objects | An array of `customDetail` objects that represent custom details that have been passed in to this chat via the Deployment API or Pre-Chat Form API. |
| `geoLocation` | Object | An object representing the details of a chat visitor's location, derived from a geoIP lookup on the chat visitor's IP address. |
| `ipAddress` | String | The IP address of the chat visitor in dot-decimal format. |
| `isEnded` | Boolean | Specifies whether a chat has ended (`true`) or not (`false`). |
| `isEngaged` | Boolean | Specifies whether a chat is currently engaged (`true`) or not (`false`). |
| `isPushRequest` | Boolean | Specifies whether a chat was routed to an agent through a push-based routing method such as Least Active or Most Available (`true`) or not (`false`). |
| `isTransferringOut` | Boolean | Specifies whether a chat is currently in the process of being transferred to another agent (`true`) or not (`false`). |
| `liveChatButtonId` | String | The 15-digit record ID for the chat button from which the chat request originated. |
| `liveChatDeploymentId` | String | The 15-digit record ID for the deployment from which the chat request originated. |
| `name` | String | The name of the chat visitor. |
| `requestTime` | Date/Time | The date and time the chat was requested. |
| `visitorInfo` | Object | An object containing information about the visitor's web browser. |

## `breadcrumb`

A breadcrumb represents a Web page viewed by a chat visitor. The `breadcrumb` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| `location` | String | The URL of a Web page viewed by a chat visitor. |
| `time` | Date/Time | The date and time a chat visitor visited a specific breadcrumb URL. |

## customDetail

Custom details are details have been passed into the chat through the Deployment API or Pre-Chat Form API. The `customDetail` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| *label* | String | The name of the custom detail as specified in the Deployment API or Pre-Chat Form API. |
| *value* | String | The value of the custom detail as specified in the Deployment API or Pre-Chat Form API. |
| `transcriptFields` | Array of Strings | The names of fields where the customer's details on the chat transcript are saved. |
| `entityMaps` | Array of `entityMap` objects | An array of pre-created records used for mapping custom detail information. |

## entityMap

Entities are records that are created when a customer starts a chat with an agent. You can use the API to auto-populate these records with customer details. The `entityMap` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| `entityName` | String | The record to search for or create. |
| `fieldName` | String | The name of the field associated with the details. |
| `isFastFillable` | Boolean | Specifies whether the value can be used to populate the field when an agent creates or edits a record (`true`) or not (`false`) (Live Agent console only). |
| `isAutoQueryable` | Boolean | If you're using the Live Agent console, specifies whether to perform a a SOSL query (in the Live Agent console) (`true`) or not (`false`) to find records with a `fieldName` containing the value.<br>If you're using the Salesforce console, specifies whether to perform a SOQL query (in the Salesforce console) (`true`) or not (`false`) to find records with a `fieldName` containing the value. |
| `isExactMatchable` | Boolean | Specifies whether to only search for records that have fields exactly matching the field `fieldName` (`true`) or not (`false`). |

## geoLocation

The `geoLocation` object represents the details of a chat visitor's location. It contains the following properties:

| Property | Type | Description |
|---|---|---|
| `city` | String | The name of the chat visitor's city. |

| Property | Type | Description |
|---|---|---|
| `countryCode` | String | The two-digit ISO-3166 country code for the chat visitor's country. |
| `countryName` | String | The name of chat visitor's country. |
| `latitude` | String | The chat visitor's approximate latitude. |
| `longitude` | String | The chat visitor's approximate longitude. |
| `organization` | String | The organization name of the chat visitor's internet service provider. |
| `region` | String | The chat visitor's region, such as state or province. |

### `visitorInfo`

The `visitorInfo` object represents information about the visitor's web browser. It contains the following properties:

| Property | Type | Description |
|---|---|---|
| `browserName` | String | The name and version of the chat visitor's web browser. |
| `language` | String | The language of the chat visitor's web browser. |
| `originalReferrer` | String | The original URL of the Web page from which the chat visitor requested a chat. |
| `screenResolution` | String | The screen resolution of the chat visitor's computer, as passed by the chat visitor's browser. |
| `sessionKey` | String | the sessionKey of the visitor which will ultimately be stored on the LiveChatVisitor record as a unique reference to this live chat visitor |

## `getDetailsByPrimaryTabId()`

Returns the details of the chat associated with a specific primary tab ID. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.getDetailsByPrimaryTabId(primaryTabId:String, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `primaryTabId` | String | The ID of the primary tab associated with the chat for which to retrieve details. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetDetailsByPrimaryTabId();">Get Chat Details</a>

    <script type="text/javascript">

        function testGetDetailsByPrimaryTabId() {
            //Get the value for 'myPrimaryTabId'from the getPrimaryTabIds() or
getEnclosingPrimaryTabId() methods.
            //These values are for example purposes only
            var primaryTabId = 'myPrimaryTabId';
           sforce.console.chat.getDetailsByPrimaryTabId(primaryTabId, getDetailsSuccess);

        }

        function getDetailsSuccess(result) {
            //Report whether accepting the chat was succesful
            if (result.success == true) {
                console.log(result);
                chatKey = result.details.chatKey;
                alert('The chatKey for this chat is: ' + chatKey);
            } else {
                alert('Getting the details was not Succesful');
            }
        };

    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `primaryTabId` | String | The ID of the primary tab associated with the chat. |
| `details` | Object | An object that contains all the details for a chat associated with a particular primary tab. |
| `success` | Boolean | `true` if retrieving the details was successful; `false` if retrieving the details wasn't successful. |

## `details`

The `details` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| `acceptTime` | Date/Time | The date and time an agent accepted the chat request. |
| `breadcrumbs` | Array of `breadcrumb` objects | An array of `breadcrumb` objects representing a list of Web pages visited by the visitor before and during the chat. |
| `chatKey` | String | The chat key associated with the chat. |
| `customDetails` | Array of `customDetail` objects | An array of `customDetail` objects that represent custom details that have been passed in to this chat via the Deployment API or Pre-Chat Form API. |
| `geoLocation` | Object | An object representing the details of a chat visitor's location, derived from a geoIP lookup on the chat visitor's IP address. |
| `ipAddress` | String | The IP address of the chat visitor in dot-decimal format. |
| `isEnded` | Boolean | Specifies whether a chat has ended (`true`) or not (`false`). |
| `isEngaged` | Boolean | Specifies whether a chat is currently engaged (`true`) or not (`false`). |
| `isPushRequest` | Boolean | Specifies whether a chat was routed to an agent through a push-based routing method such as Least Active or Most Available (`true`) or not (`false`). |
| `isTransferringOut` | Boolean | Specifies whether a chat is currently in the process of being transferred to another agent (`true`) or not (`false`). |
| `liveChatButtonId` | String | The 15-digit record ID for the chat button from which the chat request originated. |
| `liveChatDeploymentId` | String | The 15-digit record ID for the deployment from which the chat request originated. |
| `name` | String | The name of the chat visitor. |
| `requestTime` | Date/Time | The date and time the chat was requested. |
| `visitorInfo` | Object | An object containing information about the visitor's web browser. |

## `breadcrumb`

A breadcrumb represents a Web page viewed by a chat visitor. The `breadcrumb` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| `location` | String | The URL of a Web page viewed by a chat visitor. |
| `time` | Date/Time | The date and time a chat visitor visited a specific breadcrumb URL. |

## `customDetail`

Custom details are details that have been passed into the chat through the Deployment API or Pre-Chat Form API. The `customDetail` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| *`label`* | String | The name of the custom detail as specified in the Deployment API or Pre-Chat Form API. |
| *`value`* | String | The value of the custom detail as specified in the Deployment API or Pre-Chat Form API. |
| `transcriptFields` | Array of Strings | The names of fields where the customer's details on the chat transcript are saved. |
| `entityMaps` | Array of `entityMap` objects | An array of pre-created records used for mapping the custom detail information. |

## `entityMap`

Entities are records that are created when a customer starts a chat with an agent. You can use the API to auto-populate these records with customer details. The `entityMap` object contains the following properties:

| Property | Type | Description |
|---|---|---|
| `entityName` | String | The record to search for or create. |
| `fieldName` | String | The name of the field associated the details. |
| `isFastFillable` | Boolean | Specifies whether the value can be used to populate the field when an agent creates or edits a record (`true`) or not (`false`) (Live Agent console only). |
| `isAutoQueryable` | Boolean | If you're using the Live Agent console, specifies whether to perform a a SOSL query (in the Live Agent console) (`true`) or not (`false`) to find records with a `fieldName` containing the value.<br>If you're using the Salesforce console, specifies whether to perform a SOQL query (in the Salesforce console) (`true`) or not (`false`) to find records with a `fieldName` containing the value. |
| `isExactMatchable` | Boolean | Specifies whether to only search for records that have fields exactly matching the field `fieldName` (`true`) or not (`false`). |

## `geoLocation`

The `geoLocation` object represents the details of a chat visitor's location. It contains the following properties:

| Property | Type | Description |
|---|---|---|
| `city` | String | The name of the chat visitor's city. |
| `countryCode` | String | The two-digit ISO-3166 country code for the chat visitor's country. |
| `countryName` | String | The name of chat visitor's country. |
| `latitude` | String | The chat visitor's approximate latitude. |

| Property | Type | Description |
|---|---|---|
| `longitude` | String | The chat visitor's approximate longitude. |
| `organization` | String | The organization name of the chat visitor's internet service provider. |
| `region` | String | The chat visitor’s region, such as state or province. |

### visitorInfo

The `visitorInfo` object represents information about the visitor's web browser. It contains the following properties:

| Property | Type | Description |
|---|---|---|
| `browserName` | String | The name and version of the chat visitor's web browser. |
| `language` | String | The language of the chat visitor's web browser. |
| `originalReferrer` | String | The original URL of the Web page from which the chat visitor requested a chat. |
| `screenResolution` | String | The screen resolution of the chat visitor's computer, as passed by the chat visitor’s browser. |
| `sessionKey` | String | the sessionKey of the visitor which will ultimately be stored on the LiveChatVisitor record as a unique reference to this live chat visitor |

## getEngagedChats()

Returns the chat keys of the chats in which the agent is currently engaged. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.getEngagedChats(callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetEngagedChats();return false;">Get Engaged Chats</a>

    <script type="text/javascript">
```

```
        function testGetEngagedChats() {
            sforce.console.chat.getEngagedChats(function(result) {
                if (result.success) {
                    alert('Number Engaged Chats: ' + result.chatKey.length);
                } else {
                    alert('getEngagedChats has failed');
                }
            });
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | array | Array of `chatKey` values, one for each of the currently engaged chats. |
| `success` | boolean | `true` if getting engaged chats was successful; `false` if getting engaged chats wasn’t successful. |

## getMaxCapacity()

Returns the maximum chat capacity for the current agent, as specified in the agent's assigned agent configuration. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.getMaxCapacity(callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testGetMaxCapacity();return false;">Get Max Capacity</a>

    <script type="text/javascript">
```

```
        function testGetMaxCapacity() {
            sforce.console.chat.getMaxCapacity(function(result) {
                if (result.success) {
                    alert('max capacity '+result.count);
                } else {
                    alert('getMaxCapacity failed, agent my not be online');
                }
            });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `count` | integer | Agent's current, maximum chat capacity. |
| `success` | boolean | `true` if getting the agent's capacity was successful; `false` if getting the agent's capacity wasn't successful. |

## `initFileTransfer()`

Initiates the process of transferring a file from a customer to an agent. Available in API version 31.0 or later.

### Syntax

```
sforce.console.chat.initFileTransfer(chatKey:String, entityId:String,
(optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the chat the file is transferred from. |
| `entityId` | String | The ID of the transcript object to attach the transferred file to. |
| `callback` | function | JavaScript method that is called when the method is completed. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <a href="#" onClick="testInitFileTransfer();return false;">Init file transfer</a>

    <script type="text/javascript">
        function testInitFileTransfer() {
           //Gets the value for 'myChatKey'from the getChatRequests() or onChatRequested()
 methods.
            //These values are for example purposes only.
            var chatKey = 'myChatKey';  var entityId = 'myEntityId';
            sforce.console.chat.initFileTransfer(chatKey, entityId, fileSuccess);
        }

        function fileSuccess(result) {
            //Reports whether initiating the file transfer was successful.
            if (result.success == true) {
                alert('Initiating file transfer was successful.');
            } else {
                alert('Initiating file transfer was not successful.');
            }
        };
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if the request to transfer a file was sent successfully; `false` if the request wasn't sent successfully. Note: A value of `true` doesn't necessarily mean that the file was successfully transferred to an agent. Rather, it indicates that the request to begin a file transfer was sent successfully. |

## onAgentSend()

Registers a function to call when an agent sends a chat message through the Salesforce console. This method intercepts the message and occurs before it is sent to the chat visitor. Available in API version 29.0 or later.

Note: This method is only called when an agent sends a message through the chat window interface. This method doesn't apply when a `sendMessage()` method is called in the API.

## Syntax

```
sforce.console.chat.onAgentSend(chatKey:String, callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which to call a function when the agent sends a message. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            var theMessage = result.content;
            alert('The agent is attempting to send the following message: ' +
result.content);
            sforce.console.chat.sendMessage(chatKey, theMessage)
            alert('The following message has been sent: ' + theMessage);
        }
        //Get the value for 'myChatKey' from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
        //These values are for example purposes only
        var chatKey = 'myChatKey';
        sforce.console.chat.onAgentSend(chatKey, eventHandler);
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `content` | String | The text of the agent's message. |
| `name` | String | The name of the agent who is attempting to send the message as it appears in the chat log. |
| `type` | String | The type of message that was received—for example, agent. |
| `timestamp` | Date/Time | The date and time the agent attempted to send the chat message. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## onAgentStateChanged()

Registers a function to call when agents change their Live Agent status, such as from Online to Away. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onAgentStateChanged(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when the agent's Live Agent status has changed. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert("Agent's State has Changed to: " + result.state);
        };
        sforce.console.chat.onAgentStateChanged(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `state` | String | String that represents the agent’s current Live Agent status—for example, Online, Away, or Offline. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn’t successful. |

## onChatCanceled()

Registers a function to call when a chat visitor cancels a chat request. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onChatCanceled(callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page >
<apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
          alert('The chat request has been canceled for this chatKey: ' + result.chatKey);

        }
        sforce.console.chat.onChatCanceled(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | string | The chat key for the chat request that has been canceled. |

## onChatCriticalWaitState()

Registers a function to call when a chat becomes critical to answer or a waiting chat is answered. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onChatCanceled(chatKey:String, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which the critical wait state has changed. |

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('This chat has reached a critical wait');
        }
        //Get the value for 'myChatKey' from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
        //These values are for example purposes only
        var chatKey = 'myChatKey';
        sforce.console.chat.onChatCriticalWaitState(chatKey, eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `state` | Boolean | Indicates whether the chat is in critical wait state (`true`) or not (`false`). |

## onChatDeclined()

Registers a function to call when an agent declines a chat request. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onChatDeclined(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when a chat request is declined. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
           alert('A chat request with this chatKey has been declined: ' + result.chatKey);

        }
        sforce.console.chat.onChatDeclined(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the chat request that has been declined. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## onChatEnded()

Registers a function to call when an engaged chat ends. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onChatEnded(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when an engaged chat ends. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('A chat with this chatKey has ended: ' + result.chatKey);
        }
        sforce.console.chat.onChatEnded(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the engaged chat that has ended. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## `onChatRequested()`

Registers a function to call when an agent receives a chat request. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onChatRequested(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when a chat request is assigned to an agent. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('There is a new incoming chat request with this chatKey: ' +
result.chatKey);
        }
        sforce.console.chat.onChatRequested(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the incoming chat request. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## `onChatStarted()`

Registers a function to call when an agent starts a new chat with a customer. Available in API version 29.0 or later.

### Usage

### Syntax

```
sforce.console.chat.onChatStarted(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when a chat request is accepted and becomes an engaged chat. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('A new engaged chat has started for this chatKey: ' + result.chatKey);
        }
        sforce.console.chat.onChatStarted(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the chat request that has become an engaged chat. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## onChatTransferredOut()

Registers a function to call when an engaged chat is transferred out to another agent. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onChatTransferredOut(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when a chat has been successfully transferred out to another agent. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('A chat with this chatKey has been transferred out: ' + result.chatKey);

        }
        sforce.console.chat.onChatTransferredOut(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the chat that has been transferred. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## onCurrentCapacityChanged()

Registers a function to call when an agent's capacity for accepting chats changes—for example, if an agent accepts a new chat, ends a currently engaged chat, or otherwise changes the number of chats to which they are assigned, or if a chat request is pushed to their chat queue. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onCurrentCapacityChanged(eventHandler:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `eventHandler` | function | JavaScript method called when the agent's capacity for accepting chats has changed. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('Capacity Changed. Current Requests + Engaged Chats is now: ' +
result.count);
        }
        sforce.console.chat.onCurrentCapacityChanged(eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `count` | integer | The number of chats in which the agent is currently engaged plus the number of chat requests currently assigned to the agent. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## onCustomEvent()

Registers a function to call when a custom event takes place during a chat. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onCustomEvent(chatKey:String, type:String, callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which to call a function when a custom event takes place. |
| `type` | String | The name of the custom event you want to listen for. This should match the name of the custom event sent from the chat window. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('A new custom event has been received of type ' + result.type + ' and
with data: ' + result.data );
        }
        //Get the value for 'myChatKey' from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
        //These values are for example purposes only
        var chatKey = 'myChatKey';
        var type = 'myCustomEventType';
        sforce.console.chat.onCustomEvent(chatKey, type, eventHandler);
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `type` | String | The type of the custom event that was sent to this chat; corresponds to the `type` argument of the `liveagent.chasitor.sendCustomEvent()` method used to send this event from the chat window. |
| `data` | String | The data of the custom event that was sent to this chat; corresponds to the `data` argument of the `liveagent.chasitor.sendCustomEvent()` method used to send this event from the chat window. |
| `source` | String | The source of the custom event that was sent to this chat; corresponds to either the agent or the chat visitor, depending on who triggered the custom event. |
| `timestamp` | Date/Time | The time and date the event was received. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## onFileTransferCompleted()

Registers a function to call when a file is transferred from a customer to an agent. Available in API version 31.0 or later.

### Syntax

```
sforce.console.chat.onFileTransferCompleted(chatKey:String, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The chat key for the chat the file was transferred from. |
| `callback` | function | JavaScript method that is called when the method is complete. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/31.0/integration.js"/>
    <a href="#" onClick="testOnFileComplete();return false;">test on file transfer
complete</a>

    <script type="text/javascript">
        function testOnFileComplete() {
           //Gets the value for 'myChatKey'from the getChatRequests() or onChatRequested()
 methods.
            //These values are for example purposes only.
            var chatKey = 'myChatKey';
            sforce.console.chat.onFileTransferCompleted(chatKey, fileSuccess);
        }

        function fileSuccess(result) {
            //Reports status of the file transfer.
            if (result.success == true) {
                alert('File transfer was successful.');
            } else {
                alert('File transfer was not successful.');
            }
        };
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `attachmentId` | String | The ID of the object created for the transferred file. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event was unsuccessful. |

## onNewMessage()

Registers a function to call when a new message is sent from a customer, agent, or supervisor. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onNewMessage(chatKey:String, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | string | The `chatKey` associated with the chat for which to call a function when a new customer message is received. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('There is a new message in this chat: ' + result.content);
        }
        //Get the value for 'myChatKey'from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
        //These values are for example purposes only
        var chatKey = 'myChatKey';
        sforce.console.chat.onNewMessage(chatKey, eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `content` | String | The text of a message in the chat log. |

| Name | Type | Description |
|---|---|---|
| `name` | String | The name of the user who sent the message. This appears exactly as it is displayed in the chat log. |
| `type` | String | The type of message that was received, such as an Agent or Visitor message. |
| `timestamp` | Date/Time | The date and time the message was received. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## `onTypingUpdate()`

Registers a function to call when the customer's text in the chat window changes. If Sneak Peek is enabled, this function is called whenever the customer edits the text in the chat window. If Sneak Peek is not enabled, this function is called whenever a customer starts or stops typing in the chat window. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.onTypingUpdate(chatKey:String, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which to call a function when a customer begins typing a new message to the agent. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <script type="text/javascript">
        var eventHandler = function (result) {
            alert('There is a new typing update in this chat');
        }
        //Get the value for 'myChatKey'from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
        //These values are for example purposes only
        var chatKey = 'myChatKey';
        sforce.console.chat.onTypingUpdate(chatKey, eventHandler);
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `isTyping` | Boolean | Indicates whether a chat visitor is typing (`true`) or not (`false`). |
| `sneakPeek` | String | The text the chat visitor is currently typing into their input box in the chat window. This is visible only if Sneak Peek is enabled for the agent. |
| `success` | Boolean | `true` if firing event was successful; `false` if firing event wasn't successful. |

## sendCustomEvent()

Sends a custom event to the client-side chat window for a chat with a specific chat key. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.sendCustomEvent(chatKey:String, type:String, data:String,
callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat to which to send a custom event. |
| `type` | String | The name of the custom event you want to send to the chat window. |
| `data` | String | Additional data you want to send to the chat window along with the custom event. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testSendCustomEvent();">Send Custom Event</a>

    <script type="text/javascript">

        function testSendCustomEvent() {
            //Get the value for 'myChatKey'from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            var type = 'myCustomEventType'
            var data = 'myCustomEventData'
          sforce.console.chat.sendCustomEvent(chatKey, type, data, sendCustomEventSuccess);

        }

        function sendCustomEventSuccess(result) {
            //Report whether sending the custom event was successful
            if (result.success == true) {
                alert('The customEvent has been sent');
            } else {
                alert('Sending the customEvent was not successful');
            }
        };

    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if sending the custom event was successful; `false` if sending the custom event wasn't successful. |

## sendMessage()

Sends a new chat message from the agent to a chat with a specific chat key. Available in API version 29.0 or later.

## Syntax

```
sforce.console.chat.sendMessage(chatKey:String, message:String, callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` of the chat where the agent's message is sent. |
| `message` | String | The message you would like to send from the agent to the customer in a chat. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testSendMessage();">Send Message</a>

    <script type="text/javascript">

        function testSendMessage() {
            //Get the value for 'myChatKey'from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            var text ='This is sample text to send as a message';
            sforce.console.chat.sendMessage(chatKey, text, sendMessageSuccess);
        }

        function sendMessageSuccess(result) {
            //Report whether getting the chat log was successful
            if (result.success == true) {
                alert('Message Sent');
            } else {
                alert('Sending the message was not successful');
            }
        };


    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if sending the message was successful; `false` if sending the message wasn't successful. |

## `setAgentInput()`

Sets the string of text in the agent's text input area in the chat log of a chat with a specific chat key.Available in API version 29.0 or later.

## Syntax

```
sforce.console.chat.setAgentInput(chatKey:String, text:String, callback:Function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `chatKey` | String | The `chatKey` associated with the chat for which to set the agent's input text. |
| `text` | String | The string of text which you want to set into an agent's input. |
| `callback` | function | JavaScript method called upon completion of the method. |

## Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testSetAgentInput();">Set Agent Input</a>

    <script type="text/javascript">

        function testSetAgentInput() {
            //Get the value for 'myChatKey'from the
sforce.console.chat.getDetailsByPrimaryTabId() or other chat methods.
            //These values are for example purposes only
            var chatKey = 'myChatKey';
            var text = 'This is example text to set the agent input'
            sforce.console.chat.setAgentInput(chatKey, text, setAgentInputSuccess);
        }

        function setAgentInputSuccess(result) {
            //Report whether setting the agent's input was succesful
            if (result.success == true) {
                alert('The text in the agent input has been updated');
            } else {
                alert('Setting the agent input was not Succesful');
            }
        };


    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if setting the agent's input was successful; `false` if setting the agent's input wasn't successful. |

## `setAgentState()`

Sets an agent's Live Agent status, such as Online, Away, or Offline. Available in API version 29.0 or later.

### Syntax

```
sforce.console.chat.setAgentState(state:String, (optional)callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `state` | String | Live Agent status you want to set the agent to—for example, Online, Away, or Offline. |
| `callback` | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/29.0/integration.js"/>
    <a href="#" onClick="testSetAgentState('Online');return false;">Set Agent Status to
Online</a>
    <script type="text/javascript">
        function testSetAgentState(state) {
            sforce.console.chat.setAgentState(state, function(result) {
                if (result.success) {
                    alert('Agent State Set to Online');
                } else {
                    alert('setAgentState has failed');
                }
            });
        }
    </script>
</apex:page>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if setting the agent's Live Agent status was successful; `false` if setting the agent's Live Agent status wasn't successful. |

# Methods for Live Agent Chat Visitors

There are a few methods available that you can use to customize the chat visitor experience for Live Agent in a custom Visualforce chat window.

IN THIS SECTION:

## chasitor.addCustomEventListener()

Registers a function to call when a custom event is received in the chat window. Available in API version 29.0 or later.

### Syntax

```
liveagent.chasitor.addCustomEventListener(type:String, callback:Function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| type | string | The type of custom event you want to listen for. |
| callback | function | JavaScript method called upon completion of the method. |

### Sample Code–Visualforce

```
<script type="text/javascript">
        function testAddCustomEventListener() {
            //These values are for example purposes only
            var type = 'myCustomEventType'
            liveagent.chasitor.addCustomEventListener(type, customEventReceived)
        }

        function customEventReceived(result) {
            eventType = result.getType();
            eventData = result.getData();
            alert('A custom event of type: ' + eventType + ' has been received with the
following data: ' + eventData);
        };
        testAddCustomEventListener();
</script>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following methods:

| Name | Type | Description |
|---|---|---|
| `getType` | method | Accesses the type of the custom event that was sent to this chat window. Returns the `type` argument of the `sforce.console.chat.sendCustomEvent()` method used to send this event. |
| `getData` | method | Accesses the data of the custom event that was sent to this chat window. Returns the `data` argument of the `sforce.console.chat.sendCustomEvent()` method used to send this event. |
| `getSource` | method | Accesses the source of the custom event that was sent to this chat window—for example, agent or chat visitor. |
| `getDate` | method | Accesses the date of the custom event that was sent to this chat window. Returns the date and time the event was received. |

## `chasitor.getCustomEvents()`

Retrieves a list of custom events that have been received in this chat window during this chat session. Available in API version 29.0 or later.

### Syntax

```
liveagent.chasitor.getCustomEvents()
```

### Sample Code–Visualforce

```
<a href="#" onClick="testGetCustomEvents();">Get Custom Events</a>

<script type="text/javascript">
        function testGetCustomEvents() {
            events = liveagent.chasitor.getCustomEvents();
            eventsCount = events.length;
            alert('The following number of custom events have occurred: ' + eventsCount);

        };
</script>
```

### Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following methods and properties:

| Name | Type | Description |
|---|---|---|
| `events` | Array of `event` objects | An array of `event` objects. Each object represents a custom event that has occurred in this chat. Data on each message object can be accessed by the following methods: <br>• `getType()` <br>• `getData()` <br>• `getSource()` <br>• `getDate()` |
| `getType` | method | Accesses the type of the custom event that was sent to this chat window. Returns the `type` argument of the `sforce.console.chat.sendCustomEvent()` method used to send this event. |
| `getData` | method | Accesses the data of the custom event that was sent to this chat window. Returns the `data` argument of the `sforce.console.chat.sendCustomEvent()` method used to send this event. |
| `getSource` | method | Accesses the source of the custom event that was sent to this chat window—for example, agent or chat visitor. |
| `getDate` | method | Accesses the date of the custom event that was sent to this chat window. Returns the date and time the event was received. |

## `chasitor.sendCustomEvent()`

Sends a custom event to the agent console of the agent who is currently chatting with a customer. Available in API version 29.0 or later.

### Syntax

```
liveagent.chasitor.sendCustomEvent(type:String, data:String)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `type` | string | The name of the custom event to send to the agent console. |
| `data` | string | Additional data you want to send to the agent console along with the custom event. |

## Sample Code–Visualforce

```
<a href="#" onClick="testSendCustomEvent();">Send Custom Event</a>

<script type="text/javascript">
        function testSendCustomEvent() {
            type = 'myCustomEventType';
            data = 'myCustomEventData';
            liveagent.chasitor.sendCustomEvent(type, data);
            alert('The custom event has been sent');
        };
</script>
```

## Response

This method returns no responses.

# CHAPTER 10 Methods for Omni-Channel

Omni-Channel is a comprehensive customer service solution that lets your call center route any type of incoming work item—including cases, chats, phone calls, or leads—to the most qualified, available agents in your organization. Omni-Channel provides a customizable customer service solution that integrates seamlessly into the Salesforce console and benefits your customers and support agents.

For more information on Omni-Channel, see *Set Up Omni-Channel*.

IN THIS SECTION:

## acceptAgentWork

Accepts a work item that's assigned to an agent. Available in API versions 32.0 and later.

### Syntax

```
sforce.console.presence.acceptAgentWork(workId:String, (optional) callback:function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `workId` | String | The ID of the work item the agent accepts. |
| `callback` | function | JavaScript method to call when an agent accepts the work item associated with the `workId`. |

### Sample Code–Visualforce

```
<apex:page>
     <apex:includeScript value="/support/console/34.0/integration.js"/>
     <a href="#" onClick="testAcceptWork();return false;">Accept Assigned Work Item</a>

     <script type="text/javascript">
         function testAcceptWork() {
             //First get the ID of the assigned work item to accept it
             sforce.console.presence.getAgentWorks(function(result) {
                 if (result.success) {
                     var works = JSON.parse(result.works);
                     var work = works[0];
                     if (!work.isEngaged) {
                         //Now that we have the assigned work item ID, we can accept it
                         sforce.console.presence.acceptAgentWork(work.workId,
function(result) {
                             if (result.success) {
                                 alert('Accepted work successfully');
                             } else {
                                 alert('Accept work failed');
                             }
                         });
                     } else {
                         alert('The work item has already been accepted');
                     }
                 }
             });
         }
     </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if accepting the work item was successful; `false` if accepting the work item wasn't successful. |

## closeAgentWork

Changes the status of a work item to "Closed" and removes it from the list of work items in the Omni-Channel widget. Available in API versions 32.0 and later.

### Syntax

```
sforce.console.presence.closeAgentWork(workId:String, (optional) callback:function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `workId` | String | The ID of the work item that's closed. |
| `callback` | function | JavaScript method to call when the work item associated with the `workId` is closed. |

### Sample Code–Visualforce

```
<apex:page>
     <apex:includeScript value="/support/console/34.0/integration.js"/>
     <a href="#" onClick="testCloseWork();return false;">Close Engaged Work Item</a>
     <script type="text/javascript">
         function testCloseWork() {
             //First get the ID of the engaged work item to close it
             sforce.console.presence.getAgentWorks(function(result) {
                 if (result.success) {
                     var works = JSON.parse(result.works);
                     var work = works[0];
                     if (work.isEngaged) {
                         //Now that we have the engaged work item ID, we can close it
                          sforce.console.presence.closeAgentWork(work.workId,function(result)
 {

                                 if (result.success) {
                                      alert('Closed work successfully');
                                 } else {
```

```
                                alert('Close work failed');
                            }
                        });
                    } else {
                        alert('The work item should be accepted first');
                    }
                }
            });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if closing the work item was successful; `false` if closing the work item wasn't successful. |

## `declineAgentWork`

Declines a work item that's assigned to an agent. Available in API versions 32.0 and later.

### Syntax

```
sforce.console.presence.declineAgentWork(workId:String, (optional) callback:function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `workId` | String | The ID of the work item that the agent declines. |
| `callback` | function | JavaScript method to call when an agent declines the work item associated with the `workId`. |

### Sample Code–Visualforce

```
<apex:page >
    <apex:includeScript value="/support/console/34.0/integration.js"/>
    <a href="#" onClick="testAcceptWork();return false;">Accept Assigned Work Item</a>

    <script type="text/javascript">
```

```
        function testAcceptWork() {
            //First get the ID of the assigned work item to accept it
            sforce.console.presence.getAgentWorks(function(result) {
                if (result.success) {
                    var works = JSON.parse(result.works);
                    var work = works[0];
                    if (!work.isEngaged) {
                        //Now that we have the assigned work item ID, we can accept it
                     sforce.console.presence.acceptAgentWork(work.workId, function(result)
 {
                            if (result.success) {
                                alert('Accepted work successfully');
                            } else {
                                alert('Accept work failed');
                            }
                        });
                    } else {
                        alert('The work item has already been accepted');
                    }
                }
            });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if declining the work item was successful; `false` if declining the work item wasn't successful. |

## `getAgentWorks`

Returns a list of work items that are currently assigned to an agent and open in the agent's workspace. Available in API versions 32.0 and later.

### Syntax

```
sforce.console.presence.getAgentWorks((optional) callback:function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| callback | function | JavaScript method to call when the list of an agent's work items is retrieved. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/34.0/integration.js"/>
    <a href="#" onClick="testGetWorks();return false;">Get Agent’s Current Work Items</a>


    <script type="text/javascript">
        function testGetWorks() {
            //These values are for example purposes only.
            sforce.console.presence.getAgentWorks(function(result) {
                if (result.success) {
                    alert('Get work items successful');
                    var works = JSON.parse(result.works);
                    alert('First Agent Work ID is: ' +  works[0].workId);
                    alert('Assigned Entity Id of the first Agent Work is: ' +
works[0].workItemId);
                    alert('Is first Agent Work Engaged: ' + works[0].isEngaged);
                } else {
                    alert('Get work items failed');
                }
            });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| success | Boolean | true if retrieving the agent's work items was successful; false if retrieving the agent's work items wasn't successful. |
| works | JSON string of work objects | A JSON string of work objects that represents the work items assigned to the agent that are open in the agent's workspace. |

### work

The work object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `workItemId` | String | The ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent. |
| `workId` | String | The ID of a work assignment that's routed to an agent. |
| `isEngaged` | Boolean | Indicates whether an agent is working on a work item that's been assigned to them (`true`) or not (`false`). |

## getAgentWorkload

In API version 35.0 and later, we can retrieve an agent's currently assigned workload. Use this method for rerouting work to available agents.

### Syntax

```
sforce.console.presence.getAgentWorkload((optional) callback:function)
```

### Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method to call when the agent's configured capacity and work retrieved. |

### Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/35.0/integration.js"/>
    <a href="#" onClick="testGetAgentWorkload();return false;">
        Get Agent's configured capacity and current workload
    </a>

    <script type="text/javascript">
        function testGetAgentWorkload() {
            sforce.console.presence.getAgentWorkload(function(result) {
                if (result.success) {
                    alert('Retrieved Agent Configured Capacity and Current Workload
successfully');
                  alert('Agent\'s configured capacity is: ' + result.configuredCapacity);

                    alert('Agent\'s currently assigned workload is: ' +
result.currentWorkload);
                } else {
                    alert('Get Agent Workload failed');
                }
            });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if retrieving the agent's work items was successful; `false` if retrieving the agent's work items wasn't successful. |
| `configuredCapacity` | Number | Indicates the agent's configured capacity (work that's assigned to the current user) through Presence Configuration. |
| `currentWorkload` | Number | Indicates the agent's currently assigned workload. |

### `work`

The `work` object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `workItemId` | String | The ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent. |
| `workId` | String | The ID of a work assignment that's routed to an agent. |
| `isEngaged` | Boolean | Indicates whether an agent is working on a work item that's been assigned to them (`true`) or not (`false`). |

## `getServicePresenceStatusChannels`

Retrieves the service channels that are associated with an Omni-Channel user's current presence status. Available in API versions 32.0 and later.

## Syntax

```
sforce.console.presence.getServicePresenceStatusChannels((optional) callback:function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method to call when the channels associated with a presence status are retrieved. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/34.0/integration.js"/>
    <a href="#" onClick="testGetChannels();return false;">
        Get Channels Associated with a Presence Status
    </a>

    <script type="text/javascript">
        function testGetChannels() {
            //These values are for example purposes only.
            sforce.console.presence.getServicePresenceStatusChannels(function(result) {
                if (result.success) {
                    alert('Retrieved Service Presence Status Channels successfully');
                    var channels = JSON.parse(result.channels);
                    //For example purposes, just retrieve the first channel
                    alert('First channel ID is: ' + channels[0].channelId);
                   alert('First channel developer name is: ' + channels[0].developerName);

                } else {
                    alert('Get Service Presence Status Channels failed');
                }
            });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if retrieving the current presence status channels was successful; `false` if the retrieving the current presence status channels wasn't successful. |
| `channels` | JSON string of `channel` objects | Returns the IDs and API names of the channels associated with the presence status. |

## getServicePresenceStatusId

Retrieves an agent's current presence status. Available in API versions 32.0 and later.

## Syntax

```
sforce.console.presence.getServicePresenceStatusId((optional) callback:function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method to call when the agent's presence status is retrieved. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/34.0/integration.js"/>
    <a href="#" onClick="testGetStatusId();return false;">Get Omni-Channel Status ID</a>

    <script type="text/javascript">
        function testGetStatusId() {
            sforce.console.presence.getServicePresenceStatusId(function(result) {
                if (result.success) {
                    alert('Get Status Id successful');
                    alert('Status Id is: ' + result.statusId);
                } else {
                    alert('Get Status Id failed');
                }
            });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if retrieving the presence status ID was successful; `false` if the retrieving the presence status ID wasn't successful. |
| `statusId` | String | The ID of the agent's current presence status. |

## login

Logs an agent into Omni-Channel with a specific presence status. Available in API versions 32.0 and later.

## Syntax

```
sforce.console.presence.login(statusId:String, (optional) callback:function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `statusId` | String | The ID of the presence status. Agents must be given access to this presence status through their associated profile or permission set. |
| `callback` | function | JavaScript method to call when the agent is logged in with the presence status associated with `statusId`. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/32.0/integration.js"/>
    <a href="#" onClick="testLogin('0N5xx00000000081');return false;">Log In to
Omni-Channel</a>

    <script type="text/javascript">
        function testLogin(statusId) {
            //Gets the Salesforce ID of the presence status entity which the current user
 has been assigned through their permission set or profile.
            //These values are for example purposes only.
            sforce.console.presence.login(statusId, function(result) {
                if (result.success) {
                    alert('Login successful');
                } else {
                    alert('Login failed');
                }
           });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if the login was successful; `false` if the login wasn't successful. |

## `logout`

Logs an agent out of Omni-Channel. Available in API versions 32.0 and later.

## Syntax

```
sforce.console.presence.logout((optional) callback:function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `callback` | function | JavaScript method to call when the agent is logged out of Omni-Channel. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/32.0/integration.js"/>
    <a href="#" onClick="testLogout();return false;">Log out of Omni-Channel</a>

    <script type="text/javascript">
        function testLogout() {
            sforce.console.presence.logout(function(result) {
                if (result.success) {
                    alert('Logout successfully');
                } else {
                    alert('Logout failed');
                }
           });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if the logout was successful; `false` if the logout wasn't successful. |

## setServicePresenceStatus

Sets an agent's presence status to a status with a particular ID. In API version 35.0 and later, we log the user into presence if that user is not already logged in. This will remove the need for you to make additional calls.

## Syntax

```
sforce.console.presence.setServicePresenceStatus(statusId:String,
      (optional) callback:function)
```

## Arguments

| Name | Type | Description |
|---|---|---|
| `statusId` | String | The ID of the presence status you want to set the agent to. Agents must be given access to this presence status through their associated profile or permission set. |
| `callback` | function | JavaScript method to call when the agent's status is changed to the presence status associated with `statusId`. |

## Sample Code–Visualforce

```
<apex:page>
    <apex:includeScript value="/support/console/32.0/integration.js"/>
    <a href="#" onClick="testSetStatus('0N5xx00000000081');return false;">Set Presence
Status</a>

    <script type="text/javascript">
        function testSetStatus(statusId) {

            //Sets the user’s presence status to statusID. Assumes that the user was
assigned this presence status through Setup.
            //These values are for example purposes only
            sforce.console.presence.setServicePresenceStatus(statusId, function(result) {

                if (result.success) {
                    alert('Set status successful');
                    alert('Current statusId is: ' + result.statusId);
                    alert('Channel list attached to this status is: ' + result.channels);
 //printout in console for lists
                } else {
                    alert('Set status failed');
                }
           });
        }
    </script>
</apex:page>
```

## Response

This method is asynchronous so it returns its response in an object in a callback method. The response object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `success` | Boolean | `true` if setting the agent's status was successful; `false` if setting the agent's status wasn't successful. |
| `statusId` | String | The ID of the agent's current presence status. |
| `channels` | JSON string of `channel` objects | Returns the IDs and API names of the channels associated with the presence status. |

## Methods for Omni-Channel Console Events

JavaScript can be executed when certain types of events occur in a console, such as when a user closes a tab. In addition to the standard methods for console events, there are a few events that are specific to Omni-Channel.

### Omni-Channel Console Events

| Event | Description | Payload |
|---|---|---|
| `sforce.console.ConsoleEvent.PRESENCE.LOGIN_SUCCESS` | Fired when a Omni-Channel user logs into Omni-Channel successfully.<br>Available in API version 32.0 or later. | • `statusId`—the ID of the agent's current presence status |
| `sforce.console.ConsoleEvent.PRESENCE.STATUS_CHANGED` | Fired when a user changes his or her presence status.<br>Available in API version 32.0 or later. | • `statusId`—the ID of the agent's current presence status<br>• `channels`—an array of `channel` objects on page 175 |
| `sforce.console.ConsoleEvent.PRESENCE.SFDC_LOGOUT` | Fired when a user logs out of Omni-Channel.<br>Available in API version 32.0 or later. | None |
| `sforce.console.ConsoleEvent.PRESENCE.WORK_ASSIGNED` | Fired when a user is assigned a new work item.<br>Available in API version 32.0 or later. | • `workItemId`—the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent.<br>• `workId`—the ID of a work assignment that's routed to an agent. |
| `sforce.console.ConsoleEvent.PRESENCE.WORK_ACCEPTED` | Fired when a user accepts a work assignment, or | • `workItemId`—the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent. |

| Event | Description | Payload |
|---|---|---|
| | when a work assignment is automatically accepted.<br><br>Available in API version 32.0 or later. | • `workId`—the ID of a work assignment that's routed to an agent. |
| `sforce.console.ConsoleEvent.PRESENCE.WORK_DECLINED` | Fired when a user declines a work assignment.<br><br>Available in API version 32.0 or later. | • `workItemId`—the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent.<br>• `workId`—the ID of a work assignment that's routed to an agent. |
| `sforce.console.ConsoleEvent.PRESENCE.WORK_CLOSED` | Fired when a user closes a tab in the console that's associated with a work item. When the tab for that work item is closed, the status of the Omni-Channel object associated with it automatically changes to "Closed."<br><br>Available in API version 32.0 or later. | • `workItemId` —the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent.<br>• `workId` — the ID of a work assignment that's routed to an agent. |

## `channel`

The `channel` object contains the following functions:

| Name | Type | Description |
|---|---|---|
| `channelId` | function | Retrieves the ID of a service channel that's associated with a presence status. |
| `getDeveloperName` | function | Retrieves the developer name of the the service channel that's associated with the `channelId`. |

# GLOSSARY

A |B |C |D |E |F |G |H |I |J |K |L |M |N |O |P |Q |R |S |T |U |V |W |X |Y |Z

## A

**Administrator (System Administrator)**
One or more individuals in your organization who can configure and customize the application. Users assigned to the System Administrator profile have administrator privileges.

**Application Programming Interface (API)**
The interface that a computer system, library, or application provides to allow other computer programs to request services from it and exchange data.

**Asynchronous Calls**
A call that does not return results immediately because the operation may take a long time. Calls in the Metadata API and Bulk API are asynchronous.

## B

**Boolean Operators**
You can use Boolean operators in report filters to specify the logical relationship between two values. For example, the AND operator between two values yields search results that include both values. Likewise, the OR operator between two values yields search results that include either value.

## C

**Custom Links**
Custom links are URLs defined by administrators to integrate your Salesforce data with external websites and back-office systems. Formerly known as Web links.

## D

**Database**
An organized collection of information. The underlying architecture of the Force.com platform includes a database where your data is stored.

**Database Table**
A list of information, presented with rows and columns, about the person, thing, or concept you want to track. See also Object.

**Developer Edition**
A free, fully-functional Salesforce organization designed for developers to extend, integrate, and develop with the Force.com platform. Developer Edition accounts are available on developer.salesforce.com.

**Salesforce Developers**

The Salesforce Developers website at developer.salesforce.com provides a full range of resources for platform developers, including sample code, toolkits, an online developer community, and the ability to obtain limited Force.com platform environments.

## E

**Enterprise Edition**

A Salesforce edition designed for larger, more complex businesses.

## F

**Field**

A part of an object that holds a specific piece of information, such as a text or currency value.

**Field-Level Security**

Settings that determine whether fields are hidden, visible, read only, or editable for users. Available in Enterprise, Unlimited, Performance, and Developer Editions only.

**Force.com**

The Salesforce platform for building applications in the cloud. Force.com combines a powerful user interface, operating system, and database to allow you to customize and deploy applications in the cloud for your entire enterprise.

## G

**Group Edition**

A product designed for small businesses and workgroups with a limited number of users.

## H

No Glossary items for this entry.

## I

**ID**

See Salesforce Record ID.

**Instance**

The cluster of software and hardware represented as a single logical server that hosts an organization's data and runs their applications. The Force.com platform runs on multiple instances, but data for any single organization is always stored on a single instance.

**Interaction Log**

An area in a Salesforce console where you can jot notes about the main record you're working on without clicking a button, viewing a new tab, or scrolling to the Notes & Attachments related list. Interaction logs are archived on the Activity History related list for easy review and retrieval. Administrators can customize interaction logs to include task fields.

## J

No Glossary items for this entry.

## K

No Glossary items for this entry.

## L

**Logged-in User**
In a SOAP API context, the username used to log into Salesforce. Client applications run with the permissions and sharing of the logged-in user. Also referred to as an integration user.

## M

**Metadata**
Information about the structure, appearance, and functionality of an organization and any of its parts. Force.com uses XML to describe metadata.

**Multitenancy**
An application model where all users and apps share a single, common infrastructure and code base.

## N

**Navigation Tab**
A tab with a drop-down button in a Salesforce console that lets you select and view object home pages.

## O

**Object**
An object allows you to store information in your Salesforce organization. The object is the overall definition of the type of information you are storing. For example, the case object allow you to store information regarding customer inquiries. For each object, your organization will have multiple records that store the information about specific instances of that type of data. For example, you might have a case record to store the information about Joe Smith's training inquiry and another case record to store the information about Mary Johnson's configuration issue.

**Organization**
A deployment of Salesforce with a defined set of licensed users. An organization is the virtual space provided to an individual customer of Salesforce. Your organization includes all of your data and applications, and is separate from all other organizations.

## P

**Personal Edition**
Product designed for individual sales representatives and single users.

**Platform Edition**
A Salesforce edition based on Enterprise, Unlimited, or Performance Edition that does not include any of the standard Salesforce apps, such as Sales or Service & Support.

**Primary Tab**
A tab in a Salesforce console that displays the main item to work on, such as an account.

**Production Organization**
A Salesforce organization that has live users accessing data.

**Professional Edition**
A Salesforce edition designed for businesses who need full-featured CRM functionality.

## Q

No Glossary items for this entry.

## R

**Record**
A single instance of a Salesforce object. For example, "John Jones" might be the name of a contact record.

## S

**Salesforce Record ID**
A unique 15- or 18-character alphanumeric string that identifies a single record in Salesforce.

**Sandbox**
A nearly identical copy of a Salesforce production organization for development, testing, and training. The content and size of a sandbox varies depending on the type of sandbox and the editioin of the production organization associated with the sandbox.

**Salesforce Console**
The Salesforce console is designed for users in fast-paced environments who need to find, update, and create records quickly. It improves upon the Agent Console in the Console tab by displaying records and related items as tabs on one screen.

**Session ID**
An authentication token that is returned when a user successfully logs in to Salesforce. The Session ID prevents a user from having to log in again every time he or she wants to perform another action in Salesforce. Different from a record ID or Salesforce ID, which are terms for the unique ID of a Salesforce record.

**Session Timeout**
The period of time after login before a user is automatically logged out. Sessions expire automatically after a predetermined length of inactivity, which can be configured in Salesforce from Setup by clicking **Security Controls**. The default is 120 minutes (two hours). The inactivity timer is reset to zero if a user takes an action in the Web interface or makes an API call.

**Sharing**
Allowing other users to view or edit information you own. There are different ways to share data:

- Sharing Model—defines the default organization-wide access levels that users have to each other's information and whether to use the hierarchies when determining access to data.
- Role Hierarchy—defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.
- Sharing Rules—allow an administrator to specify that all information created by users within a given group or role is automatically shared to the members of another group or role.
- Manual Sharing—allows individual users to share records with other users or groups.
- Apex-Managed Sharing—enables developers to programmatically manipulate sharing to support their application's behavior. See Apex-Managed Sharing.

**SOAP (Simple Object Access Protocol)**
A protocol that defines a uniform way of passing XML-encoded data.

**Standard Object**
A built-in object included with the Force.com platform. You can also build custom objects to store information that is unique to your app.

**System Log**
Part of the Developer Console, a separate window console that can be used for debugging code snippets. Enter the code you want to test at the bottom of the window and click Execute. The body of the System Log displays system resource information, such as how long a line took to execute or how many database calls were made. If the code did not run to completion, the console also displays debugging information.

## T

**Test Organization**
See Sandbox.

**Trigger**
A piece of Apex that executes before or after records of a particular type are inserted, updated, or deleted from the database. Every trigger runs with a set of context variables that provide access to the records that caused the trigger to fire, and all triggers run in bulk mode—that is, they process several records at once, rather than just one record at a time.

## U

**Unlimited Edition**
Unlimited Edition is Salesforce's solution for maximizing your success and extending that success across the entire enterprise through the Force.com platform.

**URL (Uniform Resource Locator)**
The global address of a website, document, or other resource on the Internet. For example, http://www.salesforce.com.

## V

**Version**
A number value that indicates the release of an item. Items that can have a version include API objects, fields and calls; Apex classes and triggers; and Visualforce pages and components.

**Visualforce**

A simple, tag-based markup language that allows developers to easily define custom pages and components for apps built on the platform. Each tag corresponds to a coarse or fine-grained component, such as a section of a page, a related list, or a field. The components can either be controlled by the same logic that is used in standard Salesforce pages, or developers can associate their own logic with a controller written in Apex.

## W

**Web Service**

A mechanism by which two applications can easily exchange data over the Internet, even if they run on different platforms, are written in different languages, or are geographically remote from each other.

**Web Services API**

A Web services application programming interface that provides access to your Salesforce organization's information. See also SOAP API and Bulk API.

**Wrapper Class**

A class that abstracts common functions such as logging in, managing sessions, and querying and batching records. A wrapper class makes an integration more straightforward to develop and maintain, keeps program logic in one place, and affords easy reuse across components. Examples of wrapper classes in Salesforce include theAJAX Toolkit, which is a JavaScript wrapper around the Salesforce SOAP API, wrapper classes such as `CCritical Section` in the CTI Adapter for Salesforce CRM Call Center, or wrapper classes created as part of a client integration application that accesses Salesforce using the SOAP API.

**WSDL (Web Services Description Language) File**

An XML file that describes the format of messages you send and receive from a Web service. Your development environment's SOAP client uses the Salesforce Enterprise WSDL or Partner WSDL to communicate with Salesforce using the SOAP API.

## X

No Glossary items for this entry.

## Y

No Glossary items for this entry.

## Z

No Glossary items for this entry.

# INDEX