Version 29.0: Winter '14

Live Agent Developer's Guide

Last updated: January 3, 2014

Table of Contents

Chapter 1

About This Guide

Live Agent lets service organizations connect with customers or website visitors in real time through a Web-based, text-only live chat. This guide is for developers who are responsible for customizing Live Agent according to their company's needs. It provides several examples to help you understand and create customized chat windows, buttons, forms, and pages.

You can customize Live Agent to create a personalized chat experience for your customer service agents and the customers they serve using custom code. In this guide, we'll show you how to:

- Customize deployments using the Deployment API.
- Customize the appearance of customer-facing chat windows using Visualforce pages and components.
- Create pre-chat forms to gather information from customers before they begin a chat with an agent.
- Create post-chat pages that appear to customers after a chat is complete.

Additionally, you can customize these and other Live Agent components through Salesforce settings. For more information, see Setting Up Live Agent in the Salesforce online help.

Chapter 2

Prerequisites

Before you customize Live Agent, make sure:

- Live Agent is enabled in your organization. Refer to the Live Agent Implementation Guide for detailed information.
- Your administrator has granted you a Live Agent feature license. Although you can customize the product without a feature license, having one will allow you to access and test your customizations.
- You've created a Force.com site and uploaded images as static resources for your chat buttons and windows. Your site needs to have the following information:
  - A site label and a site name
  - A site contact
  - The active site's home page
  - A site template

  If you plan to customize Live Agent without using a Force.com site, skip this step.

Chapter 3

API Versions

Different methods and parameters are available in different versions of Live Agent's APIs. Before you begin developing with the Deployment API or the Pre-Chat API, make sure you're using the correct API version number in your code.

Deployment API Versions

You can find out what version of the Deployment API your organization uses from the deployment code that's generated after you create a deployment.

Summer '13 and earlier releases support version 28.0 of the Deployment API. The URL for API version 28.0 looks like this:
`https://hostname.salesforceliveagent.com/content/g/deployment.js`

Winter '14 supports version 29.0 of the Deployment API. The URL for API version 29.0 contains the version number:
`https://hostname.salesforceliveagent.com/content/g/js/29.0/deployment.js`

Note: To use new methods and parameters in your deployments, you must update the deployment code on each of your Web pages to use the URL for version 29.0 of the Deployment API.

Pre-Chat Information API Versions

Winter '14 supports version 29.0 of the Pre-Chat API. The URL for API version 29.0 contains the version number:
`https://hostname.salesforceliveagent.com/content/g/js/29.0/prechat.js`

You can find your organization's hostname by looking in the deployment code that's generated after you create a deployment.

Chapter 4

Customizing Deployments with the Deployment API

A deployment is a place on your company's website that's enabled for Live Agent. You can customize deployments using the Live Agent Deployment API.

A deployment consists of a few lines of JavaScript that you add to a Web page. Your organization can have a single Live Agent deployment or multiple deployments. For example, if you have a single service center that supports multiple websites, creating a separate deployment for each site lets you present different chat windows to your visitors. Each deployment includes a chat window, which visitors use to chat with support agents.

The Deployment API is a JavaScript-based API that lets you customize your deployments to specify back-end functionality.

Creating Deployments

Create a deployment to host Live Agent on your website. Each deployment includes a chat window, which visitors use to chat with support agents.

Logging Deployment Activity with the Deployment API

Log the activity that occurs in a particular deployment using the Deployment API.

Customizing Your Chat Window with the Deployment API

Customize the dimensions of your customer-facing chat windows using the Deployment API.

Launching a Chat Request with the Deployment API

Use the Deployment API to customize how chat requests are launched.

Customizing Visitor Details with the Deployment API

Use the Deployment API to customize the visitor information of customers who request chats. This information is visible to the agent before they begin their chat with the customer.

Creating Records Automatically with the Deployment API

Use the Deployment API to search for or create customer records automatically when an agent begins a chat with a customer.

Customizing Chat Buttons with the Deployment API

Customize the chat buttons that appear on your website using the Deployment API.

Customizing Automated Chat Invitations with the Deployment API

Use the Deployment API to customize automated chat invitations that appear to customers on your website.

Deployment API Code Sample

Test and preview how the Deployment API can help you customize your deployments using this code sample.

Creating Deployments

Create a deployment to host Live Agent on your website. Each deployment includes a chat window, which visitors use to chat with support agents.

Once you create a deployment, you can customize it using the Deployment API to meet your company's needs.

To create a deployment:

1. From Setup, click **Customize** > **Live Agent** > **Deployments**.
2. Click **New**.
3. Enter a name for the deployment. This name, or a version of it, automatically becomes the `Developer Name`.
4. Enter a title for the chat window.
5. Select `Allow Visitors to Save Transcripts` to let visitors download a copy of the chat session when it ends.
6. Select the site that you'll associate with the deployment.
7. In `Chat Window Branding Image`, select the graphic that will appear in the chat window.
8. In `Mobile Chat Window Branding Image`, select the graphic that visitors using mobile devices will see in the chat window.
9. Click **Save**. Salesforce generates the deployment code.
10. Copy the deployment code and paste it on each Web page where you want to deploy Live Agent. For best performance, paste the code right before the closing body tag.

For more information on creating a deployment, see Creating Deployments in the Salesforce online help.

Logging Deployment Activity with the Deployment API

Log the activity that occurs in a particular deployment using the Deployment API.

Use the following deployment methods to enable logging on a particular deployment. Logging lets you store information about the activity that occurs within a customer's Web browser as they chat with an agent through a particular deployment. You can add these methods as an additional script within the code that's automatically generated when you create a deployment.

enableLogging

Use the `enableLogging` deployment method to enable logging on a particular deployment.

`enableLogging`

Use the `enableLogging` deployment method to enable logging on a particular deployment.

Usage

Enables logging for a particular deployment, allowing your Web browser's JavaScript console to store information about the activity that occurs within a deployment. Available in API versions 28.0 and later.

Syntax

```
liveagent.enableLogging();
```

Parameters

None

Customizing Your Chat Window with the Deployment API

Customize the dimensions of your customer-facing chat windows using the Deployment API.

Use the following deployment methods to customize the height and width of the chat window that customers will see when they begin a chat with an agent. You can add either of these methods as additional scripts within the code that's automatically generated when you create a deployment.

setChatWindowHeight

Use the `setChatWindowHeight` method to customize the height of your chat window.

setChatWindowWidth

Use the `setChatWindowWidth` method to customize the width of your chat window.

`setChatWindowHeight`

Use the `setChatWindowHeight` method to customize the height of your chat window.

Usage

Sets the height in pixels of the chat window that appears to customers. Available in API versions 28.0 and later.

Syntax

```
void setChatWindowHeight(Number height)
```

Parameters

Name	Type	Description	Available Versions
`height`	Number	The height in pixels of your custom chat window.	Available in API versions 28.0 and later.

`setChatWindowWidth`

Use the `setChatWindowWidth` method to customize the width of your chat window.

Usage

Sets the width in pixels of the chat window that appears to customers. Available in API versions 28.0 and later.

Syntax

```
void setChatWindowWidth(Number width)
```

Parameters

Name	Type	Description	Available Versions
`width`	Number	The width in pixels of your custom chat window.	Available in API versions 28.0 and later.

Launching a Chat Request with the Deployment API

Use the Deployment API to customize how chat requests are launched.

Use the following deployment methods to determine how to launch and route chats when a customer clicks a chat button. You can add either of these methods as additional scripts within the code that's automatically generated when you create a deployment.

startChat

Use the `startChat` method to request a chat from a button in a new window.

startChatWithWindow

Use the `startChatWithWindow` method to request a chat from a button using the name of a window.

`startChat`

Use the `startChat` method to request a chat from a button in a new window.

Usage

Requests a chat from the provided button in a new window.

Syntax

```
void startChat(String buttonId)
```

Parameters

Name	Type	Description	Available Versions
`buttonId`	String	The ID of the chat button for which to request a chat in a new window.	Available in API versions 28.0 and later.

`startChatWithWindow`

Use the `startChatWithWindow` method to request a chat from a button using the name of a window.

Usage

Requests a chat from the provided button using the provided window name. Available in API versions 28.0 and later.

Syntax

```
void startChatWithWindow(String buttonId, String windowName)
```

Parameters

Name	Type	Description	Available Versions
`buttonId`	String	The ID of the chat button for which to request a chat in a new window.	Available in API versions 28.0 and later.

Name	Type	Description	Available Versions
`windowName`	String	The name of the window.	Available in API versions 28.0 and later.

Customizing Visitor Details with the Deployment API

Use the Deployment API to customize the visitor information of customers who request chats. This information is visible to the agent before they begin their chat with the customer.

Use the following deployment methods to customize visitor information when customers request to chat with an agent. You can add any of these methods as additional scripts within the code that's automatically generated when you create a deployment.

addCustomDetail

Use the `addCustomDetail` method to add custom details for each chat visitor.

setName

Use the `setName` method to override the visitor name displayed in the Live Agent console or the Salesforce console.

`addCustomDetail`

Use the `addCustomDetail` method to add custom details for each chat visitor.

Usage

Adds a new custom detail for the chat visitor in the Details chatlet in the Live Agent console. Returns an instance of `CustomDetailMapper` on page 9. Available in API versions 28.0 and later.

Syntax

```
addCustomDetail(String label, String value, (optional) Boolean displayToAgent)
```

Parameters

Name	Type	Description	Available Versions
`label`	String	The label for the custom detail—for example, `"Name"`.	Available in API versions 28.0 and later.
`value`	String	The value of the custom detail—for example, `"John Doe"`.	Available in API versions 28.0 and later.
(Optional) `displayToAgent`	Boolean	Specifies whether to display the custom details customers provide in a pre-chat form to the agent (`true`) or not (`false`).	Available in API versions 29.0 and later.

CustomDetailMapper

Use the `CustomDetailMapper` object to add custom details for each chat visitor to the appropriate Live Agent session records.

CustomDetailMapper

Use the `CustomDetailMapper` object to add custom details for each chat visitor to the appropriate Live Agent session records.

Use the following deployment methods to customize visitor information when customers request to chat with an agent. You can add any of these methods as additional scripts within the code that's automatically generated when you create a deployment.

map

Use the `map` method for the `CustomDetailMapper` object to map the values of your custom visitor details to records and their fields.

saveToTranscript

Use the `saveToTranscript` method for the `CustomDetailMapper` object to map the values of your custom visitor details to LiveChatTranscript records.

map

Use the `map` method for the `CustomDetailMapper` object to map the values of your custom visitor details to records and their fields.

Usage

Maps the value of the custom detail to the specified field on the specified entity in the CRM chatlet in the Live Agent console or the relevant record in the Salesforce console. Available in API versions 28.0 and later.

Syntax

```
void map(String entityName, String fieldName, Boolean fastFill, Boolean autoQuery, Boolean exactMatch)
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The entity to which to map the value of the custom detail.	Available in API versions 28.0 and later.
`fieldName`	String	The field within the entity you specified to which to map the value of the custom detail.	Available in API versions 28.0 and later.
`fastFill`	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).	Available in API versions 28.0 and later.
`autoQuery`	Boolean	Specifies whether to perform a a SOSL query (in the Live Agent console) or a SOQL query (in the Salesforce console) (`true`) or not (`false`) to find records with a `fieldName` containing the value.	Available in API versions 28.0 and later.
`exactMatch`	Boolean	Specifies whether to look for exact matches in the comparison syntax (`true`) or to query using a wildcard for the comparison syntax (`false`).	Available in API versions 28.0 and later.

saveToTranscript

Use the `saveToTranscript` method for the `CustomDetailMapper` object to map the values of your custom visitor details to LiveChatTranscript records.

Usage

Saves the value of the custom detail to the specified field on the LiveChatTranscript record that is created at the end of the chat. Available in API versions 28.0 and later.

Syntax

```
saveToTranscript(String fieldName)
```

Parameters

Name	Type	Description	Available Versions
`fieldName`	String	The field within the LiveChatTranscript record to which to save the value of the custom detail.	Available in API versions 28.0 and later.

setName

Use the `setName` method to override the visitor name displayed in the Live Agent console or the Salesforce console.

Usage

Overrides the visitor name displayed in the Live Agent console or the Salesforce console. Available in API versions 28.0 and later.

Syntax

```
setName(String name)
```

Parameters

Name	Type	Description	Available Versions
`name`	String	The visitor name that appears in the Live Agent console or the Salesforce console.	Available in API versions 28.0 and later.

Creating Records Automatically with the Deployment API

Use the Deployment API to search for or create customer records automatically when an agent begins a chat with a customer.

You can add any of these methods as additional scripts within the code that's automatically generated when you create a deployment.

findOrCreate

Use the `findOrCreate` method to find existing records or create new ones based on certain criteria.

Creating Records Deployment API Code Sample

Test and preview how automatically creating records can work with your Live Agent deployments using this code sample.

findOrCreate

Use the `findOrCreate` method to find existing records or create new ones based on certain criteria.

Usage

Finds or creates a record of the specified type when an agent accepts a chat request.

Note: The `findOrCreate` method begins the API call that finds existing records or create new records when an agent begins a chat with a customer. You must use this method before calling any of the other `findOrCreate` sub-methods for finding or creating records with the Deployment API.

Available in API versions 29.0 and later.

Syntax

```
liveagent.findOrCreate(String EntityName)
```

Parameters

Name	Type	Description	Available Versions
`EntityName`	String	The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.	Available in API versions 29.0 and later.

findOrCreate.map

Use the `findOrCreate.map` method to search for or create records that contain specific customer details.

findOrCreate.saveToTranscript

Use the `findOrCreate.saveToTranscript` method to save the record you find or create to the chat transcript associated with the chat.

findOrCreate.showOnCreate

Use the `findOrCreate.showOnCreate` method to automatically open the record you find or create in a subtab in the Salesforce console.

findOrCreate.linkToEntity

Use the `findOrCreate.linkToEntity` method to link the record you found or created to another record type.

findOrCreate.map

Use the `findOrCreate.map` method to search for or create records that contain specific customer details.

Usage

Searches for or creates records that contain customer data specified by the `addCustomDetail` Deployment API method. This method maps the value of the custom details to the fields on the specified record in the Salesforce console.

You can call the `findOrCreate.map` method as many times as necessary to find the appropriate records. Call the method once for every field and its corresponding custom detail value you want to search for.

Available in API versions 29.0 and later.

Syntax

```
liveagent.findOrCreate(Object EntityName).map(String FieldName, String DetailName, Boolean doFind, Boolean isExactMatch, Boolean doCreate)
```

Parameters

Name	Type	Description	Available Versions
`FieldName`	String	The name of the field in the record `EntityName`to which to map the corresponding custom detail `DetailName`.	Available in API versions 29.0 and later.
`DetailName`	String	The value of the custom detail to map to the corresponding field `FieldName`.	Available in API versions 29.0 and later.
`doFind`	Boolean	Specifies whether to search for a record that contains the custom detail `DetailName` in the field `FieldName` (`true`) or not (`false`).	Available in API versions 29.0 and later.
`isExactMatch`	Boolean	Specifies whether to search for a record that contains the exact value of the custom detail `DetailName` you specified in the field `FieldName` (`true`) or not (`false`).	Available in API versions 29.0 and later.
`doCreate`	Boolean	Specifies whether to create a new record with the custom detail `DetailName` in the field `FieldName` if one isn't found (`true`) or not (`false`).	Available in API versions 29.0 and later.

findOrCreate.saveToTranscript

Use the `findOrCreate.saveToTranscript` method to save the record you find or create to the chat transcript associated with the chat.

Usage

Saves the record that you found or created using the `findOrCreate` and `findOrCreate.map` Deployment API methods to the chat transcript associated with the chat.

Available in API versions 29.0 and later.

Syntax

```
liveagent.findOrCreate(String EntityName).saveToTranscript(String TranscriptFieldName)
```

Parameters

Name	Type	Description	Available Versions
`TranscriptFieldName`	String	The name of the field on the chat transcript record to which to save the ID of the record you found or created.	Available in API versions 29.0 and later.

findOrCreate.showOnCreate

Use the `findOrCreate.showOnCreate` method to automatically open the record you find or create in a subtab in the Salesforce console.

Usage

Opens the record you found or created using the `findOrCreate` and `findOrCreate.map` Deployment API methods automatically in a subtab in the to the Salesforce console.

Available in API versions 29.0 and later.

Syntax

```
liveagent.findOrCreate(String EntityName).showOnCreate()
```

findOrCreate.linkToEntity

Use the `findOrCreate.linkToEntity` method to link the record you found or created to another record type.

Usage

Links the record that you found or created using the `findOrCreate` and `findOrCreate.map` Deployment API methods to another record of a different record type that you created using a separate `findOrCreate` API call. For example, you can link a case record you found within your organization to a contact record you create.

Note: You can only link records if the parent record is created with a `findOrCreate` API call. You can't link a child record to a record you found using the `findOrCreate.linkToEntity` method.

Available in API versions 29.0 and later.

Syntax

```
liveagent.findOrCreate(String EntityName).linkToEntity(String EntityName, String FieldName)
```

Parameters

Name	Type	Description	Available Versions
`EntityName`	String	The type of record to which to link the child record you found or created.	Available in API versions 29.0 and later.
`FieldName`	String	The name of the field in the record `EntityName` to which to save the ID of the child record you found or created.	Available in API versions 29.0 and later.

Creating Records Deployment API Code Sample

Test and preview how automatically creating records can work with your Live Agent deployments using this code sample.

The following code searches for and creates records when an agent begins a chat with a customer using the following methods:

- `findOrCreate`
- `findOrCreate.map`
- `findOrCreate.saveToTranscript`
- `findOrCreate.linkToEntityfindOrCreate.showOnCreate`

```
liveagent.addCustomDetail("First Name", "Ryan");
liveagent.addCustomDetail("Last Name", "Smith");
liveagent.addCustomDetail("Phone Number", "555-1212");
liveagent.addCustomDetail("Case Subject", "Problem with my iPhone");
liveagent.addCustomDetail("Case Status", "New", false);
liveagent.findOrCreate("Contact").map("FirstName", "First Name", true, true,
true).map("LastName", "Last Name", true, true, true).map("Phone", "Phone Number", false,
false,true).saveToTranscript("contactId").showOnCreate().linkToEntity("Case", "ContactId");
liveagent.findOrCreate("Case").map("Subject", "Case Subject", true, false, true).map("Status",
 "Case Status", false, false, true).showOnCreate();
```

Customizing Chat Buttons with the Deployment API

Customize the chat buttons that appear on your website using the Deployment API.

Use the following deployment methods to customize your chat buttons. You can add any of these methods as additional scripts within the code that's automatically generated when you create a deployment.

showWhenOnline

Use the `showWhenOnline` method to specify what customers see when particular button is online.

showWhenOffline

Use the `showWhenOffline` method to specify what customers see when particular button is offline.

addButtonEventHandler

Use the `addButtonEventHandler` method to define a chat button's behavior when certain events occur.

`showWhenOnline`

Use the `showWhenOnline` method to specify what customers see when particular button is online.

Usage

Displays a particular element when the specified button is online. Available in API versions 28.0 and later.

Syntax

```
void showWhenOnline(String buttonId, Object element)
```

Parameters

Name	Type	Description	Available Versions
`buttonId`	String	The ID of the chat button for which to display the specified `element` object when agents associated with the button are available to chat.	Available in API versions 28.0 and later.
`element`	Object	The element to display when the specified button is online.	Available in API versions 28.0 and later.

showWhenOffline

Use the `showWhenOffline` method to specify what customers see when particular button is offline.

Usage

Displays a particular element when the specified button is offline. Available in API versions 28.0 and later.

Syntax

```
void showWhenOffline(String buttonId, element)
```

Parameters

Name	Type	Description	Available Versions
`buttonId`	String	The ID of the chat button for which to display the specified `element` object when no agents are available to chat.	Available in API versions 28.0 and later.
`element`	Object	The element to display when the specified button is offline.	Available in API versions 28.0 and later.

addButtonEventHandler

Use the `addButtonEventHandler` method to define a chat button's behavior when certain events occur.

Usage

Defines the behavior for a chat button when the following events occur:

- An agent is available to chat.
- No agents are available to chat.

Available in API versions 28.0 and later.

Syntax

```
void addButtonEventHandler(String buttonId, Function callback)
```

Parameters

Name	Type	Description	Available Versions
`buttonId`	String	The ID of the chat button for which to define the behavior when certain events occur.	Available in API versions 28.0 and later.
`callback`	function	The function to call when a particular event occurs. You must specify the button's behavior for each of the required event types on page 16.	Available in API versions 28.0 and later.

Event Types

Incorporate the following event types into your `callback` function to customize the behavior of your button when certain events occur. You must specify the button's behavior for each of the following event types.

Function	Event Type	Syntax	Description
`callback`	BUTTON_AVAILABLE	`liveagent.BUTTON_EVENT.BUTTON_AVAILABLE`	Specifies the behavior of the button when the criteria are met for customers to be able to chat with an agent, such as when an agent with the correct skills is available to chat.
	BUTTON_UNAVAILABLE	`liveagent.BUTTON_EVENT.BUTTON_UNAVAILABLE`	Specifies the behavior of the button when no agents are available to chat.

Customizing Automated Chat Invitations with the Deployment API

Use the Deployment API to customize automated chat invitations that appear to customers on your website.

Use the following deployment methods to customize your automated chat invitations.

rejectChat

Use the `rejectChat` method to reject and retract an invitation that's been sent to a customer.

addButtonEventHandler

Use the `addButtonEventHandler` method to define an automated invitation's behavior when certain events occur.

setCustomVariable

Use the `setCustomVariable` method to create customized criteria in your sending rules that must be met in order for your automated invitation to be sent to customers.

Automated Chat Invitation Code Sample

Test and preview how automated chat invitations can work on your website using this code sample.

rejectChat

Use the `rejectChat` method to reject and retract an invitation that's been sent to a customer.

Usage

Rejects an invitation and causes it to be retracted.

Available in API versions 28.0 and later.

Syntax

```
void rejectChat(String buttonId)
```

Parameters

Name	Type	Description	Available Versions
`buttonId`	String	The ID of the chat button for which to reject chats.	Available in API versions 28.0 and later.

addButtonEventHandler

Use the `addButtonEventHandler` method to define an automated invitation's behavior when certain events occur.

Usage

Defines the behavior for an invitation when the following events occur:

- The criteria are met for the invitation to appear on-screen.
- The criteria are not met for the invitation to appear on-screen.
- A customer accepts an invitation to chat.
- A customer rejects an invitation to chat.

Available in API versions 28.0 and later.

Syntax

```
void addButtonEventHandler(String buttonId, Function callback)
```

Parameters

Name	Type	Description	Available Versions
`buttonId`	String	The ID of the chat button associated with the automated invitation for which to define the behavior when certain events occur.	Available in API versions 28.0 and later.
`callback`	function	The function to call when a particular event occurs. You must specify the invitation's behavior for each of the required event types on page 18.	Available in API versions 28.0 and later.

Event Types

Incorporate the following event types into your `callback` function to customize the behavior of your invitation when certain events occur. You must specify the invitation's behavior for each of the following event types.

Function	Event Type	Syntax	Description
`callback`	BUTTON_AVAILABLE	`liveagent.BUTTON_EVENT.BUTTON_AVAILABLE`	Specifies the behavior of the automated invitation when the criteria are met for the invitation to appear on-screen.
	BUTTON_UNAVAILABLE	`liveagent.BUTTON_EVENT.BUTTON_UNAVAILABLE`	Specifies the behavior of the automated invitation when no agents are available to chat.
	BUTTON_ACCEPTED	`liveagent.BUTTON_EVENT.BUTTON_ACCEPTED`	Specifies the behavior of the automated invitation when a customer accepts the invitation.
	BUTTON_REJECTED	`liveagent.BUTTON_EVENT.BUTTON_REJECTED`	Specifies the behavior of the automated invitation when a customer rejects the invitation.

setCustomVariable

Use the `setCustomVariable` method to create customized criteria in your sending rules that must be met in order for your automated invitation to be sent to customers.

Usage

Creates customized criteria in your sending rules that must be met in order for your automated invitation to be sent to customers. Specifies the comparison values for custom variables used in criteria for your sending rules. Available in API versions 28.0 and later.

Syntax

```
void setCustomVariable(String variableName, Object value)
```

Parameters

Name	Type	Description	Available Versions
`variableName`	String	The name of the customized criteria for your custom sending rule.	Available in API versions 28.0 and later.
`value`	Object	The comparison value for your custom sending rule.	Available in API versions 28.0 and later.

Automated Chat Invitation Code Sample

Test and preview how automated chat invitations can work on your website using this code sample.

The following code is for an automated chat invitation that uses the `addButtonEventHandler()` method to display a customized invitation on a website. This invitation allows customers to start a chat with an agent when an agent with the correct skills is available to chat.

```
<apex:page>
<div id="liveagent_invite_button_573x00000000001O" style="display: none; position: fixed;
border: 2px solid darkblue; border-radius: 5px; background-color: lightblue; height: 100px;
 width: 200px;">
<div style="cursor: pointer; padding: 5px; right: 0px; position: absolute; color: darkred;
 font-weight: bold;" onclick="liveagent.rejectChat('573x00000000001O')">X</div>
<div style="cursor: pointer; top: 42px; left: 65px; position: absolute; font-weight: bold;
 font-size: 16px;" onclick="liveagent.startChat('573x00000000001O')">Start Chat</div>
</div>

<script type='text/javascript'
src='https://c.la1s1.saleforceliveagent.com/content/g/deployment.js'></script>
<script type='text/javascript'>
function buttonCallback(e) {
    if (e == liveagent.BUTTON_EVENT.BUTTON_AVAILABLE) {
        document.getElementById('liveagent_invite_button_573x00000000001O').style.display =
 '';
        document.getElementById('liveagent_invite_button_573x00000000001O').style.left =
'300px';
        document.getElementById('liveagent_invite_button_573x00000000001O').style.top =
'200px';
    }
    if (e == liveagent.BUTTON_EVENT.BUTTON_UNAVAILABLE) {
        document.getElementById('liveagent_invite_button_573x00000000001O').style.display =
 'none';
    }
    if (e == liveagent.BUTTON_EVENT.BUTTON_ACCEPTED) {
        document.getElementById('liveagent_invite_button_573x00000000001O').style.display =
 'none';
    }
    if (e == liveagent.BUTTON_EVENT.BUTTON_REJECTED) {
        document.getElementById('liveagent_invite_button_573x00000000001O').style.display =
 'none';
    }
}
liveagent.addButtonEventHandler('573x00000000001O', buttonCallback);
liveagent.init('https://d.la1s1.salesforceliveagent.com/chat', '572x00000000001',
'00Dx00000001gEH');
</script>
</apex:page>
```

The code above results in an invitation that looks like this:

Deployment API Code Sample

Test and preview how the Deployment API can help you customize your deployments using this code sample.

The following code sample shows a chat window that uses the following Deployment API methods:

- `startChat`
- `showWhenOnline`
- `showWhenOffline`
- `addCustomDetail`
- `setName`
- `map`
- `setChatWindowWidth`
- `setChatWindowHeight`

```
<apex:page >
 <h1>Welcome</h1>
  Thank you for contacting customer support

  <!-- START Button code -->
   <img id="liveagent_button_online_573D000000000Ar" style="display: none;
   border: 0px none; cursor: pointer" onclick="liveagent.startChat('573D000000000Ar')"
   src="https://na1.salesforce.com/resource/1319587702000/Chat_Online" />

   <img id="liveagent_button_offline_573D000000000Ar" style="display: none;
   border: 0px none; "
   src="https://na1.salesforce.com/resource/1319587748000/Chat_Offline" />

   <script type="text/javascript">
    if (!window._laq) { window._laq = []; }
     window._laq.push(function(){liveagent.showWhenOnline('573D000000000Ar',
     document.getElementById('liveagent_button_online_573D000000000Ar'));
     liveagent.showWhenOffline('573D000000000Ar',
     document.getElementById('liveagent_button_offline_573D000000000Ar'));
   });</script>
  <!-- END Button code -->

  <!-- Deployment code -->
<script type='text/javascript'
 src='https://c.la1s1.saleforceliveagent.com/content/g/deployment.js'></script>

<script type='text/javascript'>
 // An auto query that searches contacts whose email field matches "john@acme.com"
 liveagent.addCustomDetail('Contact E-mail', 'john@acme.com').map('Contact', 'Email', false,
 true);
```

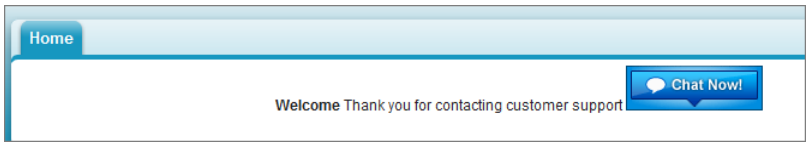
```
 // A fast-fill to populate a contact's name with "John Doe"
 liveagent.addCustomDetail('Contact Name', 'John Doe').map('Contact', 'Name', true, false);

 // Saves the custom detail to a custom field on LiveChatTranscript at the end of a chat
 liveagent.addCustomDetail('Company', 'Acme').saveToTranscript('Company__c');
 // Overrides the display name of the visitor in the agent console when engaged in a chat
 liveagent.setName('John Doe');

 // Sets the width of the chat window to 500px
 liveagent.setChatWindowWidth(500);
 // Sets the height of the chat window to 500px
 liveagent.setChatWindowHeight(500);

 liveagent.init('https://d.la1s1.salesforceliveagent.com/chat', '572D0000000002R',
'00DD0000000JXbY');
</script>
</apex:page>
```

The deployment code above results in a page that looks like this:



Chapter 5

Pre-Chat Forms Overview

The pre-chat experience is what happens between the time visitors request a chat and the time they're connected to a support agent. Use pre-chat forms in Live Agent to customize this experience and collect information from visitors.

You can create a pre-chat form to gather information, such as a customer's name, email address, and reason for contacting customer support. This information can help direct chat requests more efficiently and can reduce the amount of time agents need to spend collecting information before beginning a chat session.

You can create a Visualforce page to host your pre-chat form, or you can develop the form on your own. The information in this guide focuses on using Visualforce.

Note: If a chat is disconnected after the customer completes a pre-chat form, the pre-chat data may be lost when the chat connection is reestablished.

Pre-Chat Form Code Sample

Test and preview how pre-chat forms can work for your agents and customers using this code sample.

See Also:

Creating a Visualforce Page

Pre-Chat Form Code Sample

Test and preview how pre-chat forms can work for your agents and customers using this code sample.

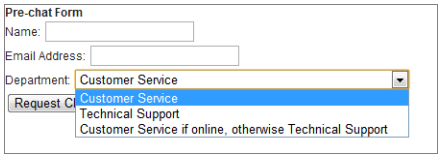
The following code is for a pre-chat form that:

- Requests a visitor's name and email address.
- Displays that information in the Details chatlet in the Live Agent console .
- Enables mapping and auto-query, which triggers a search on the visitor's email address when a support agent opens the CRM chatlet in the Live Agent console.
- Displays a drop-down list that lets visitors choose a different Live Chat button through which to route their chat request.

```
<apex:page showHeader="false">
<!-- This script takes the endpoint URL parameter passed from the deployment page and makes
 it the action for the form -->
<script type="text/javascript">
     (function() {
     function handlePageLoad() {
       var endpointMatcher = new RegExp("[\\?\\&]endpoint=([^&#]*)");
       document.getElementById('prechatForm').setAttribute('action',
       decodeURIComponent(endpointMatcher.exec(document.location.search)[1]));
     } if (window.addEventListener) {
               window.addEventListener('load', handlePageLoad, false);
   } else { window.attachEvent('onload', handlePageLoad, false);
               }})();
```

```
</script>
<h1>Pre-chat Form</h1>
<form method='post' id='prechatForm'>
      Name: <input type='text' name='liveagent.prechat.name' id='prechat_field' /><br />
      Email Address: <input type='text' name='liveagent.prechat:Email' /><br />
      Department: <select name="liveagent.prechat.buttons">
          <!-- Values are LiveChatButton IDs. -->
          <option value="573a00000000001">Customer Service</option>
          <option value="573a00000000002">Technical Support</option>
          <option value="573a00000000001,573a00000000002">Customer Service if online,
          otherwise Technical Support</option>
      </select><br />
      <!-- Creates an auto-query for a matching Contact record's Email field based on the
      value of the liveagent.prechat:Email field -->
      <input type="hidden" name="liveagent.prechat.query:Email"
      value="Contact,Contact.Email" />
      <input type="hidden" name="liveagent.prechat.save:Email" value="Email__c" />
      <input type='submit' value='Request Chat' id='prechat_submit'/>
<style type="text/css">
p {font-weight: bolder }
</style>
</form>
</apex:page>
```

The code above results in a pre-chat form that looks like this:



Chapter 6

Accessing Chat Details with the Pre-Chat API

Use the Pre-Chat API to access customer details from the Deployment API and incorporate them into a pre-chat form.

preChatInit

Use the `preChatInit` method to access the custom details that have been passed into the chat through the `addCustomDetail` Deployment API method.

`preChatInit`

Use the `preChatInit` method to access the custom details that have been passed into the chat through the `addCustomDetail` Deployment API method.

Usage

Extracts the custom details that have been passed into the chat through the `addCustomDetail` Deployment API method and integrates them into a pre-chat form.

Available in API versions 29.0 and later.

Syntax

```
liveagent.details.preChatInit(String chatUrl, function detailCallback, (optional) String chatFormName)
```

Parameters

Name	Type	Description	Available Versions
`chatUrl`	String	The URL of the chat to retrieve custom details from.	Available in API versions 29.0 and later.
`detailCallback`	String	Name of the JavaScript function to call upon completion of the method.	Available in API versions 29.0 and later.
(Optional)`chatFormName`	String	The name of the HTML form tag for the pre-chat form to which to incorporate the custom details.	Available in API versions 29.0 and later.

Responses

Name	Type	Description	Available Versions
`details`	Object	An object containing all of the custom details that were included in the pre-chat form using the `preChatInit` method.	Available in API versions 29.0 and later.

The `details` object has a structure similar to the following example object:

```
{
    "geoLocation":{
        "countryCode":"US",
        "countryName":"United States",
        "longitude":-122.4294,
        "organization":"SALESFORCE.COM",
        "latitude":37.764496,
        "region":"CA",
        "city":"San Francisco"

    },
    "customDetails":[
        {
            "label":"Email",
            "value":"sonic@sega.com",
            "transcriptFields":["Email__c"],
            "entityMaps":[
            {
                "fieldName":"Email",
                "isAutoQueryable":true,
                "entityName":"Contact",
                "isExactMatchable":true,
                "isFastFillable":false
            }]
        },
        {
            "label":"Name",
            "value":"Sonic H.",
            "transcriptFields":[],
            "entityMaps":[]
        }
    ],
    "visitorId":"251a5956-bcbc-433d-b822-a87c062e681c"
}
```

detailCallback

The `detailCallback` method specifies the behavior that should occur after the `preChatInit` method returns the `details` object.

Syntax	Parameters	Description	Available Versions
`function myCallBack(details) { // Customer specific code }`	`details`	Specifies the actions to occur after the custom details are retrieved using the `preChatInit` method.	Available in API versions 29.0 and later.

Chapter 7

Creating Records Automatically with the Pre-Chat API

Use the Pre-Chat API to search for or create customer records automatically when a customer completes a pre-chat form.

findOrCreate.map

Use the `findOrCreate.map` method to search for or create records that contain specific customer details.

findOrCreate.saveToTranscript

Use the `findOrCreate.saveToTranscript` method to save the record you find or create to the chat transcript associated with the chat.

findOrCreate.showOnCreate

Use the `findOrCreate.showOnCreate` method to automatically open the record you find or create in a subtab in the Salesforce console.

findOrCreate.linkToEntity

Use the `findOrCreate.linkToEntity` method to link the record you found or created to another record type.

findOrCreate.displayToAgent

Use the `findOrCreate.displayToAgent` method to specify which pre-chat details will be displayed to an agent in the Details tab when they receive a chat request.

`findOrCreate.map`

Use the `findOrCreate.map` method to search for or create records that contain specific customer details.

Usage

Searches for or creates records that contain the customer data specified in the pre-chat form completed by the customer. This method maps the value of the custom details to the fields on the specified record in the Salesforce console.

You can call the `findOrCreate.map` method as many times as necessary to find the appropriate records. You can list multiple fields and their corresponding details to map the detail values to the appropriate fields within the record.

Available in API versions 29.0 and later.

Syntax

```
<input type= "hidden" name= "liveagent.prechat.findorcreate.map: String entityName"value=
"String fieldName, String detailName;" />
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.	Available in API versions 29.0 and later.
`fieldName`	String	The name of the field in the record `EntityName` to which to map the corresponding custom detail `value`.	Available in API versions 29.0 and later.
`detailName`	String	The value of the custom detail to map to the corresponding field `fieldName`.	Available in API versions 29.0 and later.

findOrCreate.map.doFind

Use the `findOrCreate.map.doFind` method to specify which fields to use to search for existing customer records when a customer completes a pre-chat form.

findOrCreate.map.isExactMatch

Use the `findOrCreate.map.isExactMatch` method to specify whether a field name must exactly match the field name in an existing record when you conduct a search with the `findOrCreate.map` method.

findOrCreate.map.doCreate

Use the `findOrCreate.map.doCreate` method to specify which fields in `findOrCreate.map` method to use to create a new record if an existing record isn't found.

`findOrCreate.map.doFind`

Use the `findOrCreate.map.doFind` method to specify which fields to use to search for existing customer records when a customer completes a pre-chat form.

Usage

Specifies which fields in your `findOrCreate.map` method to use to search for an existing record. You can search for one or more fields within records.

Available in API versions 29.0 and later.

Syntax

```
<input type= "hidden" name= "liveagent.prechat.findorcreate.map.doFind: String entityName"
value= "String fieldName, Boolean find;" />
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.	Available in API versions 29.0 and later.
`fieldName`	String	The name of the field to search for in existing records.	Available in API versions 29.0 and later.

Name	Type	Description	Available Versions
`find`	Boolean	Specifies whether to search for existing records that contain the field `fieldName` (`true`) or not (`false`). **Note:** You only need to specify fields for which `find` equals `true`. The method will not search for records containing fields for which `find` equals `false`.	Available in API versions 29.0 and later.

findOrCreate.map.isExactMatch

Use the `findOrCreate.map.isExactMatch` method to specify whether a field name must exactly match the field name in an existing record when you conduct a search with the `findOrCreate.map` method.

Usage

Specifies which fields in your `findOrCreate.map` method require an exact field name match when you search for existing records. You can specify this for one or more fields within records.

Available in API versions 29.0 and later.

Syntax

```
<input type= "hidden" name= "liveagent.prechat.findorcreate.map.isExactMatch: String entityName" value= "String fieldName, Boolean exactMatch;" />
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.	Available in API versions 29.0 and later.
`fieldName`	String	The name of the field to search for in existing records.	Available in API versions 29.0 and later.
`find`	Boolean	Specifies whether to search for existing records that contain an exact match to the field `fieldName` (`true`) or not (`false`). **Note:** You only need to specify fields for which `exactMatch` equals `true`. The method will not search for records containing fields for which `exactMatch` equals `false`.	Available in API versions 29.0 and later.

findOrCreate.map.doCreate

Use the `findOrCreate.map.doCreate` method to specify which fields in `findOrCreate.map` method to use to create a new record if an existing record isn't found.

Usage

Specifies which fields in your `findOrCreate.map` method to use to create a new record if an existing record isn't found. You can specify one or more fields for creating new records.

Available in API versions 29.0 and later.

Syntax

```
<input type= "hidden" name= "liveagent.prechat.findorcreate.map.doCreate: String entityName"
value= "String fieldName, Boolean create;" />
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The type of record to create when an agent accepts a chat with a customer and an existing record isn't found—for example, a contact record.	Available in API versions 29.0 and later.
`fieldName`	String	The name of the field to include in new records.	Available in API versions 29.0 and later.
`create`	Boolean	Specifies whether to create a new record that contains the field `fieldName` (`true`) or not (`false`). **Note:** You only need to specify fields for which `create` equals `true`. The method will not create records containing fields for which `create` equals `false`.	Available in API versions 29.0 and later.

findOrCreate.saveToTranscript

Use the `findOrCreate.saveToTranscript` method to save the record you find or create to the chat transcript associated with the chat.

Usage

Saves the record that you found or created using the `findOrCreate.map.doCreate` or `findOrCreate.map.doFind` Pre-Chat API methods to the chat transcript associated with the chat when the chat ends.

Available in API versions 29.0 and later.

Syntax

```
<input type="hidden" name= "liveagent.prechat.findorcreate.saveToTranscript: String
entityName" value= "String transcriptFieldName" />
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.	Available in API versions 29.0 and later.
`transcriptFieldName`	String	The name of the field on the chat transcript record to which to save the ID of the record you found or created.	Available in API versions 29.0 and later.

findOrCreate.showOnCreate

Use the `findOrCreate.showOnCreate` method to automatically open the record you find or create in a subtab in the Salesforce console.

Usage

Opens the record you found or created using the `findOrCreate.map.doCreate` and `findOrCreate.map.doFind` Pre-Chat API methods automatically in a subtab in the to the Salesforce console.

Available in API versions 29.0 and later.

Syntax

```
<input type= "hidden" name= "liveagent.prechat.findorcreate.showOnCreate: String entityName"
value= "Boolean show" />
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.	Available in API versions 29.0 and later.
`show`	Boolean	Specifies whether to display the record you found or created in a subtab in the Salesforce console (`true`) or not (`false`).	Available in API versions 29.0 and later.

findOrCreate.linkToEntity

Use the `findOrCreate.linkToEntity` method to link the record you found or created to another record type.

Usage

Links the record that you found or created using the `findOrCreate.map.doFind` and `findOrCreate.map.doCreate` Pre-Chat API methods to another record of a different record type that you created using a separate `findOrCreate.map` API call. For example, you can link a case record you found within your organization to a contact record you create.

Note: You can only link records if the parent record is created with a `findOrCreate` API call. You can't link a child record to a record you found using the `findOrCreate.linkToEntity` method.

Available in API versions 29.0 and later.

Syntax

```
<input type= "hidden" name= "liveagent.prechat.findorcreate.linkToEntity: String entityName"
value= "String parentEntityName, String fieldName" />
```

Parameters

Name	Type	Description	Available Versions
`entityName`	String	The type of record to which to link the child record you found or created.	Available in API versions 29.0 and later.
`parentEntityName`	String	The type of parent record to link to the child record you found or created.	Available in API versions 29.0 and later.
`fieldName`	String	The name of the field in the record `parentEntityName` to which to save the ID of the child record you found or created.	Available in API versions 29.0 and later.

`findOrCreate.displayToAgent`

Use the `findOrCreate.displayToAgent` method to specify which pre-chat details will be displayed to an agent in the Details tab when they receive a chat request.

Usage

Specifies which pre-chat details to display to an agent in the Details tab in Salesforce console when the agent receives a chat request.

Available in API versions 29.0 and later.

Syntax

```
<input type= "hidden" name= "liveagent.prechat.findorcreate.displayToAgent: String detailName"
value= "Boolean display" />
```

Parameters

Name	Type	Description	Available Versions
`detailName`	String	The name of the detail to display to an agent when they receive a chat request.	Available in API versions 29.0 and later.

Name	Type	Description	Available Versions
`display`	Boolean	Specifies whether to display the customer detail to an agent in the Details tab in the Salesforce console (`true`) or not (`false`).  **Note:** You only need to specify details for which `display` equals `true`. The method will not display details for which `exactMatch` equals `false`.	Available in API versions 29.0 and later.

Chapter 8

Customizing Chat Windows with Visualforce

Chat windows are what visitors use to exchange messages with support agents. Each of your Live Agent deployments include a chat window. You can create a customized chat window using Visualforce, and you can add additional styling and functionality with HTML and JavaScript.

For more information on using Visualforce, see the Visualforce Developers' Guide.

Accessing Chat Details for your Visualforce Chat Window

Use JavaScript to access customer details and chat information from the Deployment API and the Pre-Chat API to incorporate them into a custom Visualforce chat window.

Live Agent Visualforce Components

Use Visualforce components to customize the appearance and performance of chat windows.

Live Agent Visualforce Components Code Sample

Use this code sample to test and preview how Visualforce components can help you customize your chat windows.

See Also:

Creating a Visualforce Page

Accessing Chat Details for your Visualforce Chat Window

Use JavaScript to access customer details and chat information from the Deployment API and the Pre-Chat API to incorporate them into a custom Visualforce chat window.

getDetails

Use the `getDetails` JavaScript method to access custom details and chat information that have been passed through the Deployment API and Pre-Chat API and incorporate them into your Visualforce chat window.

`getDetails`

Use the `getDetails` JavaScript method to access custom details and chat information that have been passed through the Deployment API and Pre-Chat API and incorporate them into your Visualforce chat window.

Usage

Extracts chat information and custom details that have been passed into the chat through the Deployment API and Pre-Chat API and integrates the information into a Visualforce chat window. Call this method within a custom Visualforce chat window.

Call this method when the chat information and custom details change or become available. To verify that these details have become available or have been updated, you need to include the following three required events in the code for your chat window:

```
liveagent.addEventListener(liveagent.chasitor.Events.CHAT_REQUEST_SUCCESSFUL, myCallBack);

 liveagent.addEventListener(liveagent.chasitor.Events.CHAT_ESTABLISHED, newagent);
liveagent.addEventListener(liveagent.chasitor.Events.AGENT_CHAT_TRANSFERRED, newagent);
```

You can take a look at how to incorporate these events in the code sample below.

Available in API versions 29.0 and later.

Syntax

```
liveagent.chasitor.getDetails();
```

Responses

Name	Type	Description	Available Versions
`details`	Object	An object containing all of the custom details to incorporate into your Visualforce chat window.	Available in API versions 29.0 and later.

The `getDetails` method returns a `details` JavaScript object with a structure similar to the following example object:

```
{
    "geoLocation":{
        "countryCode":"US",
        "countryName":"United States",
        "longitude":-122.4294,
        "organization":"SALESFORCE.COM",
        "latitude":37.764496,
        "region":"CA",
        "city":"San Francisco"

    },
    "customDetails":[
        {
            "label":"Email",
            "value":"example@salesforce.com",
            "transcriptFields":[],
            "entityMaps":[]
        },
        {
            "label":"Email",
            "value":"sonic@sega.com",
            "transcriptFields":["Email__c"],
            "entityMaps":[
            {
                "fieldName":"Email",
                "isAutoQueryable":true,
                "entityName":"Contact",
                "isExactMatchable":true,
                "isFastFillable":false
            }]
        }
    ],
    "prechatDetails":[
        {
            "label":"prechat field",
            "value":"any info",
            "transcriptFields":[],
```

```
            "entityMaps":[]
        }],
    "agent" : {
        "agentName":"jsmith",
        "userId":"005x0000001JGgr",
        "transfer":0
    },
    "visitorId":"251a5956-bcbc-433d-b822-a87c062e681c",
}
```

Code Sample

```
<apex:page showHeader="false">

<script type='text/javascript'>
   liveagent.addEventListener(liveagent.chasitor.Events.CHAT_REQUEST_SUCCESSFUL, myCallBack);

    liveagent.addEventListener(liveagent.chasitor.Events.CHAT_ESTABLISHED, newagent);
   liveagent.addEventListener(liveagent.chasitor.Events.AGENT_CHAT_TRANSFERRED, newagent);


    function myCallBack() {
          var details = liveagent.chasitor.getDetails();
// ..
      }

      function newagent() {
          var details = liveagent.chasitor.getDetails();
// …
      }
</script>

<style>
html {
    padding: 20px;
}
body {
    background-color: #778899;
    overflow: hidden;
    width: 100%;
    height: 100%;
    padding: 20px;
    margin: 0
}
#waitingMessage {
        color:white;
        height: 100%;
        width: 100%;
        vertical-align: middle;
        text-align: center;
        display: none;
 }
#liveAgentClientChat.liveAgentStateWaiting #waitingMessage { display: table; }
#liveAgentSaveButton, #liveAgentEndButton { z-index: 2; }
.liveAgentChatInput {
    top:8px;
    height: 25px;
    border-width: 1px;
    border-style: solid;
    border-color: #BB0000;
    padding: 2px 0 2px 4px;
    background: #fff;
    display: block;
    width: 99%;
}
.liveAgentSendButton {
    display: block;
    width: 60px;
    height: 31px;
```

```
    padding: 0 0 3px;
    position: absolute;
    top: 0;
    right: -67px;
}
#liveAgentChatLog {
    border-color: #BB0000;
    background-image:linear-gradient(#DDEEFF, #AABBCC);
    box-shadow: 10px 10px 5px #888888;
    border-radius: 10px;
    padding:10px;
    width: auto;
    height: auto;
    top: 38px;
    position: absolute;
    overflow-y: auto;
    left: 0;
    right: 0;
    bottom: 0;
}
</style>

<div style="top: 0; left: 0; right: 0; bottom: 0; position: absolute; padding:10px;">
<liveAgent:clientchat >
    <liveAgent:clientChatSaveButton label="Save Chat" />
    <liveAgent:clientChatEndButton label="End Chat" />
    <div id="prechatdata">
    </div>
    <div style="top: 5px; left: 5px; right: 5px; bottom: 5px; position: absolute; z-index:
 0;">
    <liveAgent:clientChatAlertMessage />
    <liveAgent:clientChatStatusMessage />
    <table id="waitingMessage" cellpadding="0" cellspacing="0">
    <tr>
    <td>Please wait while you are connected to an available agent.</td>
    </tr>
    </table>
    <div style="top: 0; right: 0; bottom: 41px; left: 0; padding: 0; position: absolute;
word-wrap: break-word; z-index: 0;">
    <liveAgent:clientChatLog />
    </div>
   <div style="position: absolute; height: auto; right: 0; bottom: 0; left: 0; margin-right:
 67px;">
        <liveagent:clientChatInput /><liveAgent:clientChatSendButton label="Send"/>
    </div>
    </div>
</liveAgent:clientchat>
</div>
</apex:page>
```

Live Agent Visualforce Components

Use Visualforce components to customize the appearance and performance of chat windows.

Live Agent includes the following customizable Visualforce components.

Component Name	Description
`liveAgent:clientChat`	The main parent element for any Live Agent chat window. You must create this element in order to do any additional customization of Live Agent. Note that this component can only be used once in a Live Agent deployment.
`liveAgent:clientChatEndButton`	The button within a Live Agent chat window that a visitor clicks to end a chat session. Must be used within `liveAgent:clientChat`.

Component Name	Description
`liveAgent:clientChatInput`	The text box in a Live Agent chat window where a visitor types messages to a support agent. Must be used within `liveAgent:clientChat`. Each chat window can have only one input box.
`liveAgent:clientChatLog`	The area in a Live Agent chat window that displays the chat transcript to a visitor. Must be used within `liveAgent:clientChat`. Each chat window can have only one chat log.
`liveAgent:clientChatMessages`	The area in a Live Agent chat window that displays system status messages, such as "Chat session has been disconnected." Must be used within `liveAgent:clientChat`. Each chat window can have only one message area.
`liveAgent:clientChatQueuePosition`	A text label indicating a visitor's position in a queue for a chat session that's initiated by a button that uses push routing. (This component has no effect on buttons that use pull routing.) Must be used within `liveAgent:clientChat`. For more information on this component, see Using `liveAgent:clientChatQueuePosition`.
`liveAgent:clientChatSaveButton`	The button in a Live Agent chat window that a visitor clicks to save the chat transcript as a local file. Must be used within `liveAgent:clientChat`. Each chat window can have multiple save buttons.
`liveAgent:clientChatSendButton`	The button in a Live Agent chat window that a visitor clicks to send a chat message to an agent. Must be used within `liveAgent:clientChat`. Each chat window can have multiple send buttons.

For more information about each of these components, see the Visualforce Component Guide.

Using `liveAgent:clientChatQueuePosition`

The `liveAgent:clientChatQueuePosition` component shows where in the chat queue a visitor is. In order for a chat to enter the queue:

- The button from which the chat was requested must have queuing enabled.
- All online agents (with the relevant skills, if applicable) must be at capacity, causing a queue to form.
- The chat must be in the queue and not yet assigned to an agent.

If all three of these conditions aren't met, `liveAgent:clientChatQueuePosition` doesn't display a value.

Live Agent Visualforce Components Code Sample

Use this code sample to test and preview how Visualforce components can help you customize your chat windows.

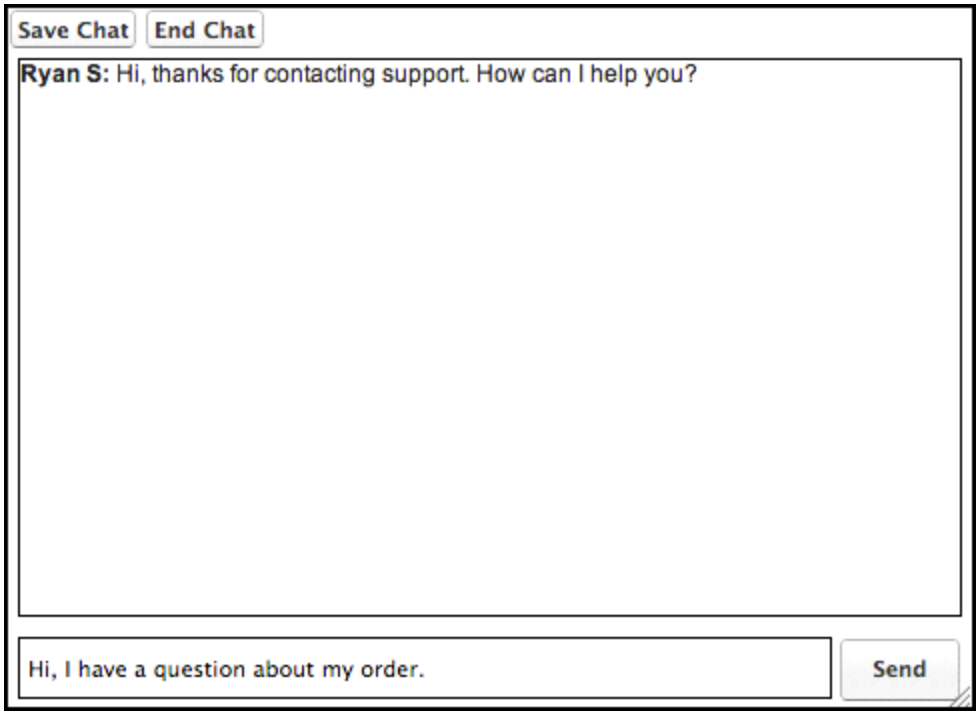
The following code sample shows a chat window that uses the following components:

- `liveAgent:clientChat`
- `liveAgent:clientChatMessages`
- `liveAgent:clientChatEndButton`
- `liveAgent:clientChatLog`
- `liveAgent:clientChatInput`
- `liveAgent:clientChatEndButton`

```
<apex:page showHeader="false">
<style>
body { overflow: hidden; width: 100%; height: 100%; padding: 0; margin: 0 }
```

```
#waitingMessage { height: 100%; width: 100%; vertical-align: middle; text-align: center;
display: none; }
#liveAgentClientChat.liveAgentStateWaiting #waitingMessage { display: table; }
#liveAgentSaveButton, #liveAgentEndButton { z-index: 2; }
.liveAgentChatInput {
    height: 25px;
    border-width: 1px;
    border-style: solid;
    border-color: #000;
    padding: 2px 0 2px 4px;
    background: #fff;
    display: block;
    width: 99%;
}
.liveAgentSendButton {
    display: block;
    width: 60px;
    height: 31px;
    padding: 0 0 3px;
    position: absolute;
    top: 0;
    right: -67px;
}
#liveAgentChatLog {
    width: auto;
    height: auto;
    top: 0px;
    position: absolute;
    overflow-y: auto;
    left: 0;
    right: 0;
    bottom: 0;
}
</style>
<div style="top: 0; left: 0; right: 0; bottom: 0; position: absolute;">
<liveAgent:clientchat >
    <liveAgent:clientChatSaveButton label="Save Chat" />
    <liveAgent:clientChatEndButton label="End Chat" />
   <div style="top: 25px; left: 5px; right: 5px; bottom: 5px; position: absolute; z-index:
 0;">
    <liveAgent:clientChatAlertMessage />
    <liveAgent:clientChatStatusMessage />
    <table id="waitingMessage" cellpadding="0" cellspacing="0">
    <tr>
    <td>Please wait while you are connected to an available agent.</td>
    </tr>
    </table>
    <div style="top: 0; right: 0; bottom: 41px; left: 0; padding: 0; position: absolute;
word-wrap: break-word; z-index: 0;">
    <liveAgent:clientChatLog />
    </div>
   <div style="position: absolute; height: auto; right: 0; bottom: 0; left: 0; margin-right:
 67px;">
        <liveagent:clientChatInput /><liveAgent:clientChatSendButton label="Send"/>
    </div>
    </div>
</liveAgent:clientchat>
</div>
</apex:page>
```

The code above results in a chat window that looks like this:


Save Chat
End Chat
Ryan S: Hi, thanks for contacting support. How can I help you?
Hi, I have a question about my order.
Send

Chapter 9

Post-Chat Pages Overview

Post-chat pages let you share information, such as chat transcripts, with customers at the end of a chat session. You can also customize the post-chat experience in other ways, such as offering options for follow-up support and asking customers to complete a survey.

You can create a Visualforce page to host your post-chat page, or can develop a page on your own. The information in this guide focuses on using Visualforce.

Post-Chat Pages Code Sample

Test and preview how post-chat pages will work for your agents and customers using this code sample.

See Also:

Creating a Visualforce Page

Post-Chat Pages Code Sample

Test and preview how post-chat pages will work for your agents and customers using this code sample.

The following code is for a post-chat page that includes basic information about the chat.

```
<apex:page showHeader="false">
      <h1>Post Chat Page</h1>
      <!-- These variables are passed to the post-chat page and can be used to customize
your post-chat experience -->
     Request Time: <apex:outputText value="{!$CurrentPage.parameters.requestTime}" /><br/>

     Start Time: <apex:outputText value="{!$CurrentPage.parameters.startTime}" /><br/>

      Button Id: <apex:outputText value="{!$CurrentPage.parameters.buttonId}" /><br/>
     Deployment Id: <apex:outputText value="{!$CurrentPage.parameters.deploymentId}" /><br/>

     Last Visited Page: <apex:outputText value="{!$CurrentPage.parameters.lastVisitedPage}"
 /><br/>
     Original Referer: <apex:outputText value="{!$CurrentPage.parameters.originalReferrer}"
 /><br/>
      Latitude: <apex:outputText value="{!$CurrentPage.parameters.latitude}" /><br/>
      Longitude: <apex:outputText value="{!$CurrentPage.parameters.longitude}" /><br/>

      City: <apex:outputText value="{!$CurrentPage.parameters.city}" /><br/>
      Region: <apex:outputText value="{!$CurrentPage.parameters.region}" /><br/>
      Country: <apex:outputText value="{!$CurrentPage.parameters.country}" /><br/>
     Organization: <apex:outputText value="{!$CurrentPage.parameters.organization}" /><br/>

      Transcript: <apex:outputText value="{!$CurrentPage.parameters.transcript}" /><br/>

      Disconnected By: <apex:outputText value="{!$CurrentPage.parameters.disconnectedBy}"
/><br/>
      Chat Key: <apex:outputText value="{!$CurrentPage.parameters.chatKey}" /><br />
```

```
      Chat Details: <apex:outputText value="{!$CurrentPage.parameters.chatDetails}" /><br
/>
      Error: <apex:outputText value="{!$CurrentPage.parameters.error}" /><br />
     <!-- Implement your post-chat message, form, or survey here -->
</apex:page>
```

The code above results in a post-chat page that looks like this:

Chapter 10

Creating a Visualforce Page

Create pre-chat forms and post-chat pages using Visualforce to enhance your customers' and agents' chat experience. In addition, creating a Visualforce page to host your pre-chat forms and post-chat pages makes it easy to customize them.

To create a new Visualforce page:

1. From Setup, click **Develop** > **Pages**.
2. Click **New**.

Add page components and specifications based on your company's needs.