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 “Customize Your Live Agent Implementation” in the Salesforce help.
CHAPTER 2 Prerequisites

Before you customize Live Agent, make sure:

- Live Agent is enabled in your organization.
- 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. If you plan to customize Live Agent without using a Force.com site, skip this step.

**Note:** When using a Force.com site for Live Agent custom chat pages, avoid using the path “/liveagent” in the URL. This path sometimes causes errors with the incoming and outgoing chat notification sounds, so agents will be unable to hear their chat updates.
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  Customize 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 by 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 enables you to present multiple 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.

IN THIS SECTION:

  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.

  Find and Create Records Automatically with the Deployment API
Use the Deployment API to search for or create Salesforce records—like a case, contact, account, or lead—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.
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.

You can customize your Live Agent deployments using the Deployment API to meet your company’s needs. The deployment code that generates after completing these steps should be placed on all pages that you want to enable chat and tracking. Pages with the deployment code are automatically tracked as part of the visitor’s chat session, and they’re shown to the agent in the Console when the visitor requests a chat. This tracking also enables automated invitations to be presented to customers.

To create a deployment:

1. From Setup, enter Deployments in the Quick Find box, then select 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.

Example: For more information on creating a deployment, see “Create Live Agent Deployments” in the Salesforce 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.

IN THIS SECTION:

enableLogging

Use the enableLogging deployment method to enable logging on a particular deployment. Available in API versions 28.0 and later.

enableLogging

Use the enableLogging deployment method to enable logging on a particular deployment. Available in API versions 28.0 and later.
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. You can retrieve the information from your browser’s developer console, so check the help for your browser if you’re not sure how to find it.

Syntax
liveagent.enableLogging();

Parameters
None

Messages for Logged Events

<table>
<thead>
<tr>
<th>Message</th>
<th>Triggered</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>System initialized. Waiting for the DOM to be ready.</td>
<td>When liveagent.init() is called, usually at page load</td>
<td>Live Agent endpoint URL, org ID and deployment ID have been set, now waiting for DOM to be ready before continuing.</td>
</tr>
<tr>
<td>No available event model. Exiting.</td>
<td>During liveagent.init(), if there is an error</td>
<td>This means no DOM event listener was found, which would be very rare. We would not be able to continue at this point, so it would be a hard stop.</td>
</tr>
<tr>
<td>DOM is ready. Setting up environment.</td>
<td>Upon DOM ready of the page</td>
<td>The page has fully loaded and the DOM is ready, so we perform our first &quot;ping&quot; to the server, which is to get the settings/information about the given deployment ID.</td>
</tr>
<tr>
<td>Setting state for button {Button ID} to online</td>
<td>When the state of a button has changed to online</td>
<td>The button is available for a chat request to be made.</td>
</tr>
<tr>
<td>Setting state for button {Button ID} to offline</td>
<td>When the state of a button has changed to offline</td>
<td>The button is not available for a chat request to be made.</td>
</tr>
<tr>
<td>Requesting new session</td>
<td>During the first ping to the server</td>
<td>No session ID cookie was found, so a new one must be generated. This means it was the first time visiting the site with this deployment code for this browsing session.</td>
</tr>
<tr>
<td>Reusing existing session</td>
<td>During the first ping to the server</td>
<td>A session cookie exists, so it is reused. This means the visitor has already been to this site during this browsing session (e.g., going from one page to another).</td>
</tr>
<tr>
<td>Received new session ID</td>
<td>As a response to the first ping</td>
<td>The server generated a new session ID, and it is being stored as a session cookie named &quot;liveagent_sid.&quot;</td>
</tr>
</tbody>
</table>
### Message Table

<table>
<thead>
<tr>
<th>Message</th>
<th>Triggered</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ping rate set to (Rate)ms</td>
<td>As a response to the first ping</td>
<td>Indicates how frequently (in milliseconds) the page will ping the Live Agent server. The default is 50000 (50 seconds). This effectively indicates when button refreshes will occur.</td>
</tr>
<tr>
<td>Pinging server to keep presence</td>
<td>When a ping to the server is made</td>
<td>Indicates the visitor is still connected to and pinging the Live Agent server, meaning no errors or disconnects have occurred.</td>
</tr>
<tr>
<td>Disconnecting from Live Agent</td>
<td>When an error occurs</td>
<td>An error was thrown, whether in response from the server or due to network connectivity issues. Indicates that the visitor will no longer ping Live Agent for this page load (i.e., they will need to refresh).</td>
</tr>
<tr>
<td>Received updated LiveAgent server url: (URL)! Consider updating this site's deployment code.</td>
<td>When an org has moved to a new core instance</td>
<td>The Live Agent instance specified in the deployment code is no longer valid for this org, so the new URL has been provided. For better performance, we recommend updating the deployment code if they receive this.</td>
</tr>
<tr>
<td>Server Warning: (Message)</td>
<td>A non-fatal exception occurred</td>
<td>A warning condition was encountered, but processing can continue. The message provides further details.</td>
</tr>
<tr>
<td>Server sent an anonymous warning</td>
<td>A non-fatal exception occurred</td>
<td>A warning condition was encountered, but processing can continue. No message was provided.</td>
</tr>
<tr>
<td>Server Error: (Message)</td>
<td>A fatal exception occurred</td>
<td>An error condition was encountered, and processing cannot be continued. The message provides further details.</td>
</tr>
<tr>
<td>Server responded with an error</td>
<td>A fatal exception occurred</td>
<td>An error condition was encountered, and processing cannot be continued. No message was provided.</td>
</tr>
<tr>
<td>Group Start: Invite (Button ID) Rule Evaluation</td>
<td>Rule evaluation has been triggered</td>
<td>Evaluation of the filter logic for the given invite button ID has begun. This means the button is online and available for chat, and the filter logic will be used to determine if it should be displayed/presented or not.</td>
</tr>
<tr>
<td>Filter Logic: (Filter Logic)</td>
<td>Rule evaluation has been triggered</td>
<td>An information log containing the string representation of the filter logic of the invite rules as specified in the admin setup area. Useful to understand how the rules will be evaluated.</td>
</tr>
</tbody>
</table>
### Meaning of Triggered Messages

<table>
<thead>
<tr>
<th>Message</th>
<th>Triggered</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluating StandardInviteRule</td>
<td>When a standard rule is being evaluated</td>
<td>Standard rules are &quot;Number of Page Views&quot; and &quot;URL Match.&quot; They are part of the out-of-the-box rules that are provided in the admin setup area.</td>
</tr>
<tr>
<td>Evaluating TimerInviteRule</td>
<td>When a timer-based rule is being evaluated</td>
<td>Timer-based rules are &quot;Seconds on Page&quot; and &quot;Seconds on Site.&quot; They are part of the out-of-the-box rules as well, except these rules will be re-evaluated again in the future when the required number of seconds has passed if the criteria was not met the first time (e.g., on page load).</td>
</tr>
<tr>
<td>Evaluating CustomInviteRule</td>
<td>When a custom rule is being evaluated</td>
<td>&quot;Custom Variable&quot; rules allow variable names to be specified which will be compared against upon evaluating these rules. The &quot;setCustomVariable&quot; API is used in conjunction with these to specify the value to compare against the value specified in the admin setup area.</td>
</tr>
<tr>
<td>CustomInviteRule evaluation failed due to missing custom variable</td>
<td>When a custom rule is being evaluated</td>
<td>A &quot;Custom Variable&quot; rule was set up, but the &quot;setCustomVariable&quot; API was never called with this variable name specified, therefore the rule can not be evaluated.</td>
</tr>
<tr>
<td>Evaluate: {From Value} (!=) {To Value}</td>
<td>When a rule with a &quot;not equal to&quot; comparator is being evaluated</td>
<td>A rule is being evaluated by comparing that the two values do not match.</td>
</tr>
<tr>
<td>Starts With - Evaluate: {From Value} indexOf {To Value} (!=) 0</td>
<td>When a rule with a &quot;starts with&quot; comparator is being evaluated</td>
<td>A rule is being evaluated by comparing that the first value starts with the second value.</td>
</tr>
<tr>
<td>Contains - Evaluate: {From Value} indexOf {To Value} (!=) -1</td>
<td>When a rule with a &quot;contains&quot; comparator is being evaluated</td>
<td>A rule is being evaluated by comparing that the first value contains the second value.</td>
</tr>
<tr>
<td>Does Not Contain - Evaluate: {From Value} indexOf {To Value} (!=) -1</td>
<td>When a rule with a &quot;does not contain&quot; comparator is being evaluated</td>
<td>A rule is being evaluated by comparing that the first value does not contain the second value.</td>
</tr>
<tr>
<td>Less Than - Evaluate: {From Value} (&lt;) {To Value}</td>
<td>When a rule with a &quot;less than&quot; comparator is being evaluated</td>
<td>A rule is being evaluated by comparing that the first value is less than the second value.</td>
</tr>
<tr>
<td>Greater Than - Evaluate: {From Value} (&gt;) {To Value}</td>
<td>When a rule with a &quot;greater than&quot; comparator is being evaluated</td>
<td>A rule is being evaluated by comparing that the first value is greater than the second value.</td>
</tr>
<tr>
<td>Less or Equal - Evaluate: {From Value} (\leq) {To Value}</td>
<td>When a rule with a &quot;less or equal&quot; comparator is being evaluated</td>
<td>A rule is being evaluated by comparing that the first value is less than or equal to the second value.</td>
</tr>
</tbody>
</table>
Meaning

Triggered

Greater or Equal - Evaluate: {From Value} >= {To Value}
A rule is being evaluated by comparing that the first value is greater than or equal to the second value.

When a rule with a "greater or equal" comparator is being evaluated

Evaluating Atom Node: {Rule ID}
Indicates that an actual rule is being evaluated.

When a rule is being evaluated

Group Start: Evaluating And Node
When multiple rules are used, this indicates when the criteria of a pair of rules must both be "true."

When two rules are being evaluated with an "AND" clause

Group Start: Evaluating Or Node
When multiple rules are used, this indicates when the criteria of a pair of rules must be "true" for one of them.

When two rules are being evaluated with an "OR" clause

Group Start: Evaluating Not Node
This indicates to check for the opposite of what the criteria evaluates to.

When two rules are being evaluated with a "NOT" clause

Setting invite delay to: {Invite Delay}
If the criteria for a timer-based rule is not met, a delay is set to attempt to evaluate the rules again in the future when the criteria will have been met.

When a timer-based rule has not yet met the criteria

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.

Note: This isn’t applicable for mobile browsers, where chats open to the full page.

IN THIS SECTION:

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

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

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Number</td>
<td>The height in pixels of your custom chat window.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>

**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

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

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Number</td>
<td>The width in pixels of your custom chat window.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>

**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.

_IN THIS SECTION:

- **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.

Optionally, you can route chats from a specific button directly to the agent with the `userId` you specify. If the agent you specify is unavailable, you can specify whether to fall back to the button’s routing rules (`true`) or not (`false`).

**Syntax**

```java
void startChat(String buttonId, (optional) String userId, (optional) Boolean fallback)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonId</td>
<td>String</td>
<td>The ID of the chat button for which to request a chat in a new window.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>(Optional) userId</td>
<td>String</td>
<td>The Salesforce.com user ID of the agent to whom to directly route chats from the button.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>(Optional) fallback</td>
<td>Boolean</td>
<td>Specifies whether to fall back to the button’s routing rules (<code>true</code>) or not (<code>false</code>) if the agent with the specified <code>sfdcUserId</code> is unavailable.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

---

**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**

```java
void startChatWithWindow(String buttonId, String windowHeight, (optional) String userId, (optional) Boolean fallback)
```
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonId</td>
<td>String</td>
<td>The ID of the chat button for which to request a chat in a new window.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>windowName</td>
<td>String</td>
<td>The name of the window.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>(Optional) userId</td>
<td>String</td>
<td>The Salesforce user ID of the agent to whom to directly route chats from the button.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>(Optional) fallback</td>
<td>Boolean</td>
<td>Specifies whether to fall back to the button’s routing rules (true) or not (false) if the agent with the specified sfdcUserId is unavailable.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

### 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.

**IN THIS SECTION:**

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

- **setName**
  - Use the `setName` method to set 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. The Custom Detail is displayed to agents in the footer widget and in the Chat Details page in the Salesforce Console while the chat is active. Available in API versions 28.0 and later.

**Syntax**

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>String</td>
<td>The label for the custom detail—for example, &quot;Name&quot;.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td>The value of the custom detail—for example, &quot;John Doe&quot;.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>(Optional) displayToAgent</td>
<td>Boolean</td>
<td>Specifies whether to display the custom details that customers provide in a pre-chat form to the agent (true) or not (false).</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

IN THIS SECTION:

**addCustomDetail.doKnowledgeSearch**

Use the `knowledgeSearch` method to automatically search for Knowledge One articles based on criteria in a pre-chat form.

**Syntax**

```java
liveagent.addCustomDetail(String label, String value, (optional) Boolean displayToAgent).doKnowledgeSearch()
```

**setName**

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

**Syntax**

```java
setName(String name)
```
Find and Create Records Automatically with the Deployment API

Use the Deployment API to search for or create Salesforce records—like a case, contact, account, or lead—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.

IN THIS SECTION:

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityName</td>
<td>String</td>
<td>The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>
IN THIS SECTION:

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 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.

Note: To find the API name of a field for a standard object, see our API documentation. For non-standard objects, look at the field detail for the object under Setup.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldName</td>
<td>String</td>
<td>The API name of the field in the record EntityName to which to map the corresponding custom detail DetailName.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>DetailName</td>
<td>String</td>
<td>The value of the custom detail to map to the corresponding field FieldName.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>doFind</td>
<td>Boolean</td>
<td>Specifies whether to search for a record that contains the custom detail DetailName in the field FieldName (true) or not (false).</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>
### 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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TranscriptFieldName</td>
<td>String</td>
<td>The name of the field on the chat transcript record to which to save the ID of the record you found or created.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

### findOrCreate.showOnCreate

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

**Usage**

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

Available in API versions 29.0 and later.
Syntax

```java
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

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

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntityName</td>
<td>String</td>
<td>The type of record to which to link the child record you found or created.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>FieldName</td>
<td>String</td>
<td>The name of the API field in the record <code>EntityName</code> to which to save the ID of the child record you found or created.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

**Note:** To find the API name of a field for a standard object, see our API documentation. For non-standard objects, look at the field detail for the object under Setup.

**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.linkToEntity`
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.

**IN THIS SECTION:**

- **showWhenOnline**: Use the `showWhenOnline` method to specify what customers see when a particular button is online.
- **showWhenOffline**: Use the `showWhenOffline` method to specify what customers see when a particular button is offline.
- **addButtonEventHandler**: Use the `addButtonEventHandler` method to define a chat button’s behavior when certain events occur. Available in API versions 28.0 and later.

### `showWhenOnline`

**Usage**

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

**Syntax**

```java
void showWhenOnline(String buttonId, Object element, (optional) String userId)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonId</td>
<td>String</td>
<td>The ID of the chat button for which to display the specified element object when agents are online.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>
showWhenOffline

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

**Usage**

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

**Syntax**

```java
void showWhenOffline(String buttonId, Object element, (optional) String userId)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonId</td>
<td>String</td>
<td>The ID of the chat button for which to display the specified element object when no agents are available to chat.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>element</td>
<td>Object</td>
<td>The element to display when the specified button is offline.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>
### addButtonEventHandler

Use the `addButtonEventHandler` method to define a chat button’s behavior when certain events occur. Available in API versions 28.0 and later.

#### 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.

⚠️ **Note:** This method is available only for automated chat invitation buttons.

#### Syntax

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

#### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonId</td>
<td>String</td>
<td>The ID of the chat button for which to define the behavior when certain events occur.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>callback</td>
<td>function</td>
<td>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 21.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Event Type</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>BUTTON_AVAILABLE</td>
<td>liveagent.BUTTON_EVENT.BUTTON_AVAILABLE</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>BUTTON_UNAVAILABLE</td>
<td>liveagent.BUTTON_EVENT.BUTTON_UNAVAILABLE</td>
<td>Specifies the behavior of the button when no agents are available to chat.</td>
</tr>
</tbody>
</table>

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.

IN THIS SECTION:

- 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonId</td>
<td>String</td>
<td>The ID of the chat button for which to reject chats.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>

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.

Note: This functionality applies only to automated chat invitation buttons and doesn’t work for other chat buttons.
Available in API versions 28.0 and later.

Syntax

void addButtonEventHandler(String buttonId, Function callback)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonId</td>
<td>String</td>
<td>The ID of the chat button associated with the automated invitation for which to define the behavior when certain events occur.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>callback</td>
<td>function</td>
<td>The function to call when a particular event occurs. You must specify the invitation’s</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>
behavior for each of the required event types on page 23.

### 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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Event Type</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>BUTTON_AVAILABLE</td>
<td>liveagent.BUTTON_EVENT.BUTTON_AVAILABLE</td>
<td>Specifies the behavior of the automated invitation when the criteria are met for the invitation to appear on-screen.</td>
</tr>
<tr>
<td></td>
<td>BUTTON_UNAVAILABLE</td>
<td>liveagent.BUTTON_EVENT.BUTTON_UNAVAILABLE</td>
<td>Specifies the behavior of the automated invitation when no agents are available to chat.</td>
</tr>
<tr>
<td></td>
<td>BUTTON_ACCEPTED</td>
<td>liveagent.BUTTON_EVENT.BUTTON_ACCEPTED</td>
<td>Specifies the behavior of the automated invitation when a customer accepts the invitation.</td>
</tr>
<tr>
<td></td>
<td>BUTTON_REJECTED</td>
<td>liveagent.BUTTON_EVENT.BUTTON_REJECTED</td>
<td>Specifies the behavior of the automated invitation when a customer rejects the invitation.</td>
</tr>
</tbody>
</table>

**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

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

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>variableName</td>
<td>String</td>
<td>The name of the customized criteria for your custom sending rule.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
<tr>
<td>value</td>
<td>Object</td>
<td>The comparison value for your custom sending rule.</td>
<td>Available in API versions 28.0 and later.</td>
</tr>
</tbody>
</table>

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.

```xml
<apex:page>
    <div id="liveagent_invite_button_573x0000000001O" 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('573x0000000001O')">X</div>
        <div style="cursor: pointer; top: 42px; left: 65px; position: absolute; font-weight: bold; font-size: 16px;" onclick="liveagent.startChat('573x0000000001O')">Start Chat</div>
    </div>
    <script type='text/javascript' src='https://c.lasl1.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_573x00000000010').style.display = '';
                document.getElementById('liveagent_invite_button_573x00000000010').style.left = '300px';
                document.getElementById('liveagent_invite_button_573x00000000010').style.top = '200px';
            }
        }
    </script>
</apex:page>
```
if (e == liveagent.BUTTON_EVENT.BUTTON_UNAVAILABLE) {
    document.getElementById('liveagent_invite_button_573x0000000001O').style.display = 'none';
}
if (e == liveagent.BUTTON_EVENT.BUTTON_ACCEPTED) {
    document.getElementById('liveagent_invite_button_573x0000000001O').style.display = 'none';
}
if (e == liveagent.BUTTON_EVENT.BUTTON_REJECTED) {
    document.getElementById('liveagent_invite_button_573x0000000001O').style.display = 'none';
}
liveagent.addButtonEventHandler('573x0000000001O', 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:

![Chat Invitation](https://example.com/chat-invitation.png)

**Deployment API Code Sample**

Test and preview how the Deployment API can help you customize your deployments.

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

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

```xml
<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.lals1.saleforceliveagent.com/content/g/deployment.js'></script>
  <script type='text/javascript'>
    // An auto query that searches contacts whose email field exactly matches "john@acme.com"
    liveagent.addCustomDetail('Contact E-mail', 'john@acme.com');
    liveagent.findOrCreate('Contact').map('Email','Contact E-mail',true,true,false);
  
    // Conducts a Knowledge One search on the provided value; in this case,
    // searches Knowledge One articles for the term "Problems with my iPhone"
    liveagent.addCustomDetail('Case Subject', 'Problem with my iPhone').doKnowledgeSearch();

    // An auto query that searches contacts whose first name field matches "John Doe"
    liveagent.addCustomDetail('Contact Name', 'John Doe');
    liveagent.findOrCreate('Contact').map('FirstName','Contact Name',true,false,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.lals1.salesforceliveagent.com/chat', '572D0000000002R', '00DD0000000JXbY');
  </script>
</apex:page>
```
This deployment code 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 also use this information to customize the customer’s experience while they chat with the agent, such as including their first name in the chat window.

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.

IN THIS SECTION:

Pre-Chat Form Code Sample
Test and preview how pre-chat forms can work for your agents and customers.

Pre-Chat Form Code Sample

Test and preview how pre-chat forms can work for your agents and customers.

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

- Requests a visitor’s name and email address.
- Displays that information in the chat log.
- Enables mapping and auto-query, which triggers a search on the visitor’s email address when a support agent accepts the chat.
- If no email match is found, creates a new contact record with the visitor’s name and email address.
- Displays a drop-down list that enables visitors choose a different Live Chat button through which to route their chat request.

```html
<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>
</apex:page>
```
This code results in a pre-chat form that looks like this:
<table>
<thead>
<tr>
<th>Pre-chat Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
</tr>
<tr>
<td>Last Name</td>
</tr>
<tr>
<td>Email Address</td>
</tr>
<tr>
<td>Department: Customer Service</td>
</tr>
<tr>
<td>Request Chat</td>
</tr>
</tbody>
</table>
CHAPTER 6  Accessing Chat Details with the Pre-Chat API

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

IN THIS SECTION:

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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>details</td>
<td>Object</td>
<td>An object containing all of the custom details that were included in the pre-chat form using the preChatInit method.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

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

```json
{
    "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-bcbbc-433d-b822-a87c062e681c"
}
```

**detailCallback**

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

```javascript
function myCallback(details) {
    // Customer specific code
}
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>details</td>
<td>Specifies the actions to occur after the custom details are retrieved using the <code>preChatInit</code> method.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>
CHAPTER 7  Create 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.

IN THIS SECTION:

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 find or create a record and save it to the chat transcript associated with the chat.

findOrCreate.showOnCreate
Use the findOrCreate.showOnCreate method to find or create a record and automatically open it 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 agents for incoming chats in the widget and in the Details tab when they receive a chat request.

Creating Records Pre-Chat API Code Sample
Test and preview how to automatically create records when a customer completes a pre-chat form using this code sample.

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 that’s specified in the pre-chat form that the customer completes. 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.
Create Records Automatically with the Pre-Chat API

Syntax

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

Parameters

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

IN THIS SECTION:

- **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 value must exactly match the field value 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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityType</td>
<td>String</td>
<td>The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the API field to search for in existing records.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>find</td>
<td>Boolean</td>
<td>Specifies whether to search for existing records that contain the field fieldName (true) or not (false).</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

**Note:** To find the API name of a field for a standard object, see our API documentation. For non-standard objects, look at the field detail for the object under Setup.

**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`.

**findOrCreate.map.isExactMatch**

Use the `findOrCreate.map.isExactMatch` method to specify whether a field value must exactly match the field value 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 value 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**

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

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

**Note:** To find the API name of a field for a standard object, see our API documentation. For non-standard objects, look at the field detail for the object under Setup.

| find       | Boolean | Specifies whether to search for existing records that contain an exact match to the field fieldName (true) or not (false). | Available in API versions 29.0 and later. |

**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.

**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**

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>String</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The API name of the field to include in new records.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>create</td>
<td>Boolean</td>
<td>Specifies whether to create a new record that contains the field fieldName (true) or not (false).</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

**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.

**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.

---

**findOrCreate.saveToTranscript**

Use the findOrCreate.saveToTranscript method to find or create a record and save it 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**

```html
<input type="hidden" name="liveagent.prechat.findorcreate.saveToTranscript: String entityName" value="String transcriptFieldName" />
```
Create Records Automatically with the Pre-Chat API

**Parameters**

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

**Note:** To find the API name of a field for a standard object, see our API documentation. For non-standard objects, look at the field detail for the object under Setup.

**findOrCreate.showOnCreate**

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

**Usage**

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

Available in API versions 29.0 and later.

**Syntax**

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

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>String</td>
<td>The type of record to search for or create when an agent accepts a chat with a customer—for example, a contact record.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>show</td>
<td>Boolean</td>
<td>Specifies whether to display the record you created in a subtab in the Salesforce console (true) or not (false).</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>
**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. The `findOrCreate.linkToEntity` method can’t be used to populate fields on records that you create by using the `findOrCreate` API call. Instead, use the `findOrCreate.map` method to update field values on records.

**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**

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

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>String</td>
<td>The type of record to which to link the child record you found or created.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>parentEntityName</td>
<td>String</td>
<td>The type of parent record to link to the child record you found or created.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>fieldName</td>
<td>String</td>
<td>The name of the field in the record parentEntityName to which to save the ID of the child record you found or created.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

**findOrCreate.displayToAgent**

Use the `findOrCreate.displayToAgent` method to specify which pre-chat details will be displayed to agents for incoming chats in the widget and 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. Typically, this method is only used to hide particular custom details from the agent but setting its value to `false`.

Available in API versions 29.0 and later.
Syntax

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

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Available Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>detailName</td>
<td>String</td>
<td>The name of the detail to display to an agent when they receive a chat request.</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
<tr>
<td>display</td>
<td>Boolean</td>
<td>Specifies whether to display the custom detail to an agent in the chat notifications and Details tab (true) or not (false).</td>
<td>Available in API versions 29.0 and later.</td>
</tr>
</tbody>
</table>

**Note:** You only need to specify details for which `display` equals `false`. The method will not display details for which `display` equals `false`. If you don’t specify the value of the `display` parameter, the default value is set to `true`.

Creating Records Pre-Chat API Code Sample

Test and preview how to automatically create records when a customer completes a pre-chat form using this code sample.

The following code searches for and creates records when a customer completes a pre-chat form using the following methods:

- `findOrCreate.map`
- `findOrCreate.map.doFind`
- `findOrCreate.map.isExactMatch`
- `findOrCreate.map.doCreate`
- `findOrCreate.saveToTranscript`
- `findOrCreate.showOnCreate`
- `findOrCreate.linkToEntity`

```html
<form method="post" action="#">
  <label>First Name: </label> <input type='text' name='liveagent.prechat:ContactFirstName' />
  <br />
  <label>Last Name: </label> <input type='text' name='liveagent.prechat:ContactLastName' />
  <br />
  <label>Subject: </label> <input type='text' name='liveagent.prechat:CaseSubject' />
  <br />
  <input type='hidden' name='liveagent.prechat:CaseStatus' value='New' />
  <br />
  <input type='hidden' name='liveagent.prechat.findorcreate.map:Contact' value='FirstName,ContactFirstName;LastName,ContactLastName' />
  <br />
  <input type='hidden' name='liveagent.prechat.findorcreate.map.doFind:Contact' value='FirstName,true;LastName,true' />
</form>
```
Create Records Automatically with the Pre-Chat API

```html
<input type="hidden" name="liveagent.prechat.findorcreate.map.isExactMatch:Contact" value="FirstName,true;LastName,true" />
<input type="hidden" name="liveagent.prechat.findorcreate.map.doCreate:Contact" value="FirstName,true;LastName,true" />
<input type="hidden" name="liveagent.prechat.findorcreate.saveToTranscript:Contact" value="ContactId" />
<input type="hidden" name="liveagent.prechat.findorcreate.showOnCreate:Contact" value="true" />
<input type="hidden" name="liveagent.prechat.findorcreate.linkToEntity:Contact" value="Case,ContactId" />
<input type="hidden" name="liveagent.prechat.findorcreate.map:Case" value="Subject,CaseSubject;Status,CaseStatus" />
<input type="hidden" name="liveagent.prechat.findorcreate.map.doCreate:Case" value="Subject,true;Status,true" />
<input type="submit" value="Submit">
</form>
```
CHAPTER 8 Customize Chat Windows with Visualforce

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

We recommend that you don’t link to Salesforce CSS stylesheets when you customize your chat window. They aren’t versioned and can change without notice. Instead, we recommend that you use Visualforce components that mimic Salesforce styles instead of directly referencing the stylesheets. That way, you’re always in control of how your chat window looks. See Using Styles from Salesforce Stylesheets to lean how to disable our stylesheets.

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

IN THIS SECTION:
- Live Agent Visualforce Components
  Use Visualforce components to customize the appearance and behavior 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.

Live Agent Visualforce Components

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

Live Agent includes the following customizable Visualforce components. These components are placed on your Visualforce form to make certain functionality available and to customize the appearance of the chat window.

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>liveAgent:clientChat</td>
<td>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.</td>
</tr>
<tr>
<td>liveAgent:clientChatAlertMessage</td>
<td>The area in a Live Agent chat window that displays system alert messages (such as “You have been disconnected”).</td>
</tr>
<tr>
<td>liveAgent:clientChatCancelButton</td>
<td>The button within a Live Agent chat window when the chat is in a waiting state that allows the visitor to cancel the chat. Must be used within liveAgent:clientChat.</td>
</tr>
<tr>
<td>liveAgent:clientChatEndButton</td>
<td>The button within a Live Agent chat window that a visitor clicks to end a chat session. Must be used within liveAgent:clientChat.</td>
</tr>
<tr>
<td>liveAgent:clientChatFileTransfer</td>
<td>The file upload area in a Live Agent chat window where a visitor can send a file to an agent. Must be used within liveAgent:clientChat.</td>
</tr>
<tr>
<td>Component Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>liveAgent:clientChatInput</td>
<td>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.</td>
</tr>
<tr>
<td>liveAgent:clientChatLog</td>
<td>The area in a Live Agent chat window that displays the chat conversation to a visitor. Must be used within liveAgent:clientChat. Each chat window can have only one chat log.</td>
</tr>
<tr>
<td>liveAgent:clientChatLogAlertMessage</td>
<td>The area in a Live Agent chat window that displays the idle time-out alert (customer warning) to a visitor.</td>
</tr>
<tr>
<td>liveAgent:clientChatMessages</td>
<td>The area in a Live Agent chat window that displays system status messages, such as &quot;Chat session has been disconnected.&quot; Must be used within liveAgent:clientChat. Each chat window can have only one message area.</td>
</tr>
<tr>
<td>liveAgent:clientChatQueuePosition</td>
<td>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.</td>
</tr>
<tr>
<td>liveAgent:clientChatSaveButton</td>
<td>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.</td>
</tr>
<tr>
<td>liveAgent:clientChatSendButton</td>
<td>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.</td>
</tr>
<tr>
<td>liveAgent:clientChatStatusMessage</td>
<td>The area in a Live Agent chat window that displays system status messages (such as &quot;You are being reconnected&quot;).</td>
</tr>
</tbody>
</table>

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:
Customize Chat Windows with Visualforce

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

```apex
<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>

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

![Chat window](image)
CHAPTER 9  Post-Chat Pages Overview

Post-chat pages let share information with customers at the end of a chat session. For example, you can direct your customers to another Web page after they complete a chat with an agent, or forward them to a survey about their chat experience.

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.

IN THIS SECTION:
Post-Chat Pages Code Sample

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

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.

```xml
<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="{$CurrentPage.parameters.lastVisitedPage}" /><br/>
    Original Referrer:
    <apex:outputText value="{$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/>
</apex:page>
```
The code above results in a post-chat page that looks like this:

```
Post Chat Page Request Time:1377692746441
Start Time:1377692753064
Button Id:573x0000000000
Deployment Id:572x0000000000
Last Visited Page:http://liveagentav182.localhost.force.com:8080/live
Original Referer:
Latitude:37.764496
Longitude:122.4294
City:San Francisco
Region:CA
Country:United States
Organization:SALESFORCE.COM
Transcript:Emily W.: Thanks for contacting support. How can I help you? You: I'm having trouble with my VX6 unit. It keeps shutting off when I download something to it. Emily W.: OK, it looks like you need to upgrade your software to the latest version, which should fix the problem. You: An OK, thanks a lot for your help.
Disconnected By:
Chat Key:8b3d051-08d8-485c-aaf0d7e8615e
Chat Details:{"prechatDetails":null,"geoLocation":{"countryCode":"US","countryName":"United States","longitude":-122.4294,"organization":"SALESFORCE.COM","latitude":37.764496,"region":"CA","city":"San Francisco"},"visitorId":"28d9355a-828a-4c81-9601-2e8c8ac610ce","customDetails":null,"transcriptFields":null,"displayToAgent":null,"entityMaps":null,"agent":null,"userId":"005x000001JGr5","agentName":"Emily W.","transfer":null}
Error:
```
Set Up Direct-to-Agent Chat Routing with the Deployment API

By editing a few parameters in Live Agent’s Deployment API, you can route all chats that originate from a specific button or invite to a specific agent. If that’s not available, the chat can be routed to a fallback queue.

For Example

Best Tech is a large, enterprise-level middleware company. Specialized Data Solutions, Inc. is one of Best Tech’s highest-profile accounts with an extremely complex, customized Best Tech implementation.

To ensure the highest level of customer service for their most important client, Best Tech implemented a new authenticated customer service portal specifically for their customers at Specialized Data. This portal features its own Live Agent deployment and chat button. Through this portal, Specialized Data’s employees can chat with a customer service agent that knows the intricacies of the company’s Best Tech implementation.

Because Specialized Data’s implementation of Best Tech is so unique and complex, Best Tech hired a high-performing customer service representative who is devoted exclusively to troubleshooting Specialized Data’s issues. Best Tech needed to make sure that all of the chats initiated through Specialized Data’s portal were routed to this agent. To make this a reality, Best Tech customized the deployment on the portal page to route chats directly to this agent, rather than routing those chats to Best Tech’s overall agent queue.

IN THIS SECTION:

Direct-to-Agent Routing with the Deployment API
You can set up direct-to-agent routing using the Deployment API which enables you to route chats from a specific button to a single agent.

Fallback Routing in Pre-Chat Forms
You can set up fallback routing rules in pre-chat forms in case the agent you specify for direct-to-agent routing isn’t available when a chat is received.

Direct-to-Agent Routing with the Deployment API
You can set up direct-to-agent routing using the Deployment API which enables you to route chats from a specific button to a single agent.

Use the following Deployment API methods to set up direct-to-agent routing with fallback routing enabled in case the agent isn’t available for chat.

- `startChat`
- `startChatWithWindow`
- `showWhenOnline`
Fallback Routing in Pre-Chat Forms

You can set up fallback routing rules in pre-chat forms in case the agent you specify for direct-to-agent routing isn’t available when a chat is received.

What if you set up direct-to-agent routing, but the agent you specified to receive the chats isn’t available? If the agent is offline, those chats might be lost.

Luckily, if your organization uses pre-chat forms to gather customer information, you can set up fallback routing options for a button that uses direct-to-agent routing.

The code sample below demonstrates how to route chats using another button’s default routing rules if the agent assigned to that button isn’t available. Let’s take a look at this section of code:

```html
<h1>Pre-chat Form</h1>
<form method='post' id='prechatForm'>
  Name: <input type='text' name='liveagent.prechat.name' id='prechat_field' />
  Email Address: <input type='text' name='liveagent.prechat:Email' />
  Department: <select name="liveagent.prechat.buttons">
    <!-- Values are LiveChatButton and/or User IDs. -->
    <option value="573xx000000000001">Route through button 573xx000000000001</option>
    <option value="005xx000001Sv1m">Route to agent 005xx000001Sv1m</option>
    <option value="005xx000001Sv1m_573xx000000000001">Route to agent 005xx000001Sv1m with Fallback to button 573xx000000000001</option>
  </select>
</form>
```

In this section, we specify that chats originating from the button should be routed to an agent with agent ID 005xx000001Sv1m. If that agent isn’t available, however, incoming chats are routed based on the default routing rules for the button with button ID 573xx000000000001.

**Code Sample**

Below is the full code sample for the pre-chat form with fallback routing rules enabled. It represents a pre-chat form that:

- Requests a visitor’s name and email address.
- Displays that information in the chat log and in the chat request window.
- If a contact record already exists that contains the customer’s name and email address, that record appears in another tab in the Salesforce console. If not, a new Contact record is created and populated with the customer’s information.
- Displays a drop-down list that lets visitors choose a different Live Chat button through which to route their chat request.
- Routes chats directly to a specific agent, or, if that agent is unavailable, routes those chats based on the button’s default routing rules.

```html
<html>
<body>
  <!-- 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',
```
Set Up Direct-to-Agent Chat Routing with the Deployment API

Fallback Routing in Pre-Chat Forms

```html
<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 and/or User IDs. -->
    <option value="573xx00000000001">Route through button 573xx00000000001</option>
    <option value="005xx000001Sv1m">Route to agent 005xx000001Sv1m</option>
    <option value="005xx000001Sv1m_573xx00000000001">Route to agent 005xx000001Sv1m with Fallback to button 573xx00000000001</option>
    otherwise Technical Support</option>
  </select><br />
  <input type='submit' value='Request Chat' id='prechat_submit'/>
</form>
```

```css
p {font-weight: bolder}
```

```
</body>
</html>
```