# Omni-Channel Developer Guide

Version 64.0, Summer '25


Last updated: June 20, 2025

# CONTENTS

# CHAPTER 1 Omni-Channel Developer Guide

In this chapter ...



Customize your Omni-Channel records and console integration with Omni-Channel API objects and console methods.

**Important:** Standard Omni-Channel reaches its End of Life (EOL) with the Summer '26 release. For continued support, access to the latest features, and an enhanced experience, update to Enhanced Omni-Channel.

Omni-Channel routes work items to queues, agents, skills, and even Einstein Bots (on supported channels) based on defined routing logic. Depending on your business needs, you can use different ways to route work, including Omni-Channel flows.

Use Omni-Channel to manage the priority of work items, which makes it a cinch to route important work items to agents quickly. Manage your agents' capacity to take on work items so that they're given only the number of assignments that they can handle. You can also define which agents can work on different types of assignments. For example, you can create one group of agents to respond to leads and sales inquiries, and another group that helps customers with support questions.

Routing logic is applied when work is assigned to an owner. If field values on the work item are changed after the item is routed, the routing logic isn't reapplied.

# Omni-Channel Objects

Use an API to create, retrieve, update or delete records, such as accounts, leads, and custom objects. The Salesforce data model includes several objects that let you control and customize your Omni-Channel records, including Omni-Channel users, routing configurations, and statuses.

For more information on Salesforce APIs, see Which API Do I Use? in *Salesforce Help*.

- AgentWork
- AgentWorkSkill
- OmniSupervisorConfig
- OmniSupervisorConfigAction
- OmniSupervisorConfigGroup
- OmniSupervisorConfigProfile
- OmniSupervisorConfigQueue
- OmniSupervisorConfigSkill
- OmniSupervisorConfigTab
- OmniSupervisorConfigUser
- PendingServiceRouting
- PresenceConfigDeclineReason
- PresenceDeclineReason
- PresenceUserConfig
- PresenceUserConfigUser
- QueueRoutingConfig
- QueueSobject
- ServiceChannel
- ServiceChannelFieldPriority
- ServiceChannelStatus
- ServicePresenceStatus
- ServiceResource
- SkillRequirement
- UserServicePresence

# Omni-Channel Metadata API Types

The Metadata API lets you access Omni-Channel feature settings and metadata information.

The following types are available with the Metadata API.

- Flow
- OmniChannelSettings
- OmniSupervisorConfig
- PresenceDeclineReason
- PresenceUserConfig

- Queue
- QueueRoutingConfig
- ServiceChannel
- ServicePresenceStatus
- WorkSkillRouting
- WorkSkillRoutingAttribute

If you need more information on the Salesforce Metadata API, see the Metadata API Developer Guide.

# Omni-Channel Components for the Salesforce Console

Omni-Channel lets your call center route any type of incoming work item to the most qualified, available agents.



## Omni-Channel Components for the Lightning Console JavaScript API

Omni-Channel lets your call center route any type of incoming work item to the most qualified, available agents. The Lightning Console JavaScript API for Lightning Experience includes several methods and events that let you control how Omni-Channel works within the Lightning Service Console for your organization.

If you need more information on the Lightning Console JavaScript API, see Lightning Console JavaScript API.



### Methods for Omni-Channel in Lightning Experience

Use these Lightning Console JavaScript API methods for Omni-Channel. Omni-Channel lets your call center route any type of incoming work item to the most qualified, available agents.

- acceptAgentWork for Lightning Experience
- closeAgentWork for Lightning Experience
- declineAgentWork for Lightning Experience
- getAgentWorkload for Lightning Experience

- getAgentWorks for Lightning Experience
- getServicePresenceStatusChannels for Lightning Experience
- getServicePresenceStatusId for Lightning Experience
- login for Lightning Experience
- logout for Lightning Experience
- lowerAgentWorkFlag for Lightning Experience
- raiseAgentWorkFlag for Lightning Experience
- setServicePresenceStatus for Lightning Experience

## Events for Omni-Channel in Lightning Experience

Use these Lightning Console JavaScript API events for Omni-Channel. JavaScript can be executed when certain types of events occur in a console, such as when a user closes a tab. There are a few events that are specific to Omni-Channel. These events apply to Lightning Experience only.

- lightning:omniChannelConnectionError
- lightning:omniChannelLoginSuccess
- lightning:omniChannelStatusChanged
- lightning:omniChannelLogout
- lightning:omniChannelWorkAssigned
- lightning:omniChannelWorkAccepted
- lightning:omniChannelWorkDeclined
- lightning:omniChannelWorkClosed
- lightning:omniChannelWorkFlagUpdated
- lightning:omniChannelWorkloadChanged

# Omni-Channel Components for the Salesforce Console Integration Toolkit

The Salesforce Console Integration Toolkit includes several objects that let you control how Omni-Channel works within the Salesforce console for your organization. If your org is using Salesforce Classic, use Salesforce Console Integration Toolkit methods.

If you need more information on the Salesforce Console Integration Toolkit, see Salesforce Console Integration Toolkit for Salesforce Classic.

Methods for Omni-Channel for Salesforce Console Integration Toolkit

Use these Salesforce Console Integration Toolkit API methods for Omni-Channel in Salesforce Classic.

Methods for Console Events

JavaScript can be executed when certain types of events occur in a console, such as when a user closes a tab. In addition to the standard methods for console events, there are a few events that are specific to Omni-Channel. These events apply to Salesforce Classic only.

## Methods for Omni-Channel for Salesforce Console Integration Toolkit

Use these Salesforce Console Integration Toolkit API methods for Omni-Channel in Salesforce Classic.

- acceptAgentWork

- closeAgentWork
- declineAgentWork
- getAgentWorks
- getAgentWorkload
- getServicePresenceStatusChannels
- getServicePresenceStatusId
- login
- logout
- setServicePresenceStatus
- Methods for Omni-Channel Console Events

## Methods for Console Events

JavaScript can be executed when certain types of events occur in a console, such as when a user closes a tab. In addition to the standard methods for console events, there are a few events that are specific to Omni-Channel. These events apply to Salesforce Classic only.

### Standard Console Events

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

## Omni-Channel Console Events

| Event | Description | Payload |
|---|---|---|
| `sforce.console.ConsoleEvent.PRESENCE.LOGIN_SUCCESS` | Fired when an Omni-Channel user logs into Omni-Channel successfully.<br>Available in API version 32.0 or later. | • `statusId`—the ID of the agent's current presence status. |
| `sforce.console.ConsoleEvent.PRESENCE.STATUS_CHANGED` | Fired when a user changes his or her presence status.<br>Available in API version 32.0 or later. | • `statusId`—the ID of the agent's current presence status.<br>• `channels`—channelJSON string of channel objects.<br>• `statusName`—the name of the agent's current presence status.<br>• `statusApiName`—the API name of the agent's current presence status. |
| `sforce.console.ConsoleEvent.PRESENCE.LOGOUT` | Fired when a user logs out of Salesforce.<br>Available in API version 32.0 or later. | None |
| `sforce.console.ConsoleEvent.PRESENCE.WORK_ASSIGNED` | Fired when a user is assigned a new work item.<br>Available in API version 32.0 or later. | • `workItemId`—the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent.<br>• `workId`—the ID of a work assignment that's routed to an agent. |
| `sforce.console.ConsoleEvent.PRESENCE.WORK_ACCEPTED` | Fired when a user accepts a work assignment, or when a work assignment is automatically accepted.<br>Available in API version 32.0 or later. | • `workItemId`—the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent.<br>• `workId`—the ID of a work assignment that's routed to an agent. |
| `sforce.console.ConsoleEvent.PRESENCE.WORK_DECLINED` | Fired when a user declines a work assignment.<br>Available in API version 32.0 or later. | • `workItemId`—the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent.<br>• `workId`—the ID of a work assignment that's routed to an agent. |

| Event | Description | Payload |
|---|---|---|
| `sforce.console.ConsoleEvent.PRESENCE.WORK_CLOSED` | Fired when the status of an AgentWork object is changed to Closed.<br>Available in API version 32.0 or later. | • `workItemId`—the ID of the object that's routed through Omni-Channel. This object becomes a work assignment with a `workId` when it's assigned to an agent.<br>• `workId`—the ID of a work assignment that's routed to an agent. |
| `sforce.console.ConsoleEvent.PRESENCE.WORKLOAD_CHANGED` | Fired when an agent's workload changes. This includes receiving new work items, declining work items, and closing items in the console. It's also fired when there's a change to an agent's capacity or Presence Configuration or when the agent goes offline in the Omni-Channel widget. | • `ConfiguredCapacity`—the configured capacity for the agent.<br>• `PreviousWorkload`—the agent's workload before the change.<br>• `NewWorkload`—the agent's new workload after the change. |

### `channel`

The `channel` object contains the following properties:

| Name | Type | Description |
|---|---|---|
| `channelId` | String | Retrieves the ID of a service channel that's associated with a presence status. |
| `developerName` | String | Retrieves the developer name of the the service channel that's associated with the `channelId`. |

## Methods for Console Events

| Method | Description |
|---|---|
| addEventListener() | Adds a listener for a custom event type or a standard event type when the event is fired. This method adds a listener for custom event types in API version 25.0 or later; it adds a listener for standard event types in API version 30.0 or later. |
| fireEvent() | Fires a custom event. This method is only available in API version 25.0 or later. |
| removeEventListener() | Removes a listener for a custom event type or a standard event type. This method removes a listener for custom event types in API version 25.0 or later; it removes a listener for standard event types in API version 30.0 or later. |

# External Routing for Omni-Channel

Multiple routing options, one console. Integrate third-party routing with Omni-Channel to give your support team more routing options for their work.

Before setting up external routing, make sure that you have a working implementation of Omni-Channel. You'll use version 41.0 or later of Salesforce standard APIs and streaming APIs to connect Salesforce with an external routing implementation in your routing configuration. Then you can create queues that use either Omni-Channel routing or your external routing implementation.

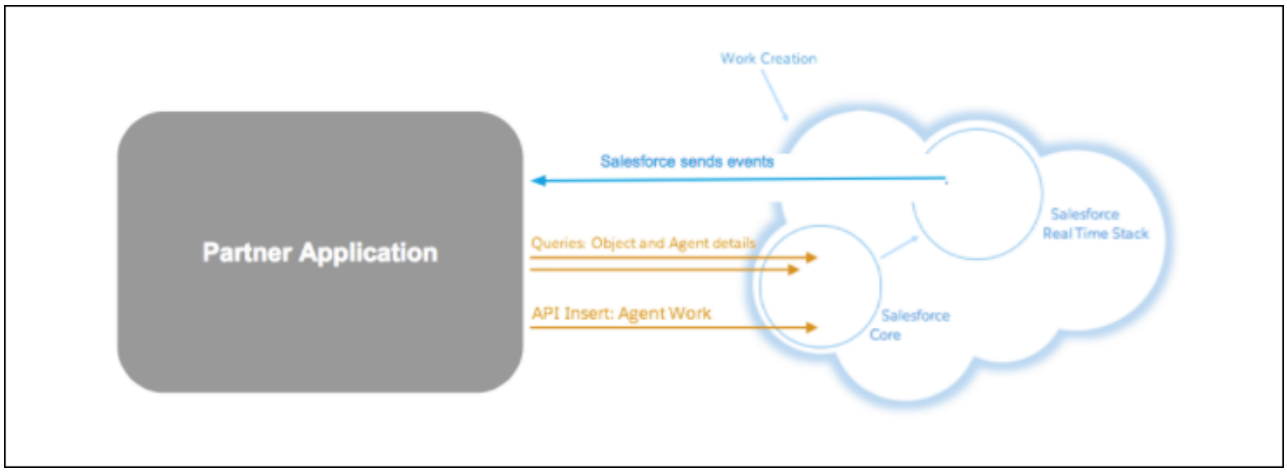
Are you ready to set up and use external routing? Let's get started.



## External Routing Technical Architecture and Process

See an overview of how external routing works to connect Salesforce with your external routing implementation.

The following image provides a visual description of how information is shared between Salesforce and your partner application.




Information is shared using Salesforce APIs and the APIs for your partner application using the following process.

1. Salesforce sends events using Salesforce Streaming APIs for the `PendingServiceRouting` object.
2. Partner application creates a session to authenticate.
3. Partner application queries Salesforce for work details, agent availability, and more.
4. Partner application determines the routing decision.
5. AgentWork is created and pushed to the specified agent in Salesforce.

Salesforce API Resources for External Routing
Use the following resources as you integrate your partner application with Omni-Channel.

## Salesforce API Resources for External Routing

Use the following resources as you integrate your partner application with Omni-Channel.

- AgentWork
- PendingServiceRouting
- UserServicePresence
- Streaming API Developer Guide
- Streaming API Message Durability
- Streaming API Limits

# Integrate External Routing for Omni-Channel

Use the following steps to integrate your external routing implementation with Omni-Channel.

1. Create a Routing Configuration and Queue for External Routing.
2. Create a PushTopic for `PendingServiceRouting`.
3. Listen to the Push Topic.
4. Create `AgentWork`.

## Step 1: Create a Routing Configuration and Queue for External Routing

External routing requires a separate routing configuration and queue in Omni-Channel. These separate objects define routing behavior and assign work to agents.

1. In Setup, enter *`Routing Configurations`* in the Quick Find box, then select **Routing Configurations**.
2. Create a routing configuration and select **External Routing** for the routing model.
3. Enter *`Queues`* in the Quick Find box, then select **Queues**.
4. Create a queue and connect it to the routing configuration you created.

SEE ALSO:

Create Routing Configurations for Your Queues

Create a Queue

## Step 2: Create a PushTopic for `PendingServiceRouting`

A PushTopic is a query that is the basis for notifying listeners of changes to records in an organization. Create a PushTopic for `PendingServiceRouting` so partners can receive event notifications.

Reference the following code sample to create a PushTopic using Apex code. For more information about PushTopic events, see Streaming API Developer Guide.

PushTopic events use the following database values to indicate work status:

- ASSIGNED (0, “Assigned”)

- UNAVAILABLE (1, "Unavailable")
- DECLINED (2, "Declined")
- OPENED (3, "Opened")
- CLOSED (4, "Closed")
- DECLINED_ON_PUSH_TIMEOUT (5, "DeclinedOnPushTimeout")
- CANCELLED (6, "Canceled")
- TRANSFERRED (7, "Transferred")

```
PushTopic pushTopic = new PushTopic();
pushTopic.Name = PSRPushTopic;
pushTopic.Query = 'Select Id, Serial, QueueId, WorkItemId, IsPushed, ServiceChannelId,
LastDeclinedAgentSession, CreatedDate from PendingServiceRouting where RoutingModel =
\'ExternalRouting\'';
pushTopic.ApiVersion = 41.0;
pushTopic.NotifyForOperationCreate = true;
pushTopic.NotifyForOperationUpdate = true;
pushTopic.NotifyForOperationDelete = true;
pushTopic.NotifyForFields = 'Referenced';
insert pushTopic;
```

### Step 3: Listen to the PushTopic

Give your event notifications somewhere to go by setting up a listener.

Salesforce's Streaming API uses the HTTP/1.1 request-response model and the Bayeux protocol (CometD implementation). The easiest way to connect to the Streaming API is to use java sdk and Oauth flow to connect to the PushTopic you created.

For reference and a code sample, see Use the Connector with OAuth Bearer Token Login from the *Streaming API Developer Guide*.

### Step 4: Create `AgentWork`

Your partner application must create an `AgentWork` record to route the work to an agent in Omni-Channel.

When the external routing implementation receives new `PendingServiceRouting` creation events (where the routing type equals *External*), it uses the standard Salesforce SOAP API to fetch further information based on the `PendingServiceRouting`. It makes a routing decision by creating `AgentWork` records. This API is existing functionality that partners use to query Salesforce.

Reference the following code sample to create `AgentWork` using Apex.

```
AgentWork work = new AgentWork();
work.ServiceChannelId = '<ServiceChannelId>';
work.WorkItemId = '<WorkItemId>';
work.UserId = '<UserId>';
work.PendingServiceRoutingId = '<PendingServiceRoutingId>';
insert work;
```

## Expected Behavior for External Routing for Omni-Channel

Verify that the behavior you observe while testing and using your implementation of external routing matches the following expected behavior scenarios.

**Agent accepts the work:**

1. Chat visitor initiates a chat request from the external routing button.
2. `PendingServiceRouting` is created.
3. Partner is notified by a pushTopic event (EventType=Create, isPushed=false).
4. Partner creates AgentWork using the PSR.
5. Agent is routed the chat request (AgentWork Status = Assigned).
6. Agent accepts the chat request (AgentWork Status = Accept).
7. Omni-Channel deletes the `PendingServiceRouting` after Agent accepts the work.
8. Partner is notified by a pushTopic event (EventType=Delete).

**Agent declines the work through Omni-Channel:**

1. Agent declines the assigned `AgentWork`.
2. Salesforce updates the `PendingServiceRouting`.
3. Partner is notified by a pushTopic event (EventType=Update, LastDeclinedAgentSession=agent's session id in Chat (not the Salesforce session), isPushed=false).
4. Partner creates a new `AgentWork` using the updated `PendingServiceRouting` for rerouting.

**Agent doesn't accept the work due to push time-out:**

1. Existing `PendingServiceRouting` is updated.
2. Partner is notified by a pushTopic event (EventType=Update, PSR Fields updated: isPushed=false, LastDeclinedAgentSession=agent's liveagent session id).
3. Partner creates a new `AgentWork` for rerouting.

**Agent transfers the work to an external routing queue:**

1. New `PendingServiceRouting` for the transfer is created.
2. Partner is notified by a pushTopic event (EventType=Create, isTransfer=true, isPushed=false).
3. The routing process is repeated.

**Agent transfers the work to another agent :**

1. The `PendingServiceRouting` from the original chat request is deleted.
2. A new `PendingServiceRouting` isn't created when the work is transferred. Subscribe to AgentWork and LiveChatTranscript to determine whether the work was transferred to an agent.
3. Two AgentWorks are created for the LiveChatTranscript:
   a. First AgentWork with the Status = *Opened*
   b. Second AgentWork with the Status = *Assigned*
4. The LiveChatTranscript is updated with the Status = *In Progress* and the Owner = *second Agent*.
5. To determine if the Transfer to Agent has occurred, check that the second AgentWork isn't inserted into the same LiveChatTranscript as the first AgentWork.

Important: We don't recommend using both external routing and Omni-Channel queue-based routing in the same implementation. If the same agent is in both queues, the agent's capacity could be exceeded. We don't have control over an agent's capacity in external routing. If you attempt this combination, there can be unknown issues.

# Troubleshooting External Routing for Omni-Channel

If you encounter issues with your implementation of External Routing for Omni-Channel, try the following troubleshooting steps.

## Recover from an External Routing Adaptor Restart

When the third-party adaptor recovers from restarting, it should leverage the durability feature of the Streaming API (since version 37.0) and replay from the last successful position of the PSR topic.

Reference the following code sample in Java.

```
// Register streaming extension
var replayExtension = new cometdReplayExtension();
replayExtension.setChannel(***<Streaming Channel to Subscribe to>***);
replayExtension.setReplay(<Event Replay Option>);
cometd.registerExtension('myReplayExtensionName', replayExtension);
```

For more information, see Message Durability in the *Streaming API Developer Guide*.

## Recover from a Salesforce Data Recovery Instance

An org instance can be recovered from a Salesforce data center switch. The recovery process involves downtime, so all online agents must be logged out. All states maintained by the third-party adaptor, such as agent presence, aren't applicable and must be reset. The third-party adaptor should reinitialize as when it first subscribed to the topic.

## Test the Client Solution

You can use the Streaming API to listen to CRUD events for `UserServicePresence` and `PendingServiceRouting`. For examples, see Code Examples in the *Streaming API Developer Guide*.

# INDEX

## O