Open CTI Developer Guide

Version 36.0, Spring '16

@salesforcedocs

Last updated: February 17, 2016

CONTENTS

CHAPTER 1 Introducing Open CTI

Salesforce CRM Call Center seamlessly integrates Salesforce with third-party computer-telephony integration (CTI) systems. Before the introduction of Open CTI, Salesforce users could only use the features of a CTI system after they installed a CTI adapter program on their machines. Yet such programs often included desktop software that required maintenance and didn't offer the benefits of cloud architecture. Open CTI lets developers:

- Build CTI systems that integrate with Salesforce without the use of CTI adapters.
- Create customizable SoftPhones (call-control tools) that function as fully integrated parts of Salesforce and the Salesforce console.
- Provide users with CTI systems that are browser and platform agnostic, for example, CTI for Microsoft® Internet Explorer®, Mozilla® Firefox®, Apple® Safari®, or Google Chrome™ on Mac, Linux, or Windows machines.

EDITIONS

Salesforce console available in Salesforce Classic and App Launcher in Lightning Experience. Setup for Salesforce console available in Salesforce Classic.

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions with the Service Cloud

Available in: **Performance** and **Developer** Editions with the Sales Cloud

Available in: **Enterprise** and **Unlimited** Editions with the Sales Cloud for an extra cost

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

- CTI
- JavaScript
- Visualforce
- Web services
- Software development
- The Salesforce console
- Salesforce CRM Call Center

This guide explains how to use Open CTI in JavaScript to embed API calls and processes. Open CTI is only available for use with JavaScript pages; the examples in this guide are in JavaScript. The functionality it describes is available to your organization if you have Professional, Enterprise, Unlimited, Performance, or Developer Edition.

Open CTI is a browser-based JavaScript API. It uses browsers as clients to display SoftPhones. Open CTI:

- Matches the API version for any given release. For example, if the current version of SOAP API is 25.0, then there's also a version 25.0 of Open CTI.
- Supports the following minimum browser requirements: Internet Explorer 8; Firefox 3.6; Safari 4; or Chrome 11.0.

When to Use Open CTI

Open CTI helps advanced administrators and developers build and integrate third-party computer-telephony integration (CTI) systems with Salesforce so that Salesforce users can use a SoftPhone without installing CTI adapters on their machines. For example, you can use Open CTI to integrate custom SoftPhones with Salesforce so that users can access the features of a CTI system without having to download and install client applications. Open CTI is an API that uses browsers as clients to display CTI functionality in Salesforce.

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). 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: • Build wizards and other multistep processes. • Create your own custom flow control through an application. • Define navigation patterns and data-specific rules for optimal, efficient application interaction. For more information, see the *Visualforce Developer's Guide*.
Salesforce Console Integration Toolkit	The Salesforce Console Integration Toolkit lets you 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.
Apex	Use Apex if you want to: • Create Web services. • Create email services. • Perform complex validation over multiple objects. • Create complex business processes that are not supported by workflow. • Create custom transactional logic (logic that occurs over the entire transaction, not just with a single record or object). • 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. For more information, see the *Apex Developer Guide*.

Open CTI Support Policy

The current release of Open CTI is the only version that receives enhancements. Previous versions might or might not receive fixes. When a new version is released, the previous version remains available.

Backward Compatibility

Salesforce strives to make backward compatibility easy when using Open CTI.

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 '12 release included API version 25.0 and the Winter '13 release included API version 26.0.

The version of Open CTI matches the API version for any given release. So if the current version of the API is 26.0, there's also a version 26.0 of Open CTI.

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

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

For example, an application written using Open CTI version 25.0, which shipped with the Summer '12 release, will continue to work with Open CTI version 25.0 on the Winter '13 release and on future releases. However, that same application might not work with Open CTI version 26.0 without modifications to the application.

API Support

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

When a Open CTI version is scheduled to be unsupported, a no-longer-available notice will be given at least one year before support for the version ends. Salesforce will directly notify customers using Open CTI versions that will no longer be available.

Other Resources

In addition to this guide, there are other resources available for you as you learn how to use Open CTI:

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

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.

Open CTI Typographical Conventions

The Open CTI guide uses the following 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: `Public class HelloWorld`

Convention	Description
`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`*]; 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: `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.
	`<apex:pageBlockTable value="{!account.Contacts}" var="contact">`
	
	`   <apex:column value="{!contact.Name}"/>`
	`   <apex:column value="{!contact.MailingCity}"/>`
	`   <apex:column value="{!contact.Phone}"/>`
	`</apex:pageBlockTable>`

{ }	In descriptions of syntax, braces ({ }) are typed exactly as shown.
	`<apex:page>`
	`    Hello {!$User.FirstName}!`
	`</apex:page>`

[ ]	In descriptions of syntax, anything included in brackets is optional. In the following example, specifying ***`value`*** is optional: ***`data_type variable_name`*** `[ =` ***`value`***`];`

|	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 HTMLPage Using Open CTI

This example shows how to add CTI functionality to the Salesforce user interface using Open CTI. It assumes that you’ve already imported a call center definition file into your Salesforce organization.

1. Create an HTML page.
2. Cut and paste the following sample code into your HTML page.

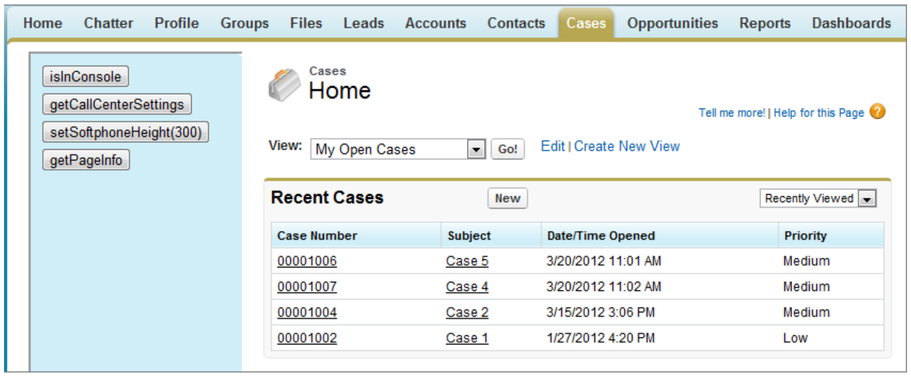
This code demonstrates various functions of Open CTI:

```
<html>
<head>
      <!-- Imports Open CTI JavaScript library. It should point to a valid Salesforce
domain. -->
      <script src="https://domain:port/support/api/25.0/interaction.js"></script>
      <script type="text/javascript">
             // Callback of API method: isInConsole
             var isInConsoleCallback = function (response) {
                  // Returns true if method is executed in Salesforce console, false
otherwise.
                  if (response.result) {
                      alert('SoftPhone is in Salesforce console.');
                  } else {
                      alert('SoftPhone is not in Salesforce console.');
                  }
              };
              // Invokes API method: isInConsole
              function isInConsole() {
                       sforce.interaction.isInConsole(isInConsoleCallback);
              }
              // Callback of API method: getCallCenterSettings
              var getCallCenterSettingsCallback = function (response) {
                     // Result returns call center settings as a JSON string.
                     if (response.result) {
                            alert(response.result);
                     } else {
                            alert('Error retrieving call center settings ' +
response.error);
                     }
               };
               // Invokes API method: getCallCenterSettings
               function getCallCenterSettings() {

sforce.interaction.cti.getCallCenterSettings(getCallCenterSettingsCallback);
               }
               // Callback of API method: setSoftphoneHeight
               var setSoftphoneHeightCallback = function (response) {
                        // Returns true if SoftPhone height was set successfully, false
 otherwise.
               if (response.result) {
                       alert('Setting SoftPhone height to 300px was successful.');
               } else {
                      alert('Setting softphone height failed.');
               }
               };
               // Invokes setSoftphoneHeight API method.
               function setSoftphoneHeight() {
                       sforce.interaction.cti.setSoftphoneHeight(300,
setSoftphoneHeightCallback);
               }
               // Callback of API method: getPageInfo
               var getPageInfoCallback = function (response) {
                      if (response.result) {
                             alert(response.result);
                      } else {
                             alert('Error occured while trying to get page info: ' +
response.error);
                      }
               }
               // Invokes API method getPageInfo
               function getPageInfo() {
                       sforce.interaction.getPageInfo(getPageInfoCallback);
               }
      </script>
</head>
<body>
      <button onclick="isInConsole();">isInConsole</button></br>
      <button onclick="getCallCenterSettings();">getCallCenterSettings</button></br>
      <button onclick="setSoftphoneHeight();">setSoftphoneHeight(300)</button></br>
      <button onclick="getPageInfo();">getPageInfo</button>
</body>
</html>
```

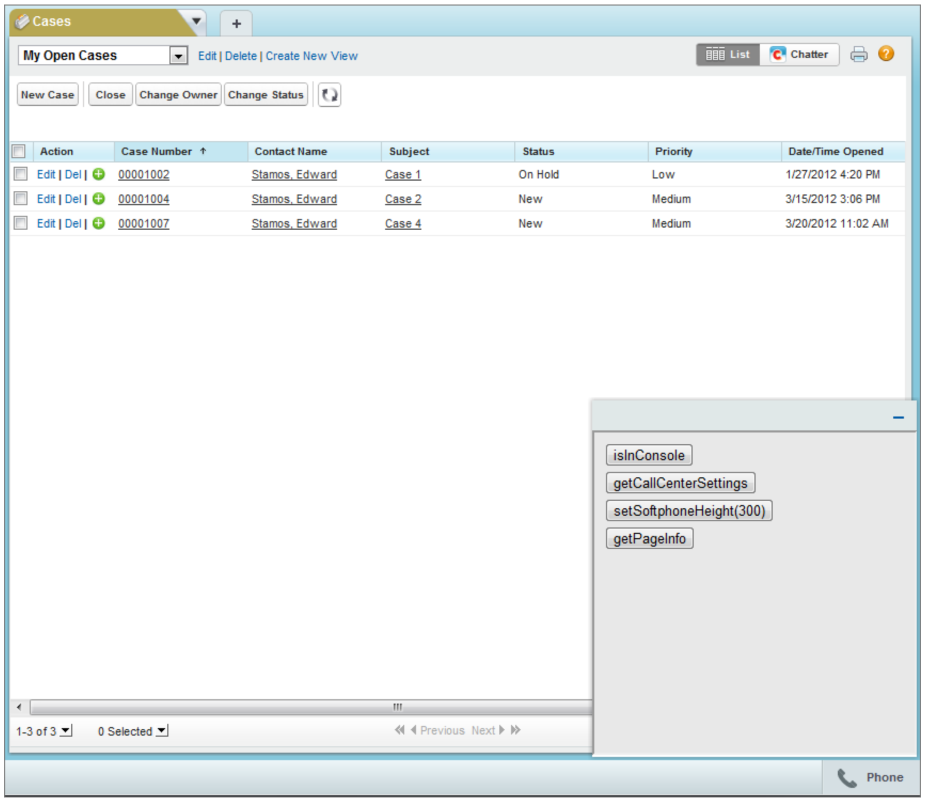
After you create the above HTML page, and add the URL to the call center definition file, the SoftPhone will be rendered on the left in Salesforce, or as custom console component in the Salesforce console:

Output of Sample HTML Page in Salesforce



Case Number	Subject	Date/Time Opened	Priority
00001006	Case 5	3/20/2012 11:01 AM	Medium
00001007	Case 4	3/20/2012 11:02 AM	Medium
00001004	Case 2	3/15/2012 3:06 PM	Medium
00001002	Case 1	1/27/2012 4:20 PM	Low

Output of Sample HTML Page in the Salesforce Console



	Action	Case Number ↑	Contact Name	Subject	Status	Priority	Date/Time Opened
	Edit | Del |	00001002	Stamos, Edward	Case 1	On Hold	Low	1/27/2012 4:20 PM
	Edit | Del |	00001004	Stamos, Edward	Case 2	New	Medium	3/15/2012 3:06 PM
	Edit | Del |	00001007	Stamos, Edward	Case 4	New	Medium	3/20/2012 11:02 AM

CHAPTER 2 Call Center Definition Files

A call center definition file specifies a set of fields and values that are used to define a call center in Salesforce for a particular SoftPhone. Salesforce uses call center definition files in order to support the integration of Salesforce CRM Call Center with multiple CTI system vendors.

A call center in Salesforce CRM Call Center must have a call center definition file that works specifically with a SoftPhone. If you build a custom SoftPhone with Open CTI, you must write a call center definition file to support it. The first instance of a call center for a particular SoftPhone must be defined by importing the adapter's call center definition file into Salesforce. Subsequent call centers can be created by cloning the original call center that was created with the import.

If your organization modifies a SoftPhone or builds a new one, you must customize the SoftPhone's call center definition file so that it includes any additional call center information that is required. For example, if you are building a SoftPhone for a system that supports a backup server, your call center definition file should include fields for the backup server's IP address and port number. SoftPhones for systems that do not make use of a backup server do not need those fields in their associated call center definition files.

Use a text or XML editor to define a call center definition file according to the guidelines in the following topics.

Note: For more information on setting up Salesforce CRM Call Center or importing and cloning call definition files, see "Setting Up Salesforce CRM Call Center" and "Creating a Call Center" in the Salesforce online help.

Call Center Definition File XML Format

A call center definition file consists of three XML elements: `callCenter`, `section`, and `item`. The following list provides details about the properties and attributes of each element:

callCenter

This element represents a definition for a single call center phone system. At least one `<callCenter>` element must be included in every call center definition file. A `<callCenter>` element consists of one or more `<section>` elements.

section

This element represents a grouping of related data fields, such as server information or dialing prefixes. When a call center is edited in Salesforce, fields are organized by the section to which they are assigned. A `<section>` element belongs to a single `<callCenter>` element, and consists of one or more `<item>` elements.

Attributes:

Name	Type	Required?	Description
`sortOrder`	Positive Integer	Required	The order in which the section should appear when the call center is edited in Salesforce. For example, a section with `sortOrder="1"` comes just before a section with `sortOrder="2"`. The values for `sortOrder` must be non-negative integers, and no numbers can be skipped within a single call center definition. For example, if there are three section elements in a call center definition file, one `<section>` element must have `sortOrder="0"`, one `<section>` element must have

Name	Type	Required?	Description
			`sortOrder="1"`, and one `<section>` element must have `sortOrder="2"`.
`name`	String	Required	The internal name of the section as defined in the Salesforce database. You can use this value to refer to the section when writing custom adapter or SoftPhone code. Names must be composed of only alphanumeric characters with no white space or other punctuation. They are limited to 40 characters each. Names beginning with `req` are reserved for required Salesforce sections only (see "Required Call Center Elements and Attributes" in the Salesforce Help). Other reserved words that cannot be used for the `name` attribute include `label`, `sortOrder`, `internalNameLabel`, and `displayNameLabel`.
`label`	String	Optional	The name of the section when viewed in Salesforce. Labels can be composed of any string of UTF-8 characters. They are limited to 1000 characters each.

item

This element represents a single field in a call center definition, such as the IP address of a primary server or the dialing prefix for international calls. When call centers are edited in Salesforce, each `<item>` element is listed under the section to which it belongs. You can have multiple `<item>` elements in a `<section>` element.

Attributes:

Name	Type	Required?	Description
`sortOrder`	Positive Integer	Required	The order in which the item should appear when the call center is edited in Salesforce. For example, an item with `sortOrder="1"` comes just before an item with `sortOrder="2"`. The values for `sortOrder` must be non-negative integers, and no numbers can be skipped within a single call center definition. For example, if there are three item elements in a call center definition file, one `<item>` element must have `sortOrder="0"`, one `<item>` element must have `sortOrder="1"`, and one `<item>` element must have `sortOrder="2"`.
`name`	String	Required	The internal name of the item as defined in the Salesforce database. You can use this value to refer to the item when writing custom adapter or SoftPhone code. Names must be composed of only alphanumeric characters with no white space or other punctuation. They are limited to 40 characters each.

Name	Type	Required?	Description
			Names beginning with `req` are reserved for required Salesforce sections only (see "Required Call Center Elements and Attributes" in the Salesforce Help). Other reserved words that cannot be used for the `name` attribute include `label`, `sortOrder`, `internalNameLabel`, and `displayNameLabel`.
`label`	String	Optional	The name of the item when viewed in Salesforce. Labels can be composed of any string of UTF-8 characters. They are limited to 1,000 characters each.

Required Call Center Elements and Attributes

There must be one `<section>` that includes `<item>` elements with the following names in every call definition file:

`<item>` Name	Description
`reqInternalName`	Represents the unique identifier for the call center in the database. It must have a `sortOrder` value of `0`, and its value must be specified in the call center definition. A value for `reqInternalName` must be composed of no more than 40 alphanumeric characters with no white space or other punctuation. It must start with an alphabetic character and must be unique from the `reqInternalName` of all other call centers defined in your organization.
`reqDisplayName`	Represents the name of the call center as displayed in Salesforce. It must have a `sortOrder` value of `1`. A value for `reqDisplayName` has a maximum length of 1,000 UTF-8 characters.
`reqAdapterUrl`	Represents the location of where the CTI adapter or SoftPhone is hosted. For example, `http://localhost:11000`. Note that relative URLs are allowed for Visualforce pages, for example, `: /apex/softphone`. Also, if you add Force.com Canvas applications to Open CTI, those apps can trump `reqAdapterUrl` when specified.
`reqUseApi`	Represents that the call center is using Open CTI (`true`) or not (`false`).
`reqSoftphoneHeight`	Represents the height of the SoftPhone in pixels as displayed in Salesforce.
`reqSoftphoneWidth`	Represents the width of the SoftPhone in pixels as displayed in Salesforce.
`reqCanvasNamespace`	Represents the namespace associated with any Force.com Canvas applications added to your call center. Required if you add canvas apps to Open CTI.
`reqCanvasApiName`	Represents the API name associated with any Force.com Canvas applications added to your call center. Required if you add canvas apps to Open CTI.

You can add additional `<item>` elements to this section if needed.

Optional Call Center Elements and Attributes

In addition to the required elements for a call definition file, you can add optional elements to configure a SoftPhone.

`<item>` Name	Description
reqStandbyUrl	Represents the location that hosts the secondary SoftPhone. The standby SoftPhone is used after the timeout period for the primary SoftPhone has elapsed and the `notifyInitializationComplete()` method hasn't been called within the required timeout period. When you specify a standby URL, you must also specify the `reqTimeout` field.
reqTimeout	Represents the time in milliseconds after which the standby URL is used to host the SoftPhone. Before the timeout period has elapsed, the SoftPhone displays a loading icon indicating that the SoftPhone is initializing. When you specify a required timeout, you must also specify the `reqStandbyUrl` field.

Specifying Values for <item> Elements

With the exception of the `reqInternalName` `<item>`, whose value must always be specified in a call center definition file, you can specify `<item>` values either in the call center definition file or in Salesforce once the definition file has been imported.

To specify a value for an `<item>` element in a call center definition file, place the value between the opening and closing tags of the `<item>`. For example:

```
<item sortOrder="0" name="reqInternalName" label="Call Center Internal
Label">MyCallCenter</item>
```

sets the value of the `reqInternalName` `<item>` to `MyCallCenter`. Note that any `<item>` value other than the value for `reqInternalName` can be edited in Salesforce after the call center definition is imported.

Sample Call Center Definition File

The following XML code makes up a sample call center definition file:

```
<!--
     All sections and items whose name value begins with "req" are
     required in a valid call center definition file. The sortOrder
     and label attributes can be changed for all required sections
     and items except reqGeneralInfo, reqInternalName, and
     reqDisplayName, in which only the label attribute can be altered.

   Note that the value for the reqInternalName item is limited to
     40 alphanumeric characters and must start with an alphabetic
     character. reqInternalName must be unique for all call centers
     that you define.
-->
<callCenter>
   <section sortOrder="0" name="reqGeneralInfo" label="General Information">
    <item sortOrder="0" name="reqInternalName" label="InternalNameAAA">DemoAdapter</item>

    <item sortOrder="1" name="reqDisplayName" label="Display Name">Demo Call Center
Adapter</item>
    <item sortOrder="2" name="reqAdapterUrl" label="CTI Adapter
URL">https://c.force.com/softphone</item>
```

```
     <item sortOrder="3" name="reqUseApi" label="Use CTI API">true</item>
     <item sortOrder="4" name="reqSoftphoneHeight" label="Softphone Height">300</item>
     <item sortOrder="5" name="reqSoftphoneWidth" label="Softphone Width">500</item>
     <item sortOrder="6" name="reqCanvasNamespace" label="Canvas Namespace">mm</item>
    <item sortOrder="7" name="reqCanvasApiName" label="Canvas API Name">Hello_World</item>

   </section>
   <section sortOrder="1" name="reqDialingOptions" label="Dialing Options">
     <item sortOrder="0" name="reqOutsidePrefix" label="Outside Prefix">9</item>
     <item sortOrder="1" name="reqLongDistPrefix" label="Long Distance Prefix">1</item>
    <item sortOrder="2" name="reqInternationalPrefix" label="International Prefix">01</item>

   </section>
</callCenter>
```

CHAPTER 3 Working with Open CTI

Use Open CTI to do the following in Salesforce:

- Set the height or width of a SoftPhone
- Enable or disable click-to-dial
- Return a call center definition file's settings
- Determine if a user is in the Salesforce console
- Show or hide a SoftPhone in the Salesforce console
- Return information about a page
- Execute an Apex method from an Apex class that's exposed in Salesforce
- Save or update an object in Salesforce
- Search keywords in Salesforce and screen pop any matching records as defined in a SoftPhone layout

Connecting to Open CTI

The first portion of any JavaScript code that uses the Open CTI 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 Open CTI file:

  ```
  <apex:page>
          <script src="/support/api/36.0/interaction.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/api/36.0/interaction.js"
  type="text/javascript"></script>
  ```

  For third-party domains, it is necessary to specify an absolute URL to `interaction.js` to use the toolkit. The default instance at which you can access the toolkit library is: `https://c.na1.visual.force.com/support/api/36.0/interaction.js`. We recommend that you use the default instance when the organization's instance cannot be determined.

The version of Open CTI is in the URL.

Asynchronous Calls with Open CTI

Open CTI 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, 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:

```
//Set SoftPhone height
   sforce.interaction.cti.setSoftphoneHeight(300, callback);
```

Note: The call result depends on the execution context. For example, calling `setSoftphoneWidth()` in the standard Salesforce application has no effect, but calling `setSoftphoneWidth()` in the Salesforce console resizes the width of the SoftPhone.

Working with Force.com Canvas

To integrate Open CTI 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 Open CTI 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 Salesforce 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 Open CTI, do the following:

1. Include the Open CTI API in `index.jsp`.
2. 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 Open CTI and canvas apps:

- The Open CTI 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

- Since many of the methods in Open CTI are asynchronous and return their results using a callback method, Salesforce recommends that you refer to the documentation for each method to understand the information for each response.
- Errors generated by Open CTI are typically emitted in a way that doesn't halt JavaScript processing. Therefore, Salesforce recommends you use a tool such as Firebug for Firefox to monitor the JavaScript console and to help you debug your code.
- For information on customizing, extending, or integrating the sidebars of the Salesforce console using Visualforce, see "Console Components" in the Salesforce online help.

CHAPTER 4 Methods for Salesforce Application Interaction

Open CTI lets your CTI system interact with the Salesforce application.

You can use the following methods to set interactions between a CTI system and Salesforce, or between elements on a Case Feed page:

CTI Methods

Method	Description
`getPageInfo()`	Returns information about the current page as a JSON string.
`isInConsole()`	Indicates if the SoftPhone is in the Salesforce console. For more information, see “Salesforce Console” in the Salesforce online help.
`isVisible()`	Returns `true` if the SoftPhone is visible or `false` if the SoftPhone is hidden.
`notifyInitializationComplete()`	Notifies Salesforce that the SoftPhone initialization is complete and that Salesforce should not switch to a standby URL. While the SoftPhone initializes, a loading icon displays in the SoftPhone area.
`onFocus()`	Registers a function to call when the browser focus changes. In the Salesforce console, the browser focus changes when a user navigates between primary tabs or the navigation tab.
`refreshPage()`	Returns `true` if page refresh is invoked, `false` otherwise. When this method is called within the Salesforce console, it refreshes the current active tab.
`refreshRelatedList()`	Returns `true` if the related list with the given `listName` is refreshed, `false` otherwise. When this method is called within the Salesforce console, only the related list with the given list name in the currently focused view will be refreshed.
`reloadFrame()`	Reloads the frame that contains the page making the call.
`runApex()`	Executes an Apex method from an Apex class that’s exposed in Salesforce.
`saveLog()`	Saves or updates an object in Salesforce.
`screenPop()`	Pops to a target URL, which must be relative.
`searchAndGetScreenPopUrl()`	Searches objects specified in the SoftPhone layout for a given string. Returns search results and the relative URL to be screen popped. Note that this method does not perform an actual screen pop. This method respects screen pop settings defined in the SoftPhone layout. For more information, see “Designing a Custom SoftPhone Layout” in the Salesforce online help.
`searchAndScreenPop()`	Searches objects specified in the SoftPhone layout for a given string. Returns search results and screen pops any matching records. This method respects screen pop settings defined in the SoftPhone layout.
`setVisible()`	Shows or hides the SoftPhone in the Salesforce console. For more information, see “Salesforce Console” in the Salesforce online help.

CTI Methods	
Case Feed Methods	
onObjectUpdate()	Registers a function to call when case fields, the case feed, or case-related list data has changed on a Case Feed page.
refreshObject()	Notifies the Case Feed page that case fields, the case feed, or case-related list data has changed, and forces an update of these on the page.

getPageInfo()

Usage

Returns information about the current page as a JSON string.

Syntax

```
sforce.interaction.getPageInfo(callback:function);
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
                  alert(response.result);
            } else {
                   alert(response.error);
           }
        };
       function getPageInfo() {
                //Invokes API method
                sforce.interaction.getPageInfo(callback);
        }
</script>
</head>
<body>
```

```
        <button onclick="getPageInfo();">getPageInfo</button>
</body>
</html>
```

Response

Name	Type	Description
`result`	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: `{"url":"http://na1.salesforce.com/001x0000003DGQR", "objectId":"001x0000003DGQR","objectName":"Acme", "object":"Account","displayName":"Company"}` For API version 31.0 and later, invoking this API method on a PersonAccount object returns the following additional information. • accountId or contactId, the associated account or contact ID • personAccount, which is `true` if the object is a PersonAccount and `false` otherwise For example: `{"url":"http://na1.salesforce.com/001x0000003DGQR", "objectId":"001x0000003DGQR","objectName":"Acme Person Account", "object":"Account", "contactId":"003D000000QOMqg", "personAccount":true}`
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

isInConsole()

Usage

Indicates if the SoftPhone is in the Salesforce console. For more information, see "Salesforce Console" in the Salesforce online help.

Note: If this method is used in a Salesforce console where multi-monitor components is turned on, any popped out SoftPhone components are indicated as in the console. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the online help.

Syntax

```
sforce.interaction.isInConsole(callback:function)
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert('User is in console.');
           }
           else {
              alert('User is not in console.');
           }
        };
</script>
</head>
<body>
       <button onclick="sforce.interaction.isInConsole(callback);">isInConsole</button>
</body>
</html>
```

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
`result`	boolean	`true` if the SoftPhone was in the Salesforce console, `false` if the SoftPhone wasn't in the Salesforce console.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`isVisible()`

Usage

Returns `true` if the SoftPhone is visible or `false` if the SoftPhone is hidden.

Syntax

```
sforce.interaction.isVisible(callback:function)
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert('SoftPhone is visible');
           } else {
              alert('SoftPhone is not visible');
           }
        };
   function isVisible() {
           sforce.interaction.isVisible(callback);
   }
</script>
</head>
<body>
       <button onclick="isVisible();">isVisible</button>
</body>
</html>
```

Response

Name	Type	Description
`result`	boolean	`true` if the SoftPhone is visible, `false` if the SoftPhone isn't visible.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`notifyInitializationComplete()`

Usage

Notifies Salesforce that the SoftPhone initialization is complete and that Salesforce should not switch to a standby URL. While the SoftPhone initializes, a loading icon displays in the SoftPhone area. To use a standby URL, you must specify it in the call center's definition file. For more information, see Optional Call Center Elements and Attributes on page 10.

Syntax

```
sforce.interaction.cti.notifyInitializationComplete()
```

Arguments

None.

Sample Code

```
<html>
<head>
<script
src="https://na1-blitz04.soma.salesforce.com/support/api/29.0/interaction.js"></script>
<script type="text/javascript">
        // Invokes API method
sforce.interaction.cti.notifyInitializationComplete();
</script>
</head>
<body>
The interaction framework has been notified that the cti initialization is complete.
</body>
</html>
```

Response

None.

`onFocus()`

Usage

Registers a function to call when the browser focus changes. In the Salesforce console, the browser focus changes when a user navigates between primary tabs or the navigation tab. See "Salesforce Console" in the Salesforce online help for more information.

Syntax

```
sforce.interaction.onFocus( listener:function );
```

Arguments

Name	Type	Description
`listener`	function	JavaScript method called when the browser focus changes.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
                  alert(response.result);
           }
        };
       //Invokes API method
                sforce.interaction.onFocus(callback);
</script>
</head>
</body>
</html>
```

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
`result`	string	Returns the URL of the page in focus 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: `{"url":"http://salesforce.com/001x0000003DGQR", "objectId":"001x0000003DGQR","objectName":"Acme", "object":"Account","displayName":"Company"}` If the page isn't focused on an object, the object ID, object name, and object will be empty. For API version 31.0 and later, invoking this API method on a PersonAccount object returns the following additional information. • accountId or contactId, the associated account or contact ID

Name	Type	Description
		• personAccount, which is `true` if the object is a PersonAccount and `false` otherwise For example: `{"url":"http://na1.salesforce.com/001x0000003DGQR", "objectId":"001x0000003DGQR","objectName":"Acme Person Account", "object":"Account", "contactId":"003D000000QOMqg", "personAccount":true}`
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

refreshPage()

Usage

Returns `true` if page refresh is invoked, `false` otherwise. When this method is called within the Salesforce console, it refreshes the current active tab. This method is only available in API version 28.0 or later.

Syntax

```
sforce.interaction.refreshPage(callback:function);
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/28.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
                  alert('Page refresh has been invoked.');
            } else {
                   alert('Page refresh has not been invoked.');
           }
        };
        function refreshPage() {
                  sforce.interaction.refreshPage(callback);
         }
</script>
</head>
<body>
        <button onclick="refreshPage();">refreshPage</button>
</body>
</html>
```

Response

Name	Type	Description
`result`	boolean	Returns `true` if page refresh has been invoked, `false` otherwise.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

refreshRelatedList()

Usage

Returns `true` if the related list with the given `listName` is refreshed, `false` otherwise. When this method is called within the Salesforce console, only the related list with the given list name in the currently focused view will be refreshed. This method is only available in API version 28.0 or later.

Syntax

```
sforce.interaction.refreshRelatedList(listName:string, callback:function)
```

Arguments

Name	Type	Description
`listName`	string	The name of the related list to refresh. For example, Contact for Contacts related list or Activity for Open Activities related list. Note that to refresh a custom related list created from a custom lookup field, `listName` must specify the ID of the custom lookup field.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/28.0/interaction.js"></script>
   <script type="text/javascript">
       function checkRefreshResult(result) {
           if (result.result) {
                  alert('The related list is refreshed!');
            } else {
                   alert('Cannot refresh the related list with the given listName! Make
sure the listName is correct and the related list is on the page.');
           }
        }
       function refreshActivityRelatedList() {
                 sforce.interaction.refreshRelatedList('Activity', checkRefreshResult);
        }

       function refreshHistoryRelatedList() {
                 sforce.interaction.refreshRelatedList('History', checkRefreshResult);
        }

       function saveAndRefresh() {
                 sforce.interaction.saveLog('Task',
'Subject=ImportantTask&WhatId=[15-character ID of an account to which you want to attach
the task]', function(result) {
           if (result.result) {
                  refreshActivityRelatedList();
            } else {
                  alert('Could not save the object! Check the developer console for error
 messages.');
            }
      });
}
</script>
</head>
<body>
      <button onclick="refreshHistoryRelatedList();">Refresh History Related List</button>

       <button onclick="saveAndRefresh();">Save and Refresh</button>
</body>
</html>
```

Response

Name	Type	Description
`result`	boolean	Returns `true` if the related list with the given name is refreshed, `false` otherwise.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

Notes

- This method cannot refresh related lists created from `<apex:relatedList>`.
- This method cannot refresh a related list from an overridden Visualforce page in the Salesforce console.
- If called from within the Salesforce console, this method will only search for the related list to refresh in the currently focused view.

`reloadFrame()`

Reloads the frame that contains the page making the call. This method is available in API version 34.0 or later.

Syntax

```
sforce.interaction.entityFeed.reloadFrame()
```

Arguments

None.

Sample Code–JavaScript

```
<apex:page standardController="Case">
    <apex:includeScript value="/support/api.34.0/interaction.js"/>
    <a href"javascript:void(0); onclick="sforce.interaction.entityFeed.reloadFrame();">
    Reload</a>
</apex:page>
```

Response

None.

`runApex()`

Usage

Executes an Apex method from an Apex class that's exposed in Salesforce. For more information, see "Apex Code Overview" in the Salesforce online help.

Syntax

```
sforce.interaction.runApex(apexClass:string, methodName:string, methodParams:string,
(optional) callback:function)
```

Arguments

Name	Type	Description
`apexClass`	string	Specifies the Apex class of the method to execute.
`methodName`	string	Specifies the method to execute.
`methodParams`	string	Specifies the method parameters to pass. The string must include field value pairs and be formatted as a valid query string. For example:`name=acme&phone=(212) 555-5555`.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

1. An administrator creates an Apex class and Apex method (see the Salesforce online help for more information):

```
global class AccountRetrieval{

webService static String getAccount(String name) {
   List<Account> accounts = new List<Account>();
   for (Account account : Database.query('Select Id, Name, phone from Account where Name
 like\"' + name + '%\"')){
       accounts.add(account);
   }
       String JSONString = JSON.serialize(accounts);
       return JSONString;
   }
  }
```

2. In the location where you've created the Apex class and method in Salesforce, click **Generate WSDL** to expose the method and class so that a third-party SoftPhone can call it.

3. Add your code to the SoftPhone:

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
                  alert(response.result);
            } else {
                   alert(response.error);
           }
        };
       function runApex() {
                //Invokes API method
             sforce.interaction.runApex('AccountRetrieval', 'getAccount', 'name=acme',
 callback);
        }
</script>
```

```
</head>
<body>
       <button onclick="runApex();">runApex</button>
</body>
</html>
```

4. Output is returned. In this example, one account named, Acme, was found:

```
[{"attributes":{"type":"Account",
"url":"/services/data/v25.0/sobjects/Account/001x0000003DGQRAA4"},
"Id":"001x0000003DGQRAA4","Name":"Acme","Phone":"(212) 555-5555"}]
```

Response

Name	Type	Description
`result`	string	Returns the result from executing the method from the specified Apex class. No specific format is returned. The format is determined by the value from the method that was executed.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`saveLog()`

Usage

Saves or updates an object in Salesforce.

Syntax

```
sforce.interaction.saveLog(object:string, saveParams:string, (optional)callback:function)
```

Arguments

Name	Type	Description
`object`	string	The name of the object to save or update.
`saveParams`	string	Specifies the fields to save or update on the object. If the object's ID is specified, a record is updated. For example: `Id=001D000000J6qIX&Name=Acme&Phone=4154561515`. If the object's ID isn't specified, a new record is created. For example: `Name=Acme&Phone=4154561515`.

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
                  alert(response.result);
           } else {
                  alert(response.error);
           }
       }
       function saveLog() {
           //Invokes API method
              sforce.interaction.saveLog('Account','Name=NewAccountName&Phone=4155551212',
 callback);
       }
</script>
</head>
   <button onclick="saveLog();">saveLog</button>
</html>
```

Response

Name	Type	Description
`result`	boolean	`true` if saving or updating the object was successful, `false` if saving or updating the object wasn't successful.
`id`	string	The Id of the newly created object.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`screenPop()`

Usage

Pops to a target URL, which must be relative.

Syntax

```
sforce.interaction.screenPop(url:string, force:boolean, (optional) callback:function)
```

Arguments

Name	Type	Description
`url`	string	A relative URL, which specifies the location of the screen pop.
`force`	boolean	Set value to `true` to force a screen pop, `false` otherwise. This argument is only available in API version 28.0 and later.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/28.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert('Screen pop was set successfully.');
           }
           else {
              alert('Screen pop failed.' + result.error);
           }
        };
       function screenPop() {
                //Invokes API method
                sforce.interaction.screenPop('/001x0000003DGQR', true, callback);
        }
</script>
</head>
<body>
       <!-- Note that '001x0000003DGQR' is an example of an object Id to screen pop. -->
       <button onclick="screenPop();">screen pop to entity Id</button>
</body>
</html>
```

Response

Name	Type	Description
`result`	boolean	`true` if the screen pop was successful, `false` if the screen pop wasn't successful.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

searchAndGetScreenPopUrl()

Usage

Searches objects specified in the SoftPhone layout for a given string. Returns search results and the relative URL to be screen popped. Note that this method does not perform an actual screen pop. This method respects screen pop settings defined in the SoftPhone layout. For more information, see “Designing a Custom SoftPhone Layout” in the Salesforce online help. This method is only available in API version 28.0 or later.

Syntax

```
sforce.interaction.searchAndGetScreenPopUrl(searchParams:string, queryParams:string,
callType:string, callback:function)
```

Arguments

Name	Type	Description
`searchParams`	string	String to search.
`queryParams`	string	Specifies the query parameters to pass to the URL.
`callType`	string	Specifies the type of call, such as inbound, outbound, internal, or null. Per the settings in the SoftPhone layout, the call type determines which objects to search for any matches. For more information, see “Designing a Custom SoftPhone Layout” in the Salesforce online help. If `callType` is null, searches are inbound by default. If `callType` is internal or outbound, no screen pops occur.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/33.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert(response.result);
           } else {
              alert(response.error);
           }
        };
       function searchAndGetScreenPopUrl() {
                //Invokes API method
                sforce.interaction.searchAndGetScreenPopUrl('Acme',
'Key1=value1&Key2=value2', 'inbound', callback);
        }
</script>
</head>
<body>
       <button onclick="searchAndGetScreenPopUrl();">searchAndGetScreenPopUrl</button>
</body>
</html>
```

Response

Name	Type	Description

`result`	string	Returns a list of objects that match the search results and the URL to the screen pop (`screenPopUrl`). The search is performed on the objects specified in the SoftPhone layout. For each object found, the object ID, field names, field values, and for API version 33.0 or later, object tab name are returned as a JSON string.
		The following is an example of searching for "Acme," and finding one Account and three Opportunity objects:
		`{"006x0000001ZcyG":{"Name":"Acme - 600 Widgets","object":"Opportunity","displayName":"Opportunity"}, "001x0000003DGQR":{"Name":"Acme","Type":"Analyst","object":"Account", "displayName":"Company"}, "006x0000001ZcyH":{"Name":"Acme - 200 Widgets","object":"Opportunity","displayName":"Opportunity"}, "006x0000001ZcyF":{"Name":"Acme - 1,200 Widgets","object":"Opportunity","displayName":"Opportunity"}, screenPopUrl:"/search/SearchResults?searchType=2&str=Acme"}`
		For API version 31.0 and later, invoking this API method on a PersonAccount object returns additional information:
		`{"001D000000Jn5C5":{"Name":"PersonAccount","contactId":"003D000000QQCEu", "Type":"Analyst","object":"Account","displayName":"Account","personAccount":true}, "screenPopUrl":"/001D000000Jn5C5"}`

`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`searchAndScreenPop()`

Usage

Searches objects specified in the SoftPhone layout for a given string. Returns search results and screen pops any matching records. This method respects screen pop settings defined in the SoftPhone layout. For more information, see "Designing a Custom SoftPhone Layout" in the Salesforce online help.

Syntax

```
sforce.interaction.searchAndScreenPop(searchParams:string, queryParams:string,
callType:string, (optional) callback:function);
```

Arguments

Name	Type	Description
`searchParams`	string	String to search.
`queryParams`	string	Specifies the query parameters to pass to the URL.
`callType`	string	Specifies the type of call, such as inbound, outbound, internal, or null. Per the settings in the SoftPhone layout, the call type determines which objects to search for any matches. For more information, see “Designing a Custom SoftPhone Layout” in the Salesforce online help. If `callType` is null, searches are inbound by default. If `callType` is internal or outbound, no screen pops occur.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/33.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
                  alert(response.result);
            } else {
                   alert(response.error);
           }
        };
       function searchAndScreenPop() {
                //Invokes API method
                sforce.interaction.searchAndScreenPop('Acme', 'Key1=value1&Key2=value2',
'inbound', callback);
        }
</script>
</head>
<body>
       <button onclick="searchAndScreenPop();">searchAndScreenPop</button>
</body>
</html>
```

Response

Name	Type	Description

`result`	string	Returns a list of objects that match the search results. The search is performed on the objects specified in the SoftPhone layout. For each object found, the object ID, field names, field values, and for API version 33.0 or later, object tab names are returned as a JSON string.
		The following is an example of searching for “Acme,” and finding one Account and three Opportunity objects:
		`{"006x0000001ZcyG":{"Name":"Acme - 600 Widgets","object":"Opportunity","displayName":"Opportunity"}, "001x0000003DGQR":{"Name":"Acme","Type":"Analyst","object":"Account", "displayName":"Company"}, "006x0000001ZcyH":{"Name":"Acme - 200 Widgets","object":"Opportunity","displayName":"Opportunity"}, "006x0000001ZcyF":{"Name":"Acme - 1,200 Widgets","object":"Opportunity","displayName":"Opportunity"}, screenPopUrl:"/search/SearchResults?searchType=2&str=Acme"}`
		For API version 31.0 and later, invoking this API method on a PersonAccount object returns additional information:
		`{"001D000000JWAW8":{"Name":"Acme","contactId":"003D000000QNwDB", "Type":"Analyst","object":"Account","personAccount":true}}`

`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`refreshObject()`

Usage

Notifies the Case Feed page that case fields, the case feed, or case-related list data has changed, and forces an update of these on the page.

Note: Use this method with Visualforce pages you want to use as custom publishers in Case Feed.

Syntax

```
sforce.interaction.entityFeed.refreshObject(
      objectId:string,
      refreshFields:boolean,
      refreshRelatedLists:boolean,
      refreshFeed:boolean,callback:function)
```

Arguments

Name	Type	Description
`objectId`	string	The record ID of the case object.
`refreshFields`	boolean	Indicates that one or more fields on the case have changed.
`refreshRelatedLists`	boolean	Indicates that one or more case-related lists have changed.
`refreshFeed`	boolean	Indicates that the case feed has changed.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<apex:page standardController="Case">
    <apex:includeScript value="/support/api/26.0/interaction.js"/>
    <a href="javascript:void(0);"
onclick="sforce.interaction.entityFeed.refreshObject('{!case.id}', true, true, true,
function(response) {alert('Case was updated: ' + response.result);});">Refresh Case</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 fields:

Name	Type	Description
`result`	boolean	`true` if the Case Feed page was successfully updated, `false` if it was not.

setVisible()

Usage

Shows or hides the SoftPhone in the Salesforce console. For more information, see "Salesforce Console" in the Salesforce online help.

Note: If this method is used in a Salesforce console where multi-monitor components is turned on, an error will be returned. For more information, see "Turn on Multi-Monitor Components for a Salesforce Console" in the online help.

Syntax

```
sforce.interaction.setVisible(value:boolean, (optional) callback:function)
```

Arguments

Name	Type	Description
`value`	boolean	Set value to `true` to show the SoftPhone or set value to `false` to hide the SoftPhone.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert(response.result);
           } else {
              alert(response.error);
           }
        };
   function setVisible(value) {
           sforce.interaction.setVisible(value, callback);
   }
</script>
</head>
<body>
       <button onclick="setVisible(false);">hide softphone</button>
</body>
</html>
```

Response

Name	Type	Description
`result`	boolean	`true` if showing or hiding the SoftPhone succeeded, `false` if showing or hiding the SoftPhone didn't succeed.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`onObjectUpdate()`

Usage

Registers a function to call when case fields, the case feed, or case-related list data has changed on a Case Feed page.

Note: Use this method with Visualforce pages you want to use as custom publishers in Case Feed.

Syntax

```
sforce.interaction.entityFeed.onObjectUpdate(callback:function)
```

Arguments

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

Sample Code–JavaScript

```
<apex:page standardController="Case">
    <apex:includeScript value="/support/api/26.0/interaction.js"/>
    <script type="text/javascript">
        var callback = function(response) {
            alert('Case was updated. Fields = ' + response.fieldsUpdated +
            ' Related lists = ' + response.relatedListsUpdated + ' Feed = ' +
              response.feedUpdated);
        };
        //Invokes API method
        sforce.interaction.entityFeed.onObjectUpdate(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
`fieldsUpdated`	boolean	`true` if one or more case fields were updated.
`relatedListsUpdated`	boolean	`true` if one or more case related lists were updated.
`feedUpdated`	boolean	`true` if the case feed was updated.

CHAPTER 5 Methods for Computer-Telephony Integration (CTI)

Open CTI lets you integrate your CTI system with Salesforce. For more information about CTI, see “Call Center Overview” in the Salesforce online help.

Use the following methods to integrate a CTI system with Salesforce:

Method	Description
`disableClickToDial()`	Disables click-to-dial.
`enableClickToDial()`	Enables click-to-dial.
`getCallCenterSettings()`	Returns the call center settings in the call center definition file as a JSON string.
`getDirectoryNumbers()`	Returns the list of phone numbers from the call center’s directory.
`getSoftphoneLayout()`	Returns the SoftPhone layout as a JSON string. For more information on SoftPhone layouts, see “Designing a Custom SoftPhone Layout” in the Salesforce online help.
`onClickToDial()`	Registers a function to call when a user clicks an enabled phone number.
`setSoftphoneHeight()`	Sets the SoftPhone height in pixels.
`setSoftphoneWidth()`	Sets the SoftPhone width in pixels for the Salesforce console. For more information, see “Salesforce Console” in the Salesforce online help.

`disableClickToDial()`

Usage

Disables click-to-dial.

Syntax

```
sforce.interaction.cti.disableClickToDial( (optional) callback:function )
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert('Click to dial was disabled.');
           } else {
              alert('Click to dial was not disabled.');
           }
        };
        function disableClickToDial() {
   //Invokes API method
   sforce.interaction.cti.disableClickToDial(callback);
   }
</script>
</head>
<body>
   <button onclick="disableClickToDial();">disable click to dial</button>
</body>
</html>
```

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
`result`	boolean	`true` if click-to-dial was disabled, `false` if click-to-dial wasn't disabled.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`enableClickToDial()`

Usage

Enables click-to-dial.

Syntax

```
sforce.interaction.cti.enableClickToDial( (optional) callback:function )
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert('Click to dial was enabled.');
           } else {
              alert('Click to dial was not enabled.');
           }
        };
        function enableClickToDial() {
   //Invokes API method
   sforce.interaction.cti.enableClickToDial(callback);
   }
</script>
</head>
<body>
   <button onclick="enableClickToDial();">enable click to dial</button>
</body>
</html>
```

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
`result`	boolean	`true` if click-to-dial was enabled, `false` if click-to-dial wasn't enabled.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`getCallCenterSettings()`

Usage

Returns the call center settings in the call center definition file as a JSON string. For more information, see Call Center Definition Files on page 8.

Syntax

```
sforce.interaction.cti.getCallCenterSettings(callback:function)
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
       alert(response.result);
   }

   //Calls getCallCenterSettings
   sforce.interaction.cti.getCallCenterSettings(callback);
   </script>
</head>
<body></body>
</html>
```

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
`result`	string	If the API call was successful, the call center settings definition is returned as a JSON string. If the API call failed, null is returned.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

getDirectoryNumbers()

Usage

Returns the list of phone numbers from the call center's directory. This method is only available in API version 31.0 or later.

Syntax

```
sforce.interaction.cti.getDirectoryNumbers(isGlobal:boolean, callCenterName:String,
(optional) callback:function, (optional) resultSetPage:Integer, (optional)
resultSetPageSize:Integer)
```

Arguments

Name	Type	Description
`isGlobal`	boolean	Set the value to `true` to return a directory number from the global call center name, or set the value to `false` to return a directory number that is specific to a call center.
`callCenterName`	string	Specifies the call center name on which to return directory numbers. If `isGlobal` is set to `false`, and this field is not specified, all directory numbers are returned.
`callback`	function	JavaScript method called upon completion of the method.
`resultSetPage`	integer	Represents the page number of the list of results to return. This number starts at 0.
`resultSetPageSize`	integer	Sets the maximum number of phone numbers to retrieve, which is defaulted to 5000 and has a maximum number of 10000. If `hasNext` returns `true` in the `callback`, use this argument with `resultSetPage` to get the next page of results. For example, if `resultSetPageSize` is set to 5000, and `resultSetPage` is set to 0, the first 5000 phone numbers are returned. If `resultSetPage` is set to 1, the next 5000 phone numbers are returned.

Sample Code–JavaScript

```
<html>
<head>
   <script src="https://domain:port/support/api/31.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
                  alert(response.result);
            } else {
                   alert(response.error);
           }
        };
       var isGlobal = false; //Do not return directories from the global call center
       var callCenterName = 'My Call Center'; //Call center name of directory numbers to
return

       function getDirectoryNumbers() {
                sforce.interaction.cti.getDirectoryNumbers(isGlobal, callCenterName,
callback);
        }
</script>
</head>
<body>
       <button onclick="getDirectoryNumbers();">Get Directory Numbers</button>
</body>
</html>
```

Response

Name	Type	Description
`result`	string	Returns a JSON string that represents the list of phone numbers from the specified call center name. Each phone number element contains a call center name, phone, and description. For example: `{ directoryNumbers: [ {callCenterName:'Demo Call Center', name:'Sales Cloud', phone:'415-555-1212', description:'Sales Cloud additional number'}, {callCenterName:'Demo Call Center 2', name:'Service Cloud', phone:'415-555-3434', description:'Service Cloud additional number'}, ], hasNext: false }`
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`getSoftphoneLayout()`

Usage

Returns the SoftPhone layout as a JSON string. For more information on SoftPhone layouts, see "Designing a Custom SoftPhone Layout" in the Salesforce online help. This method is only available in API version 27.0 or later.

Syntax

```
sforce.interaction.cti.getSoftphoneLayout(callback:function);
```

Arguments

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

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/27.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
               alert(response.result);
           }
       // Calls getSoftphoneLayout
       sforce.interaction.cti.getSoftphoneLayout(callback);
</script>
</head>
<body></body>
</html>
```

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
`result`	string	If the API call was successful, the SoftPhone layout definition is returned as a JSON string. If the API call failed, null is returned. The returned JSON string contains three elements that represent each of the call types: • `"Internal"` • `"Inbound"` • `"Outbound"` Each call-type contains three subsections: • `"callRelatedFields"`—An array of call-related fields selected to display. Possible values are `"ANI"`, `"DNIS"`, `"SEGMENT"`, and `"QUEUE"`. • `"objects"`—The set of Salesforce objects selected to display, along with the `Field Label` and `Field Name` (API name) selected to display from each object. • `"screenPopSettings"`—This object contains a `"screenPopsOpenWithin"` field with a value of either `"ExistingWindow"` or `"NewWindow"`. Additionally, it contains the

Name	Type	Description
		settings for each of the screen pop match types: `"NoMatch"`, `"SingleMatch"`, `"MultipleMatches"`. Each match type contains a corresponding `"screenPopType"` field and may also contain a `"screenPopData"` field. If `"screenPopType"` has a value of `"PopToEntity"`, then `"screenPopData"` contains the name of the target object. If `"screenPopType"` has a value of `"PopToVisualforce"`, then `"screenPopData"` contains the name of the target Visualforcepage. If `"screenPopType"` has a value of `"PopToSearch"`, then there won't be a `"screenPopData"` field.

The following is an example of a JSON response:

```
"Internal" : {
   "callRelatedFields" : [
    "ANI",
    "DNIS",
   ]
   "objects" : {
    "User" : [ {
      "displayName" : "Name",
      "apiName" : "Name"
     }
    ]
   },
   "screenPopSettings" : {}
  },
  "Inbound" : {
   "callRelatedFields" : [
    "ANI",
    "DNIS",
    "SEGMENT",
    "QUEUE"
   ],
    "objects" : {
    "Account" : [ {
      "displayName" : "Account Name",
      "apiName" : "Name"
     }
    ]

   },
   "screenPopSettings" : {
    "NoMatch" : {
     "screenPopType" : "PopToEntity",
     "screenPopData" : "Contact"
    },

    "SingleMatch" : {
     "screenPopType" : "PopToVisualforce",
     "screenPopData" : "Visualforce_Page_Name"
    },
```

Name	Type	Description
		`"MultipleMatches" : { "screenPopType" : "PopToSearch" } } }, "Outbound" : { "callRelatedFields" : [ "DNIS" ], "objects" : { "Account" : [ { "displayName" : "Account Name", "apiName" : "Name" } ] }, "screenPopSettings" : {} } }`
`error`	string or undefined	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

onClickToDial()

Usage

Registers a function to call when a user clicks an enabled phone number.

Syntax

```
sforce.interaction.cti.onClickToDial( listener:function )
```

Arguments

Name	Type	Description
`listener`	function	JavaScript method called when the user clicks a phone number.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
```

```
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var listener = function (response) {
           if (response.result) {
              alert('User clicked on a phone number.' + response.result );
           }
        };
   //Invokes API method
   sforce.interaction.cti.onClickToDial(listener);
</script>
</head>
</html>
```

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
`result`	string	Returns the phone number, object ID, the name of the object, and for API version 33.0 or later, the object tab name from where the click was initiated as a JSON string. For example: `{"number":"4155551212","objectId":"001x0000003DIGj","objectName":"Account", "displayName":"Company"}` For API version 33.0 or later, invoking this API method on a PersonAccount object returns the following additional information. • accountId or contactId, the associated account or contact ID • personAccount, which is `true` if the object is a PersonAccount and `false` otherwise. For example: `{"number":"4155551212","object Id":"001D000000JWVvP","objectName":"Howard Jones","object":"Account", "personAccount":true,"contactId":" 003D000000QOBPX"}`
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`setSoftphoneHeight()`

Usage

Sets the SoftPhone height in pixels.

Note: If this method is used in a Salesforce console where multi-monitor components is turned on, an error will be returned because resizing multi-monitor component is not allowed. For more information, see “Turn on Multi-Monitor Components for a Salesforce Console” in the online help.

Syntax

```
sforce.interaction.cti.setSoftphoneHeight(height:number, (optional) callback:function)
```

Arguments

Name	Type	Description
`height`	number	SoftPhone height in pixels. The height should be a number that’s equal or greater than 0.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert('Height was set successfully.');
           }
           else {
              alert('Height was not set successfully.');
           }
        };
</script>
</head>
<body>
       <button onclick="sforce.interaction.cti.setSoftphoneHeight(200, callback);">
       set softphone height to 200px
       </button>
</body>
</html>
```

Response

Name	Type	Description
`result`	boolean	`true` if the height was set successfully, `false` if setting the height wasn’t successful.

Name	Type	Description
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

`setSoftphoneWidth()`

Usage

Sets the SoftPhone width in pixels for the Salesforce console. For more information, see “Salesforce Console” in the Salesforce online help.

Note: If this method is used in a Salesforce console where multi-monitor components is turned on, an error will be returned because resizing multi-monitor component is not allowed. For more information, see “Turn on Multi-Monitor Components for a Salesforce Console” in the online help.

Syntax

```
sforce.interaction.cti.setSoftphoneWidth(width:number, (optional) callback:function)
```

Arguments

Name	Type	Description
`width`	number	SoftPhone width in pixels. The width should be a number that’s equal or greater than 0.
`callback`	function	JavaScript method called upon completion of the method.

Sample Code–JavaScript

```
<html>
<head>
   <script type="text/javascript"
src="http://domain:port/support/api/25.0/interaction.js"></script>
   <script type="text/javascript">
       var callback = function (response) {
           if (response.result) {
              alert('Width was set successfully.');
           }
           else {
              alert('Width was not set successfully.');
           }
        };
</script>
</head>
<body>
```

```
       <button onclick="sforce.interaction.cti.setSoftphoneWidth(100, callback);">
       set softphone width to 100px
       </button>
</body>
</html>
```

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
`result`	boolean	`true` if the width was set successfully, `false` if setting the width wasn't successful.
`error`	string	If the API call was successful, this variable is undefined. If the API call failed, this variable returns an error message.

CHAPTER 6 Glossary

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

CTI Adapter
A lightweight software program that controls the appearance and behavior of a Salesforce SoftPhone. The adapter acts as an intermediary between a third-party computer telephony integration (CTI) system, Salesforce, and a Salesforce CRM Call Center user. It must be installed on any machine that needs access to Salesforce CRM Call Center functionality.

CTI System
The hardware and software that implements computer-telephony integration (CTI) for a particular call center.

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.

SoftPhone
The telephone interface that a Salesforce CRM Call Center user sees in either the sidebar of Salesforce pages or the footer of the Salesforce console.

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