Engage Customers with Conversational Commerce

Salesforce, Spring '21

'21

@salesforcedocs

Last updated: December 16, 2020

CONTENTS

ENGAGE CUSTOMERS WITH CONVERSATIONAL COMMERCE

Provide richer customer service and drive sales when you implement live agent chat or chatbots within Commerce Cloud. Build brand loyalty, check order status, and offer extra assistance at the hands of service agents.

Get Started

Explore system architecture related to this solution.

- B2C Industry Blueprint
- B2C Reference Architecture
- B2C Solution Architectures

Take Trailhead modules related to this solution.

- Salesforce Solution Kits: Quick Look
- Customer 360 Guide for Retail: Quick Look
- Customer 360 Guides: Quick Look

This solution kit helps you:

- Engage customers at key points of their shopping experience.
- Give customers timely self-service help.
- Prevent cart abandonment via proactive chat.
- Increase the productivity and service quality of customer service agents.
- Exceed the customer service expectations of your customers.

Required Products

- Commerce Cloud (SFRA or SiteGenesis)
- Service Cloud

Recommended

Service Cloud Connector for B2C Commerce

Other Supported Use Cases

This solution kit provides guidance on how to implement live agent chat within Commerce Cloud. The kit also explains how to support other use cases that the Service Cloud connector facilitates, including:

- Letting agents order on behalf of storefront shoppers.
- Empowering shoppers to inquire about order status.

While the Conversational Commerce use cases in this solution kit don't require the Service Cloud connector, it gives agents extended capabilities and provides integration patterns that support new service-driven use cases.

Review these recommendations for the two use cases: Live Agent Chat and Chatbot.

Table 1: Live Agent vs Chatbot

	Live Agent Chat (Human)	Chatbot (Automated)
Required Setup	Send Pre-Chat Variables	Send Pre-Chat Variables
Before Placing Order		
Add Discount to Basket (Coupon, Order Discount, Shipping Discount)	Via Order on Behalf (OOBO)	Via APIs
After Placing Order		
View Latest Order Status	Via Order Object in Core	Query Order Object in Core
Cancel Order Within Forgiveness Window	Via SFCC	Via APIs
Change the Order	Custom Implementation with OMS	Transfer to Agent
Answer Common Questions	Via Knowledge	Query Knowledge

Implement This Solution

Solution Architecture

Discover how use cases map to solutions with a visual illustration. See how recommended products integrate with back end systems and how data is passed between them when you engage customers with conversational commerce.

Conversational Commerce Solution Workflow

Learn how data flows through the configurations to engage customers with conversational commerce.

Design Considerations

Keep these design considerations in mind when you engage customers with conversational commerce.

Connector

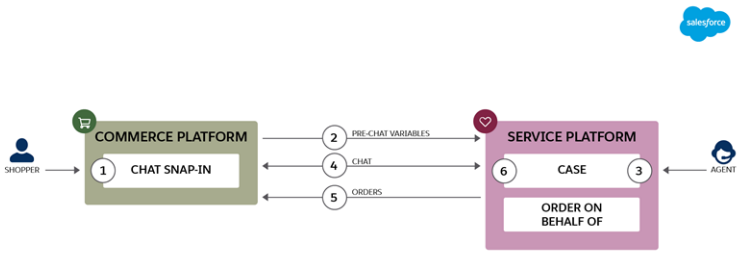
Connectors are developer enablement frameworks that accelerate cross-cloud integration by providing code, configuration, and implementation patterns. Use the Salesforce B2C Commerce to Service Cloud Connector to engage customers with conversational commerce.

Configurations

Use these configurations to engage customers with conversational commerce.

Solution Architecture

Discover how use cases map to solutions with a visual illustration. See how recommended products integrate with back end systems and how data is passed between them when you engage customers with conversational commerce.



Related Content

Review this solution's use case and purpose.

- Engage Customers with Conversational Commerce on page 1

Take the next steps in this implementation.

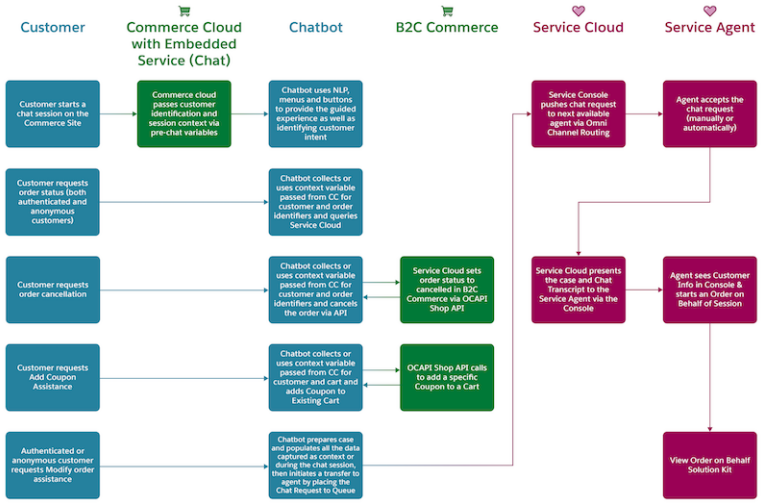
- Conversational Commerce Solution Workflow
- Design Considerations
- Connector
- Configurations

Conversational Commerce Solution Workflow

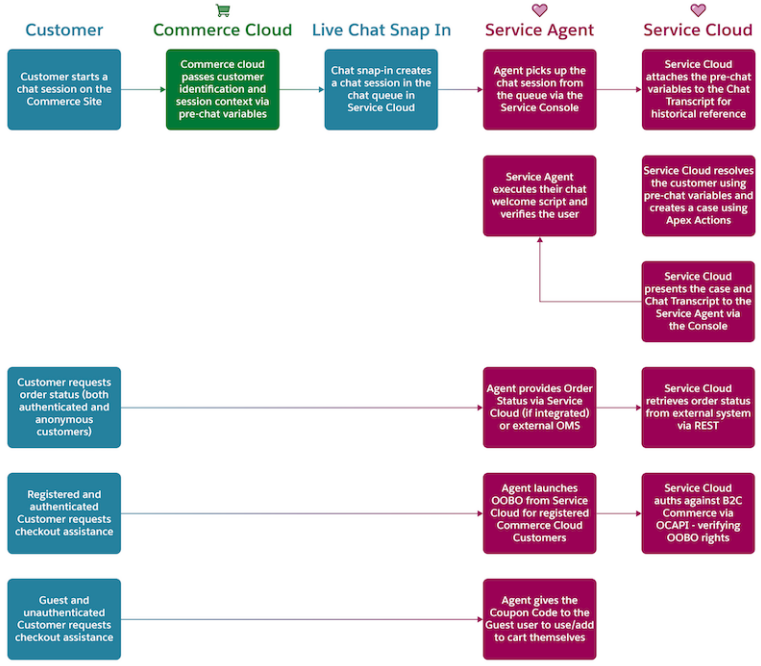
Learn how data flows through the configurations to engage customers with conversational commerce.

Workflow

Salesforce Chat



Chatbot



Related Content

Review earlier steps in this solution.

- Solution Architecture

Take the next steps in this implementation.

- Design Considerations
- Connector
- Configurations

Design Considerations

Keep these design considerations in mind when you engage customers with conversational commerce.

Identify and Resolve Customers

- Get guidance on how to customize the pre-chat form for performing more advanced activities in the Service Cloud Chat Developer Guide. These activities include customer identification and automated record creation in Service Cloud.
- If you integrate Commerce and Service Clouds via the Service Cloud Connector, every shopper who registered or placed an order via Guest Checkout has a corresponding person account.
- Show Service Cloud how to identify and resolve the shopper's person account when you extend the pre-chat form to use the shopper's Commerce Cloud and Service Cloud identifiers. To perform this resolution, use the profile identifiers stored in the customer's Commerce Cloud profile.
- Instruct Service Cloud to create a person account representing the chat customer (if one was resolved) when you extend the pre-chat form. Specify the RecordTypeId representing a person account in the pre-chat form.

Provide Order Status and Cancel Orders for Customers

- To provide Service Agents with order status and let them cancel orders on behalf of customers, see the View Order History and Cancel Orders solution kit, which explains how to use the Service Cloud Connector.
- Give customers order status or cancel orders on their behalf. A service agent performs both activities from within Service Cloud using these Service Cloud Connector features.
- To process storefront orders and place them through a fulfillment workflow, Commerce Cloud customers often use an order management system (OMS). In this scenario, integrate Service Cloud with the OMS to interact with orders.
- Review the implementation considerations in the View Order History and Cancel Orders solution kit. It supports order modification use-cases and guidance on integration with external order management systems.

Launch Proactive Chat

- Learn how to enable proactive chat invitations in the Web Chat Basics Trailhead module.
- Automate chat invitations by modifying the chat trigger to fire based on certain criteria. This includes whether a customer remains on a web page for more than a specified amount of time or the activities a customer performs on a web page.
- To set variable triggers that support automated chat invitations, use JavaScript code. Monitor storefront session length time and create a chat invitation if a shopper has items in their cart but has not entered checkout in [x] minutes when you customize the automated chat triggers.
- Prevent shopping cart abandonment by triggering a chat invitation if a shopper enters the checkout process and does not complete the checkout in [x] minutes or spends more than [x] minutes of a specific section of checkout.

Note: Use your web analytics reporting to identify pages to target as proactive chat invitation points. Analytics reports identify pages to target, estimate shopper session length, and record page-visit durations of shoppers.

- Monitor customer behavior in checkout for activities that lead to abandonment (for example, clicks to the storefront home-page icon or footer links). To pre-empt checkout abandonment, initiate a chat invitation.
- Monitor the total value of the customer's shopping cart and customize the automated chat triggers to initiate a chat invitation if the cart value exceeds a threshold value and the customer does not enter checkout within [x] minutes.
- Learn how to customize a chat invitation and create custom chat events to initiate chat invitations in the Embedded Service for Web Developer Guide.

OOBO for Authenticated Customers

- To include adding products to cart and applying coupons or discounts to a customer's basket, use the Service Cloud Connector's order-on-behalf-of capabilities.

- Order-on-behalf-of via the Service Cloud Connector is available for person accounts that represent registered Commerce Cloud customer profiles.
- To be eligible for order-on-behalf-of assistance by service agents, Commerce Cloud customers must authenticate before launching a chat session.
- To include logic to resolve customers using their Commerce Cloud profile identifiers, extend the pre-chat form. To create person accounts using the Commerce Cloud customer identifiers included in the pre-chat variables, extend the form.
- Service agents can click the Launch Shopping Cart button, which the Service Cloud Connector provides from the person account that represents the chat user. This button opens an agent shopping session with access to the chat user's shopping cart.
- Through this session, agents can adjust the customer's cart contents, apply coupons or promotions, or assist the customer through the shopping and checkout processes.
- To empower service agents to provide agent-only discounts via coupons, customize the shopping experience. This customization can include line item-level price breaks as part of their service capabilities to customers.
- Any changes that the agent makes to the customer's cart are visible to the customer with their next page request via the Commerce Cloud storefront.

OOBO for Anonymous Customers

- The Service Cloud Connector order-on-behalf-of capability is eligible for person accounts which represent registered Commerce Cloud customers. OCAPI supports the creation of agent-driven shopping sessions for registered customers.
- Create agent-driven anonymous shopping sessions by extending the storefront and integration with Service Cloud. OCAPI doesn't support creating agent-driven anonymous shopping sessions.
- Enable OOBO for anonymous Commerce Cloud shopping sessions by extending the Service Cloud Connector's authentication logic to instruct Commerce Cloud to create an agent-driven anonymous shopping session.
- The Commerce Cloud script API AgentUserMgr class contains methods to create an agent shopping session and assign a registered Commerce Cloud customer to it. Extend your storefront to support this capability and initiate it from Service Cloud.

Chatbot Design

- To provide the best customer experience, build a contextual chatbot based on customer profile and shopping cart data. Bot Builder supports rules that send customers through different dialog flows for the personalized experience.
- Design the voice and tone of the bot to match the brand. The voice and tone provide consistent branding across the webstore and the chat channel.
- Use menu- and button-based navigation and NLP to identify customer intent. In the initial rollout of a chatbot, it's critical to have a menu that lists the bot capability. This menu helps set the right expectations for the customer, describes what the bot can do, and indicates when they transfer to a live agent.
- Build a transfer-to-agent flow in the bot as a fallback for anything the bot can't handle. Though technically not required, we recommend that you implement Agent Chat before launching a chatbot.
- Chatbot has unlimited capacity so it can run 24-7 if configured to do so. If agents are not working 24-7, build an offline support flow so that the chat session doesn't encounter a dead end.
- An offline support flow gathers more customer information, such as email, mobile number, and case creation. It can also notify customers that an agent will contact them during normal business hours.
- Chat may not be not responsive when bot actions include long running processes that depend on external integrations. Consider multiple timeout limits when building integrations.
- Bot actions have a 10-second system timeout. Salesforce Platform allows for as many as 10 concurrent long-running processes. A long-running process is one that runs more than 5 seconds on the platform.
- Add a timeout of 3 seconds for each API integration to prevent bot actions from becoming a long-running process. This timeout leaves 2 seconds for the rest of the bot actions in the same running context.

Bot Interactions with Storefront Shopping Carts Via OCAPI

Authenticate agent-driven shopping sessions using the customer/auth Shop API resource and connect them to registered Commerce Cloud customers via the customer/{customerId}/auth resource.

Interact with anonymous and registered customer shopping carts by accessing the customer's session through the sessions SHOP API resource. This resource uses the dwsid and dwsecuretoken values to identify the customer's session.

Use the products SHOP API resource to add or remove products from the customer's shopping cart. Changes made to the cart are visible on the customer's next page request.

Convert the customer's basket to an agent basket so that service agents can apply custom line-item, order, and shipping discounts.

Use the basket price_adjustments SHOP API resource to apply custom line-item, order, and shipping discounts to the agent cart. These resources support custom discounts at each of these basket levels.

Assign the customer to the agent shopping cart to complete the bot interaction (required for a new agent cart).

Convert the agent basket to a storefront basket so that the agent's basket adjustments appear for the original storefront shopper. Do this after assigning the customer to the agent shopping cart.

Note: Download this Postman Collection and its Environment Template for working examples of the cart interactions outlined in this section of the solution kit. The supported scenarios include interactions with anonymous, agent, and registered carts.

Related Content

Review earlier steps in this solution.

- Solution Architecture
- Conversational Commerce Solution Workflow

Take the next steps in this implementation.

- Connector
- Configurations

See Also

Commerce Cloud

- Commerce Cloud Supported Cookies
- OCAPI Customer Authentication
- OCAPI OAuth Authentication
- OCAPI Usage Guidance
- Open Commerce REST API: OCAPI Data API
- Open Commerce REST API: OCAPI Shop API

Salesforce Chat

- Available Chat Features in Embedded Chat
- Customize the Branding and Appearance of Your Chat Window
- Customize the Offline Support Form
- Customize the Pre-Chat Form
- Service Cloud Channels

- Service Cloud Chat Developer Guide
- Set Up and Customize an Automated Invitation
- Set Up and Use Quick Text
- Set Up Chat with a Guided Setup Flow
- Test Embedded Chat

Einstein Chatbots

- Add Prebuilt Dialogs and Customize Option Menu
- Create a Dialog, Sub-Menu, and Dialog Group
- Einstein Bots and Einstein Bots Cookbook
- Learn About Einstein Bots
- Learn the Prerequisites and Enable Einstein chatbots
- Plan Your Bot Content
- Prepare for Einstein Bots
- Set Up an Einstein Bot
- Understand Why chatbots Matter to the Contacts Center
- Write Bot Conversations Using Variables and Entities

Connector

Connectors are developer enablement frameworks that accelerate cross-cloud integration by providing code, configuration, and implementation patterns. Use the Salesforce B2C Commerce to Service Cloud Connector to engage customers with conversational commerce.

The Salesforce B2C Commerce to Service Cloud Connector is a Salesforce Labs project that facilitates the integration between Salesforce B2C Commerce Cloud and Service Cloud. The connector provides a framework to integrate the clouds by using public REST APIs to share and sync data.

Before implementing the connector, sign into GitHub and download the Service Cloud Connector. Follow the installation instructions available from the repository ReadMe.md file. Are you a Commerce Cloud customer or partner and don't have access to the GitHub repository? If you have access to Xchange, you can get access to Github via this XChange article.

Note: For access to Salesforce B2C Commerce XChange content, talk to your Success Manager.

General Information About Connectors

- Connectors are developer-enablement frameworks that accelerate cross-cloud integration by providing code, configuration, and implementation patterns.
- Connectors support a core set of use cases that you can extend to support other customer-driven use cases.
- Connectors require customization and configuration in Service Cloud and Commerce Cloud. The Commerce Cloud storefront requires customization as part of the connector integration.
- Implementation and validation require operational and administrative experience with Service Cloud.
- Plan your connector implementation as you would any other B2C Commerce Cloud feature by collecting requirements, capturing work tasks, and making task estimates.

What Your Company Can Do with This Connector

- Accelerate integration time to market for Commerce Cloud and Service Cloud.

- Get a shared view of customers and order data between Commerce and Service Clouds.
- Provide self-service through automated case creation from the storefront.
- Convert service interactions into cross-sell and up-sell opportunities through order on behalf of during agent-to-customer conversations.
- Support real-time, peer-to-peer data synchronization for customer, order, and case data between Commerce Cloud and Service Cloud.
- Customize the Commerce Cloud and Service Cloud experiences by extending your use cases and implementing new ones.

Functionality Considerations

The Service Cloud Connector:

- Supports Person Accounts as the customer model in Service Cloud.
- Does not natively support accounts and contacts, households, or multi-brand customer models.
- Enables the synchronization of registered Commerce Cloud customer profile and address book data with associated Service Cloud Person Accounts.
- Provides a collection of Lightning and Visualforce components that display customer and order information from within Service Cloud.
- Enables OOBO functionality for service agents from the customer and case detail displays in Service Cloud.
- Enables data synchronization between Commerce Cloud and Service Cloud through REST Services exposed and used by the connector.
- Extends the storefront to enable data synchronization with Service Cloud at strategic transaction points (customer registration, profile updates, order placement, and case creation).
- Supports the persistence of Commerce Cloud order line item, shipment, and payment information in Service Cloud.

Related Content

Review earlier steps in this solution.

- Solution Architecture
- Conversational Commerce Solution Workflow
- Design Considerations

Take the next steps in this implementation.

- Configurations

Configurations

Use these configurations to engage customers with conversational commerce.

These configurations help you set up live agent chat in Service Cloud and enable the chat client in Commerce Cloud for both Salesforce Chat and Einstein Bots.

Configure Service Cloud to Support Salesforce Chat

Get an introduction to configuring chat on the Salesforce platform when you take the Web Chat Basics module in Trailhead.

Completing this module gives you a testable instance of live agent chat. To implement Salesforce Chat via your Commerce Cloud storefront, use the test snap-in chat page that you create when you work through this module.

Configure Einstein Chatbots to Support Self-Service

Trailhead contains several modules which educate developers on the benefits of artificial intelligence (AI) in customer service and how to configure, create, and customize Einstein chatbots.

Completing this module gives you a testable Einstein chatbot powered by your Service Cloud environment.

Define the Shopper Identification Properties

Both Salesforce Chat and Einstein Chatbots require customer identification through the chat experience. Define the Commerce Cloud shopper identification properties captured via pre-chat variables. Use those properties to identify the customer, their session and basket, or the storefront page where they launched the chat experience.

Commerce Cloud manages the shopper's storefront with cookie names and values, which represent anonymous shopper properties. To identify individual shopper sessions, obtain these properties from the customer's cookie values set by the Commerce Cloud's storefront.

Use the customer's session identifiers to inspect their shopping session via Commerce Cloud's Open Commerce API (OCAPI). Obtain a session-grant via OCAPI using these dwsid and dwsecuretoken cookie values.

- DWSID - A cookie value that identifies the browser session.
- DWSTID - The site-specific identifier associates the dwsecuretoken or dwanonymous cookie value to the current storefront.
- DWSTValue - The token value for the dwsecuretoken or dwanonymous cookie. The dwsid cookie identifies and secures the customer's session through https.

Note: If you don't have dwsecuretoken in your browser. use the dwanonymous cookie for the DWSTID and DWSTValue properties. Otherwise, to authenticate against the shopping session, always use the dwsecuretoken cookie.

Authenticated Shopper Properties represent the internal identifiers describing the customer's registered Commerce Cloud customer and profile system objects. These properties exist as attributes on both objects within Commerce Cloud.

Note: To interact with and on behalf of the customer via both cloud's REST APIs, Commerce Cloud and Service Cloud use these customer profile identifiers.

- CustomerNo - The storefront-assigned customer number that represents the shopper.
- CustomerID - The internal Commerce Cloud identifier representing the shopper's customer profile.
- SFSC AccountID - The AccountID mapped to the Service Cloud PersonAccount associated with the Commerce Cloud customer profile.
- SFSC ContactID - The PersonContactID mapped to the Service Cloud PersonAccount associated with the Commerce Cloud customer profile.

Shopper Session Properties represent the shopping session properties describing the session context, duration, and page from which the chat launched. These properties calculate at the time the pre-chat form renders.

Note: To understand the context of the user and their shopping experience, use these shopper session properties.

- IsLoggedIn - Indicates whether the shopper was logged and authenticated when the live agent chat session launched.
- PagePipeline - Describes the current storefront pipeline from which the live agent chat session launched.
- PageUrl - Represents the customer-facing storefront page URL from which the live agent chat session launched.
- BasketID - Represents the internal Commerce Cloud platform identifier for the customer's basket.
- TimeInSession - Represents the amount of time the shopper is in their Commerce Cloud storefront session.

Extend the Service Cloud Chat Transcript Object

Include fields representing the defined Commerce Cloud shopper identifiers when you extend the ChatTranscript object in Service Cloud. To capture the shopper identifiers for each submitted Commerce Cloud pre-chat variable, use ChatTranscript fields.

- From Service Cloud Setup, open the Object Manager.

- Find the Chat Transcript and select the Fields and Relationships menu option.
- Add each of the identified shopper identity properties to the ChatTranscript object as custom fields.
- Note the field labels and field names that you created on the ChatTranscript object. Map these fields to the pre-chat variables through their API names.

Extend the Chatbot Context to Include Commerce Cloud Variables

Include chat transcript fields representing the defined Commerce Cloud shopper identifiers when you extend the list of context variables in Bot Builder. For detailed steps, see Get Web Context in the Einstein Bot Cookbook.

- Download the Bot Metadata in XML.
- Add the defined shopper identifier fields to the <contextVariables> section.
- Update the Bot Metadata with modified XML.

Implement Order Actions in Bots

Bot actions build dialogs in Bot Builder for the conversation experience. They also return the query results or execute the command from the bot conversation with customers. Learn about the two types of Order Actions from an "on/off" platform perspective.

- To query synchronized data on core (such as checking order status, building Flow, and SOQL query via Apex), return the result to the dialog.
- To read off platform data or perform any update, use OCAPI API. To consume OCAPI REST endpoints, for example, Cancel an Order, create Apex Bot Actions.

Extend Commerce Cloud to Manage the Chat Experience

Implement the Salesforce chat experience via a Commerce Cloud storefront. To manage the availability of the chat experience in Commerce Cloud, put controls in place.

Note: To manage the live agent chat configuration values and control switches, use Custom Site Preferences and preference groups.

- To organize all site preferences for managing the live agent chat experience, create a site preference group named "Chat Configuration.".
- To enable or hide the live agent pre-chat web form, create a site preference that manages the rendering of the live agent chat experience via the storefront.
- To maintain a list of storefront pages that exclude chat, create a site preference.

Extend the Storefront to Render the Chat Experience

With the previous steps finished, you can now render a branded chat experience when you extend the storefront from within the Commerce Cloud storefront.

- Create a Commerce Cloud script that collects pre-chat variables by inspecting a customer's session and profile before rendering the pre-chat snap-in.
- Create site preferences that abstract configuration settings of the snap-in and Service Cloud environment. These preferences enable environment-specific configurations instead of sharing one configuration among all environments.
- To render the pre-chat form variables, create a Commerce Cloud-rendering template. As a starting point, use the test pre-chat snap-in page.
- Implement the chat snap-in as a javascript file that populates the pre-chat variables on page-load. To implement this pattern, publish the pre-chat variables via the rendering template and invoke chat on page initialization.
- Embed any custom styling for branding the pre-chat snap-in in an external .css file. Avoid rendering style-sheet definitions from within the rendering template.
- To render the pre-chat variables and configuration settings, extend the storefront footer template to include the logic. Invoke the snap-in initializer when the page initializes.

- Confirm that the Commerce Cloud platform does not cache the pre-chat snap-in rendering template. Avoid caching this template because pre-chat variables are shopper-specific.
- Layer conditional logic that prevents the pre-chat snap-in from rendering within the storefront based on the configuration of live agent-specific custom site preferences.

Extend Agent Console to Include Starting Point for Order on Behalf Flow

- Retrieve customer identification credentials from the Chat Transcript when you customize the SCCShoppingCartComponent in the Service Cloud Connector.
- Evaluate whether the service agent is in the chat console when you extend the ShoppingCartAuraController. Retrieve the customer's customer number or session cookie identifiers via an Apex query.
- Create an agent-driven shopping session for registered Commerce Cloud customers that launch chat when you use the Service Cloud Connector's agent authentication logic.
- Commerce Cloud's Open Commerce REST API (OCAPI) does not support creating agent-driven shopping sessions for anonymous Commerce Cloud customers. Fulfill this use case using a storefront experience customization.

Test the Chat Experience

Verify that the chat experience works as designed across Commerce Cloud and Service Cloud.

- Get an agent who belongs to the chat destination queue to log in from the Service Console and set the Omni status to available.
- Watch the embedded chat button in Commerce Cloud change from "not available" to "available." This change verifies that the correct chat button is set on the store front.
- Verify that clicking the chat button launches the pre-chat form and a chat session.
- Disable the rendering of the pre-chat form within Commerce Cloud using the chat site preferences, and verify that it does not appear in the footer. To continue the test, re-enable the rendering of the pre-chat form.
- Verify that the agent receives the Omni notification as the chat request enters the queue. Confirm that there aren't many other tests running that would take away the agent's time.
- Verify that a chat session appears in the Service Console as the agent accepts the chat request.
- Within Service Cloud, verify that the anonymous and authenticated shopper identification properties appear on the page layout and chat transcript for each accepted chat session.
- Within Service Cloud, verify that each chat transcript contains the variables included in the pre-chat form. Use these variables to interact with the customer's session.
- Verify that agents can launch the OOBO process starting in Service Cloud.

Test the Bot Experience Across Commerce Cloud and Service Cloud

Always start the test by disabling the bot. Follow the previous section's steps to implement live agent chat as part of the overall bot solution.

- Enable the bot and deploy it to the chat channel. Verify that the Embedded Chat button in Commerce Cloud is still available, even if the agents are offline. This confirms that the bot deploys from the correct chat button with unlimited capacity.
- Verify that clicking the chat button launches the pre-chat form in Commerce Cloud and a chat session between the customer and bot.
- Verify that the bot receives the correct context data by executing the entire end to end test or displaying the context value in a temporary dialog.
- Verify all bot actions by executing entire order flows. Verify that the integration works by assessing the return value from OCAPI calls. Verify that the order status changed in Commerce Cloud for an order update.
- Verify the "Bot to Agent" transfer use case with an agent who logs in to Service Cloud Omni Channel. To continue the chat session with the customer, have the agent accept the chat transfer request.

- Verify that agents can review previous conversations between the bot and customer. Verify that all other data points that bots capture are correctly written to the Transcript or Case fields. Make these fields available to agents.

Related Content

Review earlier steps in this solution.

- Solution Architecture
- Conversational Commerce Solution Workflow
- Design Considerations
- Connector