Platform Events Developer Guide

Version 61.0, Summer '24

Last updated: August 16, 2024

CONTENTS

PLATFORM EVENTS DEVELOPER GUIDE

Use platform events to connect business processes in Salesforce and external apps through the exchange of real-time event data. Platform events are secure and scalable messages that contain data. Publishers publish event messages that subscribers receive in real time. To customize the data published, define platform event fields.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Performance**, **Unlimited**, **Enterprise**, and **Developer** Editions

IN THIS SECTION:

Delivering Custom Notifications with Platform Events

Platform events are part of Salesforce's enterprise messaging platform. The platform provides an event-driven messaging architecture to enable apps to communicate inside and outside of Salesforce. Before diving into platform events, take a look at what an event-based software system is.

IN THIS SECTION:

Event-Driven Software Architecture

An event-driven (or message-driven) software architecture consists of event producers, event consumers, and channels. The architecture is suitable for large distributed systems because it decouples event producers from event consumers, thereby simplifying the communication model in connected systems.

Event

A change in state that is meaningful in a business process. For example, placement of a purchase order is a meaningful event because the order fulfillment center expects to receive a notification before processing an order.

Event message

A message that contains data about the event. Also known as an event notification. For example, an event message can be a notification about an order placement containing information about the order.

Event producer

The publisher of an event message.

Event channel

A stream of events on which an event producer sends event messages and event consumers read those messages. For platform events, the channel is for a single platform event or a custom channel that groups event messages for multiple platform events.

Event consumer

A subscriber to a channel that receives messages from the channel. For example, an order fulfillment app that is notified of new orders.

Event bus

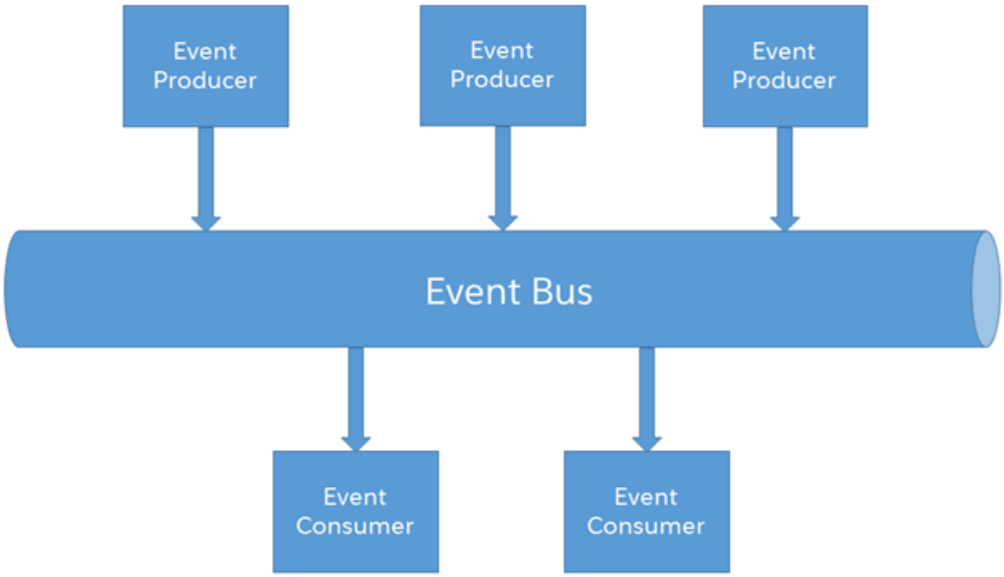
A multitenant, multicloud event storage and delivery service based on a publish-subscribe model. The event bus enables the retrieval of stored event messages at any time during the retention window. The event bus is based on a time-ordered event log, which ensures that event messages are stored and delivered in the order that they're received by Salesforce.

Systems in request-response communication models make a request to a web service or database to obtain information about a certain state. The sender of the request establishes a connection to the service and depends on the availability of the service.

In comparison, systems in an event-based model obtain information and can react to it in near real time when the event occurs. Event producers don't know the consumers that receive the events. Any number of consumers can receive and react to the same events. The only dependency between producers and consumers is the semantic of the message content.

The Event Bus

Platform event messages are published to the event bus, where they're stored temporarily. You can retrieve stored event messages from the event bus using Pub/Sub API . Each event message contains the Replay ID field, which identifies the event in the stream and enables replaying the stream after a specific event. For more information, see Event Message Durability in the *Pub/Sub API Developer Guide*.



Enterprise Messaging Platform Events

The Salesforce enterprise messaging platform offers the benefits of event-driven software architectures. Platform events are the event messages (or notifications) that your apps send and receive to take further action. Platform events simplify the process of communicating changes and responding to them without writing complex logic. Publishers and subscribers communicate with each other through events. One or more subscribers can listen to the same event and carry out actions.

For example, a software system can send events containing information about printer ink cartridges. Subscribers can subscribe to the events to monitor printer ink levels and place orders to replace cartridges with low ink levels.

Custom Platform Events

Use custom platform events to publish and process custom notifications. For example, publish custom platform events to send order information to an order fulfillment service. Or publish custom platform events to send printer ink information that is processed by a service app.

You define a custom platform event in Salesforce in the same way that you define a custom object. Create a platform event definition by giving it a name and adding custom fields. Platform events support a subset of field types in Salesforce. See Platform Event Fields. This table lists a sample definition of custom fields for a printer ink event.

Field Name	Field API Name	Field Type
Printer Model	Printer_Model__c	Text
Serial Number	Serial_Number__c	Text
Ink Percentage	Ink_Percentage__c	Number

You can publish custom platform events on the Lightning Platform by using Apex or point-and-click tools, such as Process Builder and Flow Builder, or an API in external apps. Similarly, you can subscribe to an event either on the platform through an Apex trigger or point-and-click tools or in external apps, such as Pub/Sub API. When an app publishes an event message, event subscribers receive the event message and execute business logic. Using the printer ink example, a software system monitoring a printer makes an API call to publish an event when the ink is low. The printer event message contains the printer model, serial number, and ink level. After the printer sends the event message, an Apex trigger is fired in Salesforce. The trigger creates a case record to place an order for a new cartridge.

Standard Platform Events

Salesforce provides events with predefined fields, called standard platform events. An example of a standard platform event is AssetTokenEvent, which monitors OAuth 2.0 authentication activity. Another example is BatchApexErrorEvent, which reports errors encountered in batch Apex jobs.

Salesforce publishes standard platform events in response to an action that occurred in the app or errors in batch Apex jobs. You can subscribe to a standard platform event stream using the subscription mechanism that the event supports.

High-Volume Platform Events

Use high-volume platform events to publish and process millions of events efficiently and to scale your event-based apps.

Note the following characteristics of high-volume platform events.

Asynchronous Publishing
For efficient processing of high loads of incoming event messages, high-volume platform events are published asynchronously. After the publishing call returns with a successful result, the publish request is queued in Salesforce. When system resources become available, the system carries out the publish request and saves the event message in the event bus. If the publishing of the queued event fails, the system retries the publishing internally using the at-least-once model. See Considerations for Publishing Platform Events.

Separate Event Allocations
Each Salesforce edition provides default allocations and usage-based entitlements for the number of high-volume events delivered to clients. See Platform Event Allocations.

Starting in Spring '21, standard-volume platform events are also published asynchronously.

Note: Previously, standard-volume events were available. In API version 45.0 and later, your new custom event definitions are high volume by default. High-volume platform events offer better scalability than standard-volume platform events. Standard-volume custom platform events will be retired in Winter '25. To migrate existing standard-volume events, see Migrate Standard-Volume Platform Events to High-Volume Platform Events Before Retirement.

Platform Events and sObjects

A platform event is a special kind of Salesforce entity, similar in many ways to an sObject. An event message is an instance of a platform event, similar to how a record is an instance of a custom or standard object. Unlike custom or standard objects, you can't update or delete event records. You also can't view event records in the Salesforce user interface, and platform events don't have page layouts. When you delete a platform event definition, it's permanently deleted.

Platform Event Permissions

Grant user permissions for publishing and subscribing to platform events.

	User Permissions Needed
To publish a platform event:	Create for the platform event object
To subscribe to a platform event:	Read for the platform event object

For more information about granting user permissions, see Manage Data Access in *Salesforce Help*.

Platform Events and Transactions

Platform event messages are published either immediately or after a transaction is committed, depending on the publish behavior that you set in the platform event definition. Platform events defined to be published immediately don't respect transaction boundaries, but those defined to be published after a transaction is committed do. The publish behavior doesn't apply to Pub/Sub API. See Platform Event Fields.

- If the platform event publish behavior is set to **Publish Immediately**:
  - The `allOrNone` header is ignored when publishing through the APIs. Some events can be published even when others fail in the same call.
  - You can't roll back published event messages, and the Apex `setSavepoint()` and `rollback()` `Database` class methods aren't supported.
- If the publish behavior is set to **Publish After Commit**:
  - The `allOrNone` header value takes effect for the initial enqueuing of the events when publishing through the APIs. If `allOrNone` is set to `true`, Salesforce doesn't enqueue any events for publishing if even one event fails to be enqueued in the same call. As a result, no events are published. The failures are synchronous errors, such as event-validation or limit errors. After all events in the same call are successfully enqueued for publishing, the `allOrNone` header isn't enforced for the eventual event publishing. If asynchronous system errors occur while the enqueued events are published, some of the enqueued events can be published when others fail.
  - You can roll back published event messages with the Apex `setSavepoint()` and `rollback()` `Database` class methods.
- The publishing of high-volume platform events is asynchronous. For more information, see Asynchronous Publishing.

When publishing platform events, DML limits and other Apex governor limits apply.

Event Retention in the Event Bus

High-volume platform event messages are stored for 72 hours (3 days). Legacy standard-volume platform event messages are stored for 24 hours (1 day). You can retrieve past event messages when using Pub/Sub API to subscribe to a channel.

For more information, see Event Message Durability in the *Pub/Sub API Developer Guide*.

Order of Events

If you publish multiple events in one publish call, the order of events in a batch is guaranteed for that publish request. So the order of event messages that are stored in the event bus and delivered to subscribers matches the order of events that are passed in the call. You can publish multiple events in several ways, including the Apex `EventBus.publish` method or the REST API composite resource. For events published across different requests, the order of events isn't guaranteed because publish requests can be processed by different Salesforce application servers. As a result, a later request could be processed faster than an earlier request.

Salesforce assigns a replay ID value to a received platform event message and persists it in the event bus. Subscribers receive platform event messages from the event bus in the order of the replay ID.

SEE ALSO:

- Publishing Platform Events
- Subscribing to Platform Events
- Standard Platform Event Objects

Defining Your Custom Platform Event

Define a platform event in Setup and add custom fields.

IN THIS SECTION:

Platform Event Fields
Platform events contain standard fields. Add custom fields for your custom data.

Migrate Platform Event Definitions with Metadata API
Deploy and retrieve platform event definitions from your sandbox and production org as part of your app's development life cycle.

Get the Event Schema
To discover the event fields of your platform event, get the event schema. You can get the event schema through REST API or Pub/Sub API.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Performance**, **Unlimited**, **Enterprise**, and **Developer** Editions

USER PERMISSIONS

To create and edit platform event definitions:

- Customize Application

Platform Event Fields

Platform events contain standard fields. Add custom fields for your custom data.

To define a platform event in Salesforce Classic or Lightning Experience:

1. From Setup, enter *Platform Events* in the Quick Find box, then select **Platform Events**.
2. On the Platform Events page, click **New Platform Event**.
3. Complete the standard fields, and optionally add a description.
4. For Publish Behavior, choose when the event message is published in a transaction.
   - **Publish After Commit** to have the event message published only after a transaction commits successfully. Select this option if subscribers rely on data that the publishing transaction commits. For example, a process publishes an event message and creates a task record. A second process that is subscribed to the event is fired and expects to find the task record. Another reason for choosing this behavior is when you don't want the event message to be published if the transaction fails.

- **Publish Immediately** to have the event message published when the publish call executes. Select this option if you want the event message to be published regardless of whether the transaction succeeds. Also choose this option if the publisher and subscribers are independent, and subscribers don't rely on data committed by the publisher. For example, the immediate publishing behavior is suitable for an event used for logging purposes. With this option, a subscriber can receive the event message before data is committed by the publisher transaction.

Note: The publish behavior doesn't apply to event messages published with Pub/Sub API.

5. Click **Save**.
6. To add a field, in the Custom Fields & Relationships related list, click **New**.
7. To set up the field properties, follow the custom field wizard.

Note:

- If you change the publish behavior, expect up to a 5-minute delay for the change to take effect.
- In Lightning Experience, platform events aren't shown in the Object Manager's list of standard and custom objects and aren't available in Schema Builder.

Standard Fields

Platform events include standard fields. These fields appear on the New Platform Event page.

Field	Description
`Label`	Name used to refer to your platform event in a user interface page.
`Plural Label`	Plural name of the platform event.
`Starts with a vowel sound`	If it's appropriate for your org's default language, indicate whether the label is preceded by "an" instead of "a."
`Object Name`	Unique name used to refer to the platform event when using the API. In managed packages, this name prevents naming conflicts with package installations. Use only alphanumeric characters and underscores. The name must begin with a letter and have no spaces. It can't end with an underscore or have two consecutive underscores.
`Description`	Optional description of the object. A meaningful description helps you remember the differences between your events when you view them in a list.
`Deployment Status`	Indicates whether the platform event is visible to other users.

Custom Fields

In addition to the standard fields, you can add custom fields to your custom event. Platform event custom fields support only these field types.

- Checkbox
- Date

- Date/Time
- Number
- Text
- Text Area (Long)

The maximum number of fields that you can add to a platform event is the same as for a custom object. See Salesforce Features and Edition Allocations.

`ReplayId` System Field

Each event message is assigned an opaque ID contained in the `ReplayId` field. The `ReplayId` field value, which is populated by the system when the event is delivered to subscribers, refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve events that are within the retention window. For example, a subscriber can retrieve missed events after a connection failure. Subscribers must not compute new replay IDs based on a stored replay ID to refer to other events in the stream.

To learn more about how to use the `ReplayId` field when resubscribing to the stream, see Replaying an Event Stream in the *Pub/Sub API Reference*.

To uniquely identify a platform event message, use the `EventUuid` system field and not the `ReplayId` field. The `ReplayId` field isn't guaranteed to be unique when Salesforce maintenance activities occur, such as an org migration. The `EventUuid` field is always unique.

`EventUuid` System Field

A universally unique identifier (UUID) that identifies a platform event message. The system populates the `EventUuid` field, and you can't overwrite its value. This field is available in API version 52.0 and later. The API version corresponds to the version that an Apex trigger is saved with or the version specified in a CometD subscriber endpoint.

API Name Suffix for Custom Platform Events

When you create a platform event, the system appends the `__e` suffix to create the API name of the event. For example, if you create an event with the object name `Low Ink`, the API name is `Low_Ink__e`. The API name is used whenever you refer to the event programmatically, for example, in Apex. API names of standard platform events, such as AssetTokenEvent, don't include a suffix.

SEE ALSO:

Considerations for Defining and Publishing Platform Events

Considerations for Publishing and Subscribing to Platform Events with Apex and APIs

Migrate Platform Event Definitions with Metadata API

Deploy and retrieve platform event definitions from your sandbox and production org as part of your app's development life cycle.

The CustomObject metadata type represents a platform event.

Platform event names are appended with `__e`. The file that contains the platform event definition has the suffix `.object`. Platform events are stored in the `objects` folder.

Example: Here is a definition of a platform event with a number field and two text fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>
    <eventType>HighVolume</eventType>
    <publishBehavior>PublishAfterCommit</publishBehavior>
    <fields>
        <fullName>Ink_Percentage__c</fullName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Ink Percentage</label>
        <precision>18</precision>
        <required>false</required>
        <scale>2</scale>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Printer_Model__c</fullName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Printer Model</label>
        <length>20</length>
        <required>false</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Serial_Number__c</fullName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Serial Number</label>
        <length>20</length>
        <required>false</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <label>Low Ink</label>
    <pluralLabel>Low Ink</pluralLabel>
</CustomObject>
```

The `eventType` field specifies the platform event volume. Only the `HighVolume` value is supported. The `StandardVolume` value is deprecated. If you create a platform event with the `StandardVolume` event type, you get an error.

This package.xml manifest file references the previous event definition. The name of the referenced event is `Low_Ink__e`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <types>
        <members>Low_Ink__e</members>
        <name>CustomObject</name>
    </types>
    <version>61.0</version>
</Package>
```

Retrieve Platform Events

To retrieve all platform events, in addition to custom objects defined in your org, use the wildcard character (`*`) for the `<members>` element, as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CustomObject</name>
    </types>
    <version>61.0</version>
</Package>
```

To retrieve or deploy triggers associated to a platform event, use the ApexTrigger metadata type. For more information about how to use Metadata API and its types, see the *Metadata API Developer Guide*.

Get the Event Schema

To discover the event fields of your platform event, get the event schema. You can get the event schema through REST API or Pub/Sub API.

IN THIS SECTION:

Get the Event Schema with REST API

Use REST API `eventschema` resource to retrieve the event schema by using the event name or the schema ID.

To retrieve the event schema by schema ID, perform a GET request to this REST API resource and supply the schema ID: `/services/data/v`***`XX.X`***`/event/eventSchema/`***`schemaId`***. For more information, see Platform Event Schema by Schema ID in the *REST API Developer Guide*.

To retrieve the event schema by event name, perform a GET request to this REST API resource and supply the event name: `/services/data/v`***`XX.X`***`/sobjects/`***`eventName`***`/eventSchema`. For more information, see Platform Event Schema by Event Name in the *REST API Developer Guide*.

When the schema changes, for example, after an administrator adds a field to the platform event definition, the schema ID changes. You can determine if the schema changed by comparing the schema ID with the previous schema ID value.

Get the Event Schema with Pub/Sub API

Use Pub/Sub API to retrieve the event schema with the `GetSchema` RPC method and pass in a schema ID.

Because the schema typically doesn't change often, we recommend that you call `GetSchema` once and use the returned schema for all operations. If the event schema changes, for example, when an administrator adds a field to the event definition, the schema ID changes. We recommend that you store the schema ID and compare it with the latest schema ID. If the schema ID changes, call `GetSchema` to retrieve the new schema.

```
rpc GetSchema (SchemaRequest) returns (SchemaInfo);
```

For more information, see GetSchema RPC Method in the *Pub/Sub API Developer Guide*.

Publishing Platform Events

After a platform event has been defined in your Salesforce org, publish event messages from a Salesforce app using processes, flows, or Apex or an external app using Salesforce APIs.

IN THIS SECTION:

SEE ALSO:

Decoupled Publishing and Subscription

Publish Event Messages with Flows

Use flows to publish event messages from a Salesforce app as part of some user interaction, an automated process, Apex, or workflow action.

To publish event messages, add a Create Records element to the appropriate flow. Where you'd usually pick an object to create, select the custom platform event.

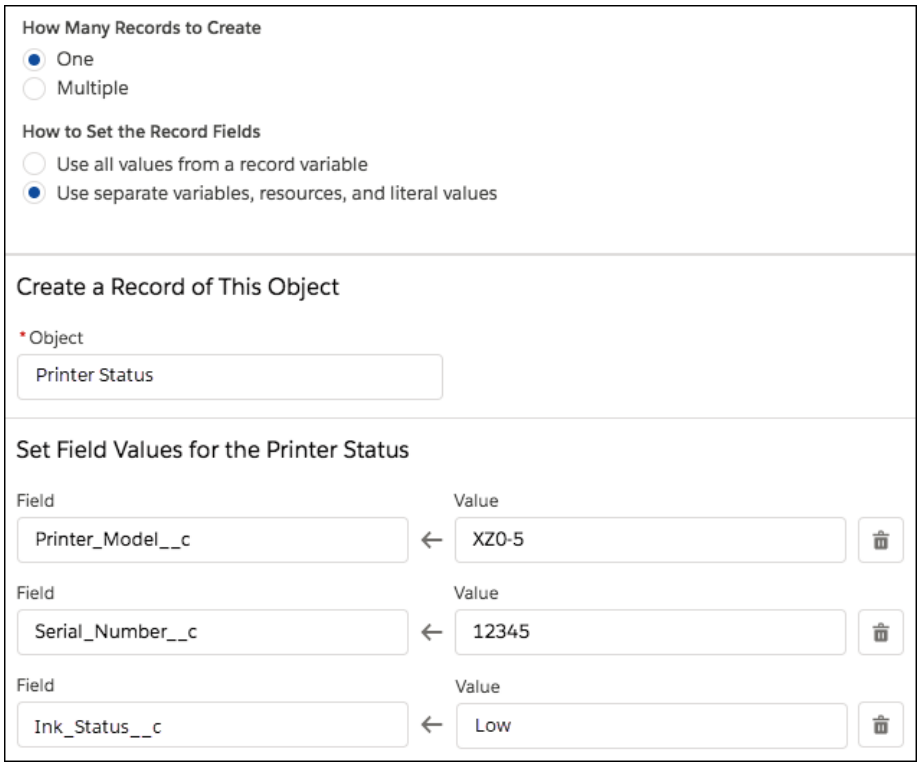
For example, here's how to configure a Create Records element that publishes a Printer Status platform event message. This example assumes that the Printer Status platform event is defined in your org and that the event has these custom fields.

- Printer Model (Text)

- Serial Number (Text)
- Ink Status (Text)

1. For How Many Records to Create, choose **One**.
2. For How to Set the Record Fields, choose **Use separate variables, resources, and literal values**.
3. For Object, enter `Printer` and select **Printer Status**.
4. Set these field values.

Field	Value
Printer Model	XZO-5
Serial Number	12345
Ink Status	Low



5. Save and activate the flow.

SEE ALSO:

Salesforce Help: Flows

Publish Event Messages with Processes

Use Process Builder to publish event messages from a Salesforce app as part of an automated process.

To publish event messages, add a Create a Record action to the appropriate process. Where you'd usually pick an object to create, select the custom platform event.

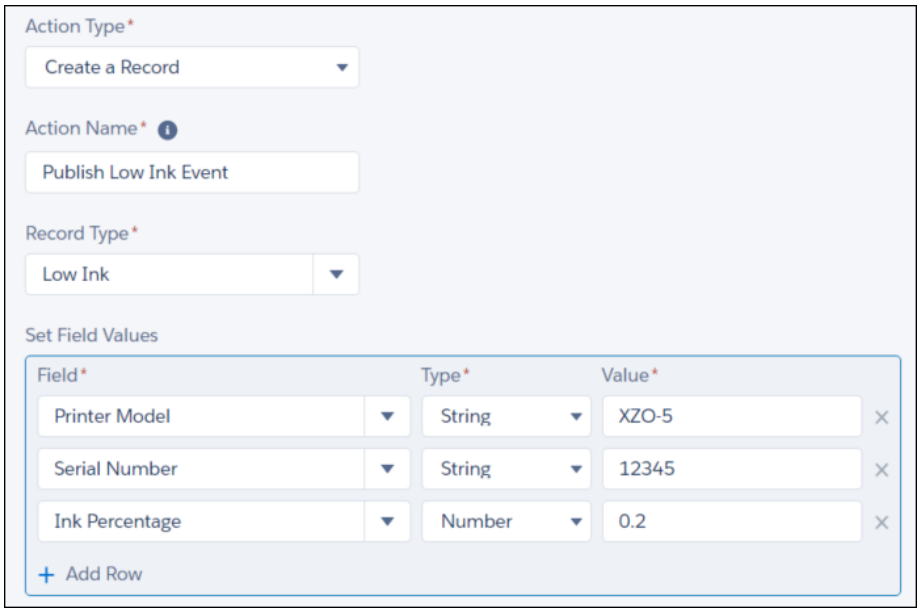
Tip: If a platform event is configured to publish immediately, the process publishes each event message outside of the database transaction. If the transaction fails and is rolled back, the event message is still published and can't be rolled back. So if you see an informational message under the selected platform event, consider whether you want the process to publish an event message only after the transaction commits successfully.

For example, here's how to configure a Create a Record action that publishes a Low Ink event message. This example assumes that the Low Ink platform event is defined in your org and that the event has these custom fields.

- Printer Model (Text)
- Serial Number (Text)
- Ink Percentage (Number)

1. For Record Type, enter `low` and select **Low Ink**.
2. Set the field values.

Field	Type	Value
Printer Model	String	XZO-5
Serial Number	String	12345
Ink Percentage	Number	0.2



3. Save the action and activate the process.

SEE ALSO:

Salesforce Help: Process Builder

Decoupled Publishing and Subscription

Platform Event Fields

Publish Event Messages with Apex

Use Apex to publish event messages from a Salesforce app.

Create Events in Apex

Before you can publish event messages, create platform event instances.

Create Events Using the Event API Name:

Create an event instance the same way that you create a Salesforce or custom object instance. Use the `new` operator with the event API name.

```
// Create event
Event_Name__e event = new Event_Name__e();
// Set field values
event.field1__c = 'value';
...

// Or create event with fields
Event_Name__e event = new Event_Name__e(field1__c='value', …);
```

Create Events With a Prepopulated `EventUuid` Field:

If you want to have the standard `EventUuid` field prepopulated in the event variable, use the Apex `sobjectType.newSobject` method to create an event.

The `EventUuid` field holds a universally unique identifier (UUID) that identifies an event message. You can use the `EventUuid` field to track the delivery of event messages. For more information, see Get the Result of Asynchronous Platform Event Publishing with Apex Publish Callbacks.

```
// Create event
Event_Name__e event = (Event_Name__e)Event_Name__e.sObjectType.newSObject(null, true);
// Set field values
event.field1__c = 'value';
...
// Display the prepopulated EventUuid
System.debug('EventUuid: ' + event.EventUuid);
```

To publish event messages, call the `EventBus.publish` method. For example, if you defined a custom platform event called `Low Ink`, reference this event type as `Low_Ink__e`. Next, create instances of this event, and then pass them to the Apex method.

Example: This example creates two events of type `Low_Ink__e`, publishes them, and then checks whether the publishing was successful or errors were encountered.

Before you can run this snippet, define a platform event with the name of `Low_Ink__e` and these fields: `Printer_Model__c` of type Text, `Serial_Number__c` of type Text (marked as required), `Ink_Percentage__c` of type Number(16, 2).

```
List<Low_Ink__e> inkEvents = new List<Low_Ink__e>();
inkEvents.add(new Low_Ink__e(Printer_Model__c='XZO-5', Serial_Number__c='12345',
              Ink_Percentage__c=0.2));
inkEvents.add(new Low_Ink__e(Printer_Model__c='MN-123', Serial_Number__c='10013',
              Ink_Percentage__c=0.15));


// Call method to publish events
List<Database.SaveResult> results = EventBus.publish(inkEvents);
```

```
// Inspect publishing result for each event
for (Database.SaveResult sr : results) {
    if (sr.isSuccess()) {
        System.debug('Successfully published event.');
    } else {
        for(Database.Error err : sr.getErrors()) {
            System.debug('Error returned: ' +
                        err.getStatusCode() +
                        ' - ' +
                        err.getMessage());
        }
    }
}
```

Immediate Publish Result in `Database.SaveResult`

For each event, `Database.SaveResult` contains information about whether the operation was successful and the errors encountered. If the `isSuccess()` method returns `true`, the publish request is queued in Salesforce and the event message is published asynchronously. For more information, see High-Volume Platform Event Persistence. If `isSuccess()` returns `false`, the event publish operation resulted in errors, which are returned in the `Database.Error` object. `EventBus.publish()` can publish some passed-in events, even when other events can't be published due to errors. The `EventBus.publish()` method doesn't throw exceptions caused by an unsuccessful publish operation. It's similar in behavior to the Apex `Database.insert` method when called with the partial success option.

`Database.SaveResult` also contains the `Id` system field. The `Id` field value isn't included in the event message delivered to subscribers. It isn't used to identify an event message, and isn't always unique.

Status Code Returned for Asynchronous Publishing

To indicate that the publish operation is asynchronous, the `OPERATION_ENQUEUED` status code is returned for a successful `EventBus.publish` call in `Database.SaveResult`, in addition to the event UUID. You can get the status code and event UUID after checking for a successful result. This example is a printout of the contents of `Database.SaveResult` after a successful publish call. The `getStatusCode` method of `Database.Error` returns the status code of `OPERATION_ENQUEUED`. The `getMessage` method returns the event UUID value for the published event message.

```
Database.SaveResult[getErrors=(
    Database.Error[getFields=();
    getMessage=d65ae914-2488-414a-85d4-4df93ea9a05c;
    getStatusCode=OPERATION_ENQUEUED;]);
getId=e02xx0000000001AAA;isSuccess=true;]
```

Publish Behavior

The platform event message is published either immediately or after a transaction is committed, depending on the publish behavior that you set in the platform event definition. For more information, see Platform Event Fields. Apex governor limits apply. For events configured with the **Publish After Commit** behavior, each method execution is counted as one DML statement against the Apex DML statement limit. You can check limit usage using the Apex `Limits.getDMLStatements()` method. For events configured with the **Publish Immediately** behavior, each method execution is counted against a separate event publishing limit of 150 `EventBus.publish()` calls. You can check limit usage using the Apex `Limits.getPublishImmediateDML()` method.

IN THIS SECTION:

SEE ALSO:

Get the Result of Asynchronous Platform Event Publishing with Apex Publish Callbacks

Get the final result of an `EventBus.publish` call through an Apex publish callback that you implement. Without the callback, you can get only the intermediate queueing result in `Database.SaveResult` of an `EventBus.publish` call, not the final result.

Event publishing is asynchronous, and the immediate result returned in `SaveResult` is the result of queuing the publish operation in Salesforce. Sometimes immediate errors are returned, for example, due to a missing required field in the event message. If no immediate errors are returned and when resources become available, the system carries out the queued publish call, and the final result is sent in an Apex publish callback.

Note: Apex publish callbacks are available for high-volume platform events. Legacy standard-volume events aren't supported.

IN THIS SECTION:

Example: Publish Events with a Callback Instance

To invoke the callback, perform an `EventBus.publish` call by passing it an instance of the `FailureCallback` class. You can publish one event or a batch of events with the callback.

Example: Publish Callback Class That Creates Follow-Up Tasks for Failed and Successful Publishes

This publish callback class is a modification of the previous example—it also implements the `EventBus.EventPublishSuccessCallback` interface and processes both success and failure cases. It creates a task when event publishing fails or succeeds. The inserted task includes the number of failed events and the event UUIDs.

Example: Publish Callback Class That Correlates Callback Results with Event Messages

This example callback class implementation shows how to retry publishing failed events. It's based on a trigger on the Order object.

Test Apex Publish Callbacks

To test your Apex publish callback class, add an Apex test class. You must provide Apex tests before you can package or deploy an Apex class to production and meet code coverage requirements.

Apex Publish Callback Limits

Keep in mind this limit for Apex publish callbacks.

Apex Publish Callback Class

An Apex publish callback contains the result of an asynchronous publish operation in Apex. After the publish operation completes and the final result is ready, the system returns a callback. You can implement one of these two interfaces: `EventBus.EventPublishFailureCallback` for failed publishes and `EventBus.EventPublishSuccessCallback` for successful publishes.

Track Event Publish Failures

To track the final failed result of asynchronous publish operations, implement the `EventBus.EventPublishFailureCallback` interface in an Apex class. In your implementation, you can decide what action to take for publish failures. For example, you can log the failures or you can attempt to republish the events.

```
public class FailureCallback implements EventBus.EventPublishFailureCallback {

   public void onFailure(EventBus.FailureResult result) {
       // Your implementation
       // Get event UUIDs from the result
       List<String> eventUuids = result.getEventUuids();
       // ...
   }
}
```

If the asynchronous publish operation fails, the `onFailure` method is invoked. In the implemented `onFailure` method, you can write logic to act in response to the final result of the publishing operation. The `onFailure` method takes a parameter that contains the result of the publish operation: `EventBus.FailureResult result`. The result contains the `EventUuid` field values for each failed event but doesn't contain the data for the event. Use the `getEventUuids` method to get the universally unique identifiers (UUIDs) of the events. Each event UUID is a UUID that identifies an event message.

Track Event Publish Successes

To track the final successful result of asynchronous publish operations, implement the `EventBus.EventPublishSuccessCallback` interface in an Apex class. Because most publish calls typically succeed,

processing successful event publishes likely isn't a concern. Also, a large volume of events can be published successfully, so be mindful about the performance and Apex limit impacts when processing the results.

```
public class SuccessCallback implements EventBus.EventPublishSuccessCallback {

    public void onSuccess(EventBus.SuccessResult result) {
        // Your implementation
        // Get event UUIDs from the result
        List<String> eventUuids = result.getEventUuids();
        // …
    }
}
```

If the asynchronous publish operation succeeds, the `onSuccess` method is invoked. In the implemented `onSuccess` method, you can write logic to act in response to the final result of the publishing operation. The `onSuccess` method takes a parameter that contains the result of the publish operation: `EventBus.SuccessResult result`. The result contains the `EventUuid` field values for each successfully published event but doesn't contain the data for the event. Use the `getEventUuids` method to get the UUIDs of the events. Each event UUID is a UUID that identifies an event message.

Track Event Publish Failures and Successes in One Callback

Alternatively, you can process failed and successful publish results in one Apex class. Implement the `EventBus.EventPublishFailureCallback` and `EventBus.EventPublishSuccessCallback` interfaces in the same Apex class. The interface includes the `onFailure` and `onSuccess` methods.

Note: Implement both failure and success callbacks only if you have valid use cases for processing both. Because most publish calls typically succeed, processing successful event publishes likely isn't a concern. Also, a large volume of events can be published successfully, so be mindful about the performance and Apex limit impacts when processing the results in the `onSuccess` method.

```
public class FailureAndSuccessCallback implements EventBus.EventPublishFailureCallback,
    EventBus.EventPublishSuccessCallback {

    public void onFailure(EventBus.FailureResult result) {
        // Your implementation
        // Get event UUIDs from the result
        List<String> eventUuids = result.getEventUuids();
        // …
    }

    public void onSuccess(EventBus.SuccessResult result) {
        // Your implementation
        // Get event UUIDs from the result
        List<String> eventUuids = result.getEventUuids();
        // …
    }
}
```

SEE ALSO:

Identify and Match Event Messages with the EventUuid Field

Apex Reference Guide: System.Quiddity Enum: PLATFORM_EVENT_PUBLISH_CALLBACK

Callback Running User and Debug Logs

A publish callback runs under the Automated Process user. As a result, all records that are created in a callback have their system user fields, such as `CreatedById` and `OwnerId`, set to `Automated Process`.

You can explicitly set the `OwnerId` to another value. For example, to assign a task to a specific user, set the task `OwnerId` to that user's ID.

To collect debug logs for the callback's execution, add a trace flag for Automated Process. For more information, see Add a Trace Flag Entry for the Default Automated Process User in the *Platform Events Developer Guide*.

When the callback is invoked, it's logged in the debug log. Logging for the callback requires the System debug log level to be set to at least Info. For more information, see Set Up Debug Logging. For example, when the callback is invoked, the debug log line looks as follows.

```
CODE_UNIT_STARTED
[EXTERNAL]|platform.event.publish.callbacks.tasks.apex.ApexCallbackMethodInvoker
```

Create an Event with an EventUuid Field

The `EventUuid` field uniquely identifies an event message and is used to match the events returned in the callback result with the events in the publish call. To have the system generate an `EventUuid` field value in each event object, use the `SObjectType.newSObject(recordTypeId, loadDefaults)` Apex method to create the event object.

```
Order_Event__e event = (Order_Event__e)Order_Event__e.sObjectType.newSObject(null, true);

// The EventUuid value is returned after object creation
System.debug('EventUuid: ' + event.EventUuid);

// Debug output
// EventUuid: 19bd382e-8964-43de-ac01-d5d82dd0bf78
```

Note: If you aren't interested in correlating the events in the publish call with the publish results, you don't need the `EventUuid` value in the created event. In this case, you can create the event by using the event API name directly, which doesn't include the `EventUuid` value in the event object. For example: `Order_Event__e event = new Order_Event__e();`.

Invoke the Publish Callback

To have the system invoke the callback when the final publish result is available, pass in an instance of the callback class as the second parameter in the `EventBus.publish` call.

First, create an instance of the callback class. For example, we use the `FailureCallback` class that we implemented earlier.

```
FailureCallback cb = new FailureCallback();
```

This publish call publishes a list of events and passes in a callback instance.

```
List<Database.SaveResult> results = EventBus.publish(eventList, cb);
```

This publish call publishes one event and passes in a callback instance.

```
Database.SaveResult sr = EventBus.publish(myEvent, cb);
```

Callback Status Code

When you publish an event with a callback instance of `EventBus.EventPublishFailureCallback` or `EventBus.EventPublishSuccessCallback` and the publish call is successful, the returned `Database.SaveResult` contains a status code of `OPERATION_WITH_CALLBACK_ENQUEUED` in the `StatusCode` field of `Database.Error`. Also, the event universally unique identifier (UUID) is returned in the `Message` field.

```
Database.SaveResult[getErrors=(
  Database.Error[
  getFields=();
  getMessage=d473406e-0922-432a-9088-b8c95ef8b548;
  getStatusCode=OPERATION_WITH_CALLBACK_ENQUEUED;]
  );
  getId=e02xx0000000001AAA;
  isSuccess=true;]
```

If the Apex publish callbacks feature is disabled in your Salesforce org, the `EventBus.publish` calls that use callbacks still execute but don't invoke the callbacks. Also, the returned status code is `OPERATION_ENQUEUED`.

Publish Callback Best Practices

Keep in mind these best practices for publish callbacks when implementing this feature.

Don't Republish the Same Event Object That Is Created with SObjectType.newSObject

If you create an event object with the `SObjectType.newSObject(recordTypeId, loadDefaults)` Apex method, we recommend that you don't publish the same event object more than once. Because the event object is populated with an `EventUuid` value, if you publish it more than once, non-unique `EventUuid` values are tracked in the callbacks. The duplicate `EventUuid` values can cause unexpected results. This behavior doesn't apply to events that you create by using the API name `Event_Name__e` `event = new Event_Name__e()`.

Publish a List of Events Instead of Individual Events with a Callback

When using a callback in an `EventBus.publish` call and you want to publish several events, we recommend that you create a list of events and publish the events in one `EventBus.publish` call. Using one `EventBus.publish` call for all events is more efficient than making a call for each event because it uses less Apex governor limits for the publish call. Also, the system attempts to batch callback executions for a list of events.

This example creates a list of events and then passes it through the events variable to the `EventBus.publish` call. This snippet results in one call to the publish method with a callback instance.

```
// BEST PRACTICE
FailureCallback cb = new FailureCallback();
List<Order_Event__e> events =  new List<Order_Event__e)();

// Create events in a loop
for(Integer i = 0;i<10;i++) {
    events.add((Order_Event__e)Order_Event__e.sObjectType.newSObject(null, true));
}

// Pass the list of events to the publish call
EventBus.publish(events, cb);
```

In contrast, this example shows what to avoid. It's inefficiently making 10 calls to the publish method with a callback, each with one event. This example can result in more callback executions later than when events are batched in one publish call.

```
// !! NOT RECOMMENDED !!
FailureCallback cb = new FailureCallback();

// Create events individually and pass each event to the publish call
for(Integer i = 0;i<10;i++) {
    EventBus.publish((Order_Event__e)Order_Event__e.sObjectType.newSObject(null, true),
                     cb);
}
```

Publish a List of Events with a Callback with a Platform Event Type

If you create events by using the API name, you can publish a list of events with a callback only if you define the list with the specific platform event type. The generic `SObject` type isn't supported. For example, you can define a list of events as:

```
List<Order_Event__e> events =  new List<Order_Event__e>();
```

But not as:

```
List<SObject> events =  new List<SObject>();
```

Then you can publish the events with a callback.

```
events.add(new Order_Event__e());
EventBus.publish(events, myCallback);
```

Keep the Event UUID Map Size Small for Improved Performance

To reduce the callback instance size, keep the map of event UUIDs small in the callback. A small callback instance size ensures better performance and helps avoid hitting the cumulative usage limit of all publish callbacks. Map the event UUID to a record ID that you can query to populate the remaining event fields. Alternatively, if you want to save the entire event as the map value, make sure that the event doesn't have too many fields and the field sizes are small. For an example of how a map is used for republishing events in the `onFailure` method, see Example: Publish Callback Class That Correlates Callback Results with Event Messages.

SEE ALSO:

Apex Publish Callback Limits

Example: Publish Callback Class That Creates Follow-Up Tasks for Failed Publishes

This publish callback class creates a task when event publishing fails in the `onFailure` method. The inserted task includes the number of failed events and the event UUIDs.

```
public class FailureCallback implements EventBus.EventPublishFailureCallback {

    public void onFailure(EventBus.FailureResult result) {
        List<String> eventUuids = result.getEventUuids();
        System.debug(eventUuids.size() + ' events failed to publish.');
        System.debug('FailureCallback eventUuids to match with event objects: ' +
            eventUuids);

        // Create a follow-up task
```

```
        insertTask(eventUuids, false);
    }

    private void insertTask(List<String> eventUuids, Boolean isSuccess) {
        String eventIdString = '';
        for (String evtId : eventUuids) {
            eventIdString += evtId + ' ';
        }
        Task t = new Task();
        if (isSuccess == false) {
            t.Subject = 'Follow up on event publishing failures.';
            t.Description = eventUuids.size() +
                ' events failed to publish. Event UUIDs: '
            + eventIdString;
        }

        // Set the due date
        t.ActivityDate = Date.today().addDays(3);
        // Set owner ID explicitly.
        // Otherwise, the task assignee is the Automated Process User.
        // Change the user ID to a valid user ID in your org.
        t.OwnerId = '005RM000002QhQ1YAK';
        // Insert task
        Database.SaveResult sr = Database.insert(t);
        if (!sr.isSuccess()) {
            for(Database.Error err : sr.getErrors()) {
                System.debug('Error returned: ' +
                             err.getStatusCode() +
                             ' - ' +
                             err.getMessage());
            }
        }
    }
}
```

Example: Publish Events with a Callback Instance

To invoke the callback, perform an `EventBus.publish` call by passing it an instance of the `FailureCallback` class. You can publish one event or a batch of events with the callback.

This example publishes two event messages. This example requires a platform event, `Order Event`, to be defined with a Text(18) field of `Order Id`. To view debug logs for the `FailureCallback` class, make sure that you set up user trace flags for the Automated Process user. For more information, see Callback Running User and Debug Logs. In this case, if all publishing is successful, the `onFailure()` method isn't invoked.

```
List<Order_Event__e> eventList = new List<Order_Event__e>();

// Create event objects with prepopulated EventUuid fields.
Order_Event__e event1 = (Order_Event__e)Order_Event__e.sObjectType.newSObject(null, true);
event1.Order_Id__c='Order1 ID';
System.debug('event1 EventUuid: ' + event1.EventUuid);


Order_Event__e event2 = (Order_Event__e)Order_Event__e.sObjectType.newSObject(null, true);
```

```
event2.Order_Id__c='Order2 ID';
System.debug('event2 EventUuid: ' + event2.EventUuid);

// Add event objects to the list.
eventList.add(event1);
eventList.add(event2);

// Publish events with an instance of the failure callback.
List<Database.SaveResult> results = EventBus.publish(eventList, new FailureCallback());

// Inspect synchronous publishing result for each event.
for (Database.SaveResult sr : results) {
    if (sr.isSuccess()) {
        System.debug('Successfully published event.');
    } else {
        for(Database.Error err : sr.getErrors()) {
            System.debug('Error returned: ' +
                        err.getStatusCode() +
                        ' - ' +
                        err.getMessage());
        }
    }
}
```

Example: Publish Callback Class That Creates Follow-Up Tasks for Failed and Successful Publishes

This publish callback class is a modification of the previous example—it also implements the `EventBus.EventPublishSuccessCallback` interface and processes both success and failure cases. It creates a task when event publishing fails or succeeds. The inserted task includes the number of failed events and the event UUIDs.

Before running this example, change the email address in the example to an email address of a user who has permission to create tasks in your org. To view debug logs for the `FailureCallback` class, make sure that you set up user trace flags for the Automated Process user. For more information, see Callback Running User and Debug Logs.

```
public class FailureAndSuccessCallback implements EventBus.EventPublishFailureCallback,
    EventBus.EventPublishSuccessCallback {

    public void onFailure(EventBus.FailureResult result) {
        List<String> eventUuids = result.getEventUuids();
        System.debug(eventUuids.size() + ' events failed to publish.');
        System.debug('Callback eventUuids to match with event objects: ' + eventUuids);

        // Create a follow-up task for failed events.
        insertTask(eventUuids, false);
    }

    public void onSuccess(EventBus.SuccessResult result) {
        List<String> eventUuids = result.getEventUuids();
        System.debug(eventUuids.size() + ' events were published successfully.');
        System.debug('Callback eventUuids to match with event objects: ' + eventUuids);

        // Create a follow-up task for successful events.
        insertTask(eventUuids, true);
    }
```

```
    private void insertTask(List<String> eventUuids, Boolean isSuccess) {
        String eventIdString = '';
        for (String evtId : eventUuids) {
            eventIdString += evtId + ' ';
        }
        Task t = new Task();
        if (isSuccess == true) {
            t.Subject = 'Follow up on successful event publishing.';
            t.Description = eventUuids.size() +
                ' events published successfully. Event UUIDs: '
            + eventIdString;
        } else {
            t.Subject = 'Follow up on event publishing failures.';
            t.Description = eventUuids.size() +
                ' events failed to publish. Event UUIDs: '
            + eventIdString;
        }

        // Set the due date
        t.ActivityDate = Date.today().addDays(3);
        // Set owner ID explicitly.
        // Otherwise, the task assignee is the Automated Process User.
        // ---
        // CHANGE EMAIL ADDRESS to the email of a valid user in your org.
        // ---
        User myUser = [SELECT Id from User WHERE Email='user@example.com'];
        t.OwnerId = myUser.Id;
        // Insert task
        Database.SaveResult sr = Database.insert(t);
        if (!sr.isSuccess()) {
            for(Database.Error err : sr.getErrors()) {
                System.debug('Error returned: ' +
                             err.getStatusCode() +
                             ' - ' +
                             err.getMessage());
            }
        }
    }
}
```

To publish events with the callback class, use the code snippet in Example: Publish Events with a Callback Instance and change the callback instance name in the `EventBus.publish` method to `FailureAndSuccessCallback`.

```
// Publish events with an instance of the callback.
List<Database.SaveResult> results = EventBus.publish(eventList,
    new FailureAndSuccessCallback());
```

Example: Publish Callback Class That Correlates Callback Results with Event Messages

This example callback class implementation shows how to retry publishing failed events. It's based on a trigger on the Order object.

Callback Class

If event publishing fails, the `onFailure` method in the `FailureCallbackWithCorrelation` class is invoked. This method retries publishing failed events up to two times. A map holds the UUID values of each published event and maps it to the order record ID. This mapping is used to populate the event `Order_Id__c` field. Alternatively, you can use the record ID to obtain field data from the record and populate event fields. The example omits this detail for simplicity.

The examples in this section require a platform event, `Order Event`, to be defined with a Text(18) field of `Order Id`.

```
public class FailureCallbackWithCorrelation implements EventBus.EventPublishFailureCallback
{
    public static final Integer MAX_RETRIES = 2;
    private Integer retryCounter = 0;
    private Map<String,String> uuidMap;

    // Callback constructor
    public FailureCallbackWithCorrelation(Map<String,String> uuidMap) {
        this.uuidMap = uuidMap;
    }

    public void onFailure(EventBus.FailureResult result) {
        List<String> eventUuids = result.getEventUuids();
        Map<String,String> newUuidMap = new Map<String,String>();

        if (retryCounter < MAX_RETRIES) {
            // Try to re-publish the failed events
            List<Order_Event__e> events = new List<Order_Event__e>();
            for (String uuid : eventUuids) {
                // Create a new event with the contents of the failed event
                Order_Event__e event = (Order_Event__e)
                    Order_Event__e.sObjectType.newSObject(null, true);
                // Fill event with the right order record Id
                event.Order_Id__c = uuidMap.get(uuid);
                events.add(event);

                // Use a new map since the new event will have a different uuid
                newUuidMap.put(event.EventUuid, event.Order_Id__c);
            }
            // Replace old uuid map because we no longer need its contents
            uuidMap = newUuidMap;

            // Republish with the same callback passed in again as 'this'
            System.debug('Republish ' + eventUuids.size() + ' failed events.');
            EventBus.publish(events, this);
            System.debug('Republish event for Order with Ids: ' +
                         String.join(uuidMap.values(), ', '));

            // Increase counter
            retryCounter++;
        } else {
            // Retry exhausted, log an error instead
            System.debug(eventUuids.size() + ' event(s) failed to publish after ' +
                         MAX_RETRIES + ' retries ' +
                         'for Order with Ids: ' + String.join(uuidMap.values(), ', '));
        }
```

```
    }

    // Getter methods so we can validate this in the unit test
    public Integer getRetryCounter() {
        return retryCounter;
    }

    public Map<String,String> getUuidMap() {
        return uuidMap;
    }
}
```

Apex Trigger

For each inserted or updated order record, the trigger publishes the Order_Event__e platform event with a populated `EventUuid` field.

```
trigger OrderTrigger on Order (after insert, after update) {
    Map<String,String> uuidMap = new Map<String,String>();
    List<Order_Event__e> events = new List<Order_Event__e>();

    for (Order o : Trigger.new) {
        Order_Event__e e = (Order_Event__e)
        Order_Event__e.sObjectType.newSObject(null, true);
        // Fill event field with Order Id
        e.Order_Id__c = o.Id;
        // Map event UUID -> Order Id so we can look up later
        uuidMap.put(e.EventUuid, o.Id);
        events.add(e);
    }

    FailureCallbackWithCorrelation cb = new FailureCallbackWithCorrelation(uuidMap);
    List<Database.SaveResult> srs = EventBus.publish(events, cb);

    // Inspect immediate publish result
    for (Database.SaveResult sr : srs) {
        if (sr.isSuccess()) {
            System.debug('Successfully enqueued event.');
        } else {
            for(Database.Error err : sr.getErrors()) {
                System.debug('Error returned: ' + err.getStatusCode() + ' - ' +
                    err.getMessage());
            }
        }
    }
}
```

To run the trigger, insert an order record. Because an order depends on an account and contract, create these records first. You can create the records in the user interface or via Apex or an API. An Apex snippet is provided for your convenience. You can run this snippet in the Developer Console, in the Execute Anonymous Window.

```
// Create account
Account a = new Account();
a.Name = 'Account Callback';
```

```
insert a;

// Create contract
Contract c = new Contract();
c.StartDate = Date.today();
c.ContractTerm = 12;
c.Status = 'Draft';
c.AccountId = a.Id;
insert c;

// Create order
Order o = new Order();
o.AccountId = a.Id;
o.ContractId = c.Id;
o.Status = 'Draft';
o.EffectiveDate = Date.today();
insert o;
```

Test Apex Publish Callbacks

To test your Apex publish callback class, add an Apex test class. You must provide Apex tests before you can package or deploy an Apex class to production and meet code coverage requirements.

In an Apex test, event messages are published synchronously in the test event bus. To simulate the execution of the callback methods in a test, you can deliver or fail the publishing of the event messages.

To simulate a failed publishing of an event or a batch of events, call this method.

```
Test.getEventBus().fail();
```

The `Test.getEventBus().fail()` method causes the publishing of events to fail immediately after the call, and event messages are removed from the test event bus. This method causes the `onFailure()` method in the callback class to be invoked. When the event messages fail to publish, none of the triggers defined on the platform event receive any failed events.

To simulate successful event delivery, call the `Test.getEventBus().deliver();` method or have your events delivered after `Test.stopTest()`. Event messages are delivered immediately after each of those statements. Successful event delivery triggers the execution of the `onSuccess()` method in the callback class.

Example: MyCallbackTest Test Class

This example class is a test class for the `FailureAndSuccessCallback` class given previously. This test class shows how to test the successful and failed publishing of test event messages in the test event bus. Before you run this test class, define a platform event in Setup with the label `Order Event` and a Text(18) field of `Order Id`.

```
@isTest
public class MyCallbackTest {
    @isTest static void testFailedEventsWithFail() {

        // Publish with callback
        FailureAndSuccessCallback cb = new FailureAndSuccessCallback();

        // Create test event with EventUuid field value
        Order_Event__e event = (Order_Event__e)Order_Event__e.sObjectType.newSObject(
            null, true);
        event.Order_Id__c='100';
```

```
        System.debug('EventUuid of created event: ' + event.EventUuid);
        // Publish an event with callback
        EventBus.publish(event, cb);

        // Fail event
        // (invoke onFailure and DO NOT deliver event to subscribers)
        Test.getEventBus().fail();

        // Verify that tasks were created by the onFailure() method
        List<Task> tasksFailed =
            [SELECT Id,Subject,Description FROM Task
             WHERE Subject='Follow up on event publishing failures.'];
        System.Assert.areEqual(1,tasksFailed.size(),
                               'Unexpected number of tasks received for failed publishing');

        System.debug('tasksFailed[0].Description=' + tasksFailed[0].Description);
        System.debug('event.EventUuid=' + event.EventUuid);
        System.Assert.isTrue(tasksFailed[0].Description.contains(event.EventUuid),
                               'EventUuid was not found in the Description field.');

    }

    @isTest static void testSuccessfulEventsWithDeliver() {

        // Publish with callback
        FailureAndSuccessCallback cb = new FailureAndSuccessCallback();

        // Create test event with EventUuid field value
        Order_Event__e event = (Order_Event__e)Order_Event__e.sObjectType.newSObject(
            null, true);
        event.Order_Id__c='99';
        // Publish an event with callback
        EventBus.publish(event, cb);

        // Deliver events published so far
        // (invokes onSuccess and delivers events to subscribers)
        Test.getEventBus().deliver();

        // Verify that tasks were created by the onSuccess() method
        List<Task> tasksSuccessful =
            [SELECT Id,Subject,Description FROM Task
             WHERE Subject='Follow up on successful event publishing.'];
        System.Assert.areEqual(1,tasksSuccessful.size(),
                               'Unexpected number of tasks received for successful publishing');

        System.Assert.isTrue(tasksSuccessful[0].Description.contains(event.EventUuid),
                               'EventUuid was not found in the Description field.');

    }

    @isTest static void testSuccessfulEventsWithStopTest() {

        // Start test
        Test.startTest();
```

```
        // Publish with callback
        FailureAndSuccessCallback cb = new FailureAndSuccessCallback();

        // Create test event with EventUuid field value
        Order_Event__e event = (Order_Event__e)Order_Event__e.sObjectType.newSObject(
            null, true);
        event.Order_Id__c='99';
        // Publish an event with callback
        EventBus.publish(event, cb);

        // After the test ends, it delivers the events published
        // (invokes onSuccess and delivers to subscribers)
        Test.stopTest();

        // Verify that we have two tasks created by the onSuccess() method:
        // one task from the earlier deliver() call and one event after Test.stopTest()
        List<Task> tasksSuccessful =
            [SELECT Id,Subject,Description FROM Task
             WHERE Subject='Follow up on successful event publishing.'];
        System.Assert.areEqual(1,tasksSuccessful.size(),
                        'Unexpected number of tasks received for successful publishing');

        System.Assert.isTrue(tasksSuccessful[0].Description.contains(event.EventUuid),
                             'EventUuid was not found in the Description field.');
    }
}
```

Example: MyCallbackTestWithCorrelation Test Class

This example class is a test class for the `FailureCallbackWithCorrelation` class given previously. This test class shows how to test the failed publishing of test event messages in the test event bus. The callback retries publishing events for a maximum of two attempts so the test fails the publishing of a test event twice in a loop. It verifies that, each time, the callback retries publishing the event by checking that the `retryCounter` variable has been increased. Before you run this test class, define a platform event in Setup with the label `Order Event` and a Text(18) field of `Order Id`.

```
@isTest
public class MyCallbackTestWithCorrelation {

    @isTest
    static void testFailedEventsWithFail() {
        // Create test event
        Order_Event__e event = (Order_Event__e)Order_Event__e.sObjectType.newSObject(
            null, true);
        event.Order_Id__c='dummyOrderId';

        // Populate map
        Map<String,String> uuidMap = new Map<String,String>();
        uuidMap.put(event.EventUuid, 'dummyOrderId');

        // Create callback
        FailureCallbackWithCorrelation cb = new FailureCallbackWithCorrelation(uuidMap);

        // Make sure retry counter is 0
```

```
        Assert.areEqual(0, cb.getRetryCounter(),
            'Newly created callback should have retry counter at 0');

        // Publish an event with callback
        EventBus.publish(event, cb);

        // If we fail all publish attempts, callback should run MAX_RETRIES times.
        // For each attempt, the callback should republish the event,
        //    increase the counter, and update the map
        String prevUuid = event.EventUuid;
        for (Integer i = 1; i <= FailureCallbackWithCorrelation.MAX_RETRIES; i++) {
            Test.getEventBus().fail();
            Assert.areEqual(i, cb.getRetryCounter(), 'Retry counter should be ' + i);
            Assert.areEqual(1, cb.getUuidMap().size(), 'Map size should be 1');
            String currUuid = (new List<String>(cb.getUuidMap().keySet())).get(0);
            Assert.areNotEqual(prevUuid, currUuid,
                'Map should be updated with newly created event Uuid');
            Assert.areEqual('dummyOrderId', cb.getUuidMap().get(currUuid),
                'Map value should be the original Order Id');
            prevUuid = currUuid;
        }

        // If we publish another failed event, callback should not retry.
        Order_Event__e event2 = (Order_Event__e)Order_Event__e.sObjectType.newSObject(
            null, true);
        event2.Order_Id__c='dummyOrderId';
        EventBus.publish(event, cb);
        Test.getEventBus().fail();
        Assert.areEqual(FailureCallbackWithCorrelation.MAX_RETRIES, cb.getRetryCounter(),

                        'Retry counter should still be ' +
                         FailureCallbackWithCorrelation.MAX_RETRIES);
    }
}
```

SEE ALSO:

Platform Events Developer Guide: Testing Your Platform Event in Apex

Apex Publish Callback Limits

Keep in mind this limit for Apex publish callbacks.

Description	Limit
Maximum cumulative usage of all publish callbacks in the last 30 minutes	5 MB
Maximum number of times a publish callback method implementation can call `EventBus.publish` with a callback recursively.	10

The publish callback size used in the callback allocation is the size of the objects contained in a callback class instance, such as the class variable objects. It isn't the length of the Apex class in characters. For example, in the `FailureCallbackWithCorrelation` class in Example: Publish Callback Class That Correlates Callback Results with Event Messages, the objects that contribute to the size counted in the allocation are these class variables: `MAX_RETRIES`, `retryCounter`, and `uuidMap`. The cumulative usage is the sum of the sizes of callback instances that were executed in the last 30 minutes. If you hit the callback size limit, try to reduce the size of the objects stored in your callback class through the class variables. Alternatively, limit the number of retried callback executions or wait before using callbacks again. The callback limit is a rolling limit and counts usage in the last 30 minutes, so usage can decrease after some time has passed. Usage is updated every time you publish an event with a callback.

To monitor the usage of all publish callbacks in the last 30 minutes, make a REST API call to the limits resource, and inspect the `PublishCallbackUsageInApex` value in the returned response. Make a GET request to:

```
/services/data/v58.0/limits
```

The `PublishCallbackUsageInApex` value in the returned response looks similar to this example. The `PublishCallbackUsageInApex` value returns the maximum usage and the remaining usage size in bytes.

```
"PublishCallbackUsageInApex" : {
    "Max" : 5242880,
    "Remaining" : 4011396
  }
```

SEE ALSO:

REST API Developer Guide: Limits

REST API Developer Guide: List Organization Limits

Publish Event Messages with Salesforce APIs

External apps use an API to publish platform event messages.

Publish events by inserting events in the same way that you insert sObjects. You can use any Salesforce API to create platform events, such as SOAP API, REST API, or Bulk API 2.0.

When publishing an event message, the result that the API returns contains information about whether the operation was successful and the errors encountered. If the `success` field is `true`, the publish request is queued in Salesforce and the event message is published asynchronously. For more details, see High-Volume Platform Event Persistence. If the `success` field is `false`, the event publish operation resulted in errors, which are returned in the `errors` field.

The returned result also contains the `Id` system field. The `Id` field value isn't included in the event message delivered to subscribers. It isn't used to identify an event message, and it isn't always unique. Subscribers can use the `ReplayId` system field, which is included in the delivered message, to identify the position of the event in the stream.

Status Code Returned for Asynchronous Publishing

To indicate that the publish operation is asynchronous, the `OPERATION_ENQUEUED` status code is returned for a successful call in the response's error field, in addition to the event UUID. This example response shows the `statusCode` field containing `OPERATION_ENQUEUED` and the message field containing the event UUID.

```
HTTP/1.1 201 Created

{
  "id" : "e01xx0000000001AAA",
  "success" : true,
```

```
  "errors" : [ {
    "statusCode" : "OPERATION_ENQUEUED",
    "message" : "232fd30e-0a71-42bd-a97b-be0e329b2ded",
    "fields" : [ ]
  } ]
}
```

The examples in the next sections are based on a high-volume platform event.

REST API

To publish a platform event message using REST API, send a POST request to this endpoint.

```
/services/data/v61.0/sobjects/Event_Name__e/
```

Example: If you defined a platform event named `Low Ink`, publish event notifications by inserting `Low_Ink__e` data. This example creates one event of type `Low_Ink__e` in REST API.

REST endpoint:

```
/services/data/v61.0/sobjects/Low_Ink__e/
```

Request body:

```
{
   "Printer_Model__c" : "XZO-5"
}
```

After the platform event message is published, the REST response looks like this output. Headers are deleted for brevity.

```
HTTP/1.1 201 Created

{
  "id" : "e01xx0000000001AAA",
  "success" : true,
  "errors" : [ {
    "statusCode" : "OPERATION_ENQUEUED",
    "message" : "232fd30e-0a71-42bd-a97b-be0e329b2ded",
    "fields" : [ ]
  } ]
}
```

REST API Composite Resource

To publish multiple platform event messages in one REST API request, use the `composite` resource. Send a POST request to this endpoint.

```
/services/data/v61.0/composite/
```

Add each platform event as a subrequest in the composite request body.

Example: This composite request contains two platform events in the request body.

```
{
  "allOrNone": true,
  "compositeRequest": [
```

```
    {
      "method": "POST",
      "url": "/services/data/v61.0/sobjects/Low_Ink__e",
      "referenceId": "event1",
      "body": {
        "Serial_Number__c" : "1000",
        "Printer_Model__c" : "XZO-5"
      }
    },
    {
      "method": "POST",
      "url": "/services/data/v61.0/sobjects/Low_Ink__e",
      "referenceId": "event2",
      "body": {
        "Serial_Number__c" : "1001",
        "Printer_Model__c" : "XY-10"
      }
    }
  ]
}
```

After the platform event messages are published, the REST response looks like this output. Headers are deleted from this sample response.

```
{
  "compositeResponse" : [ {
    "body" : {
      "id" : "e01xx0000000001AAA",
      "success" : true,
      "errors" : [ {
        "statusCode" : "OPERATION_ENQUEUED",
        "message" : "436ccd6f-cc43-4861-a260-a3ffbc1bc27c",
        "fields" : [ ]
      } ]
    },
    "httpStatusCode" : 201,
    "referenceId" : "event1"
  }, {
    "body" : {
      "id" : "e01xx0000000001AAA",
      "success" : true,
      "errors" : [ {
        "statusCode" : "OPERATION_ENQUEUED",
        "message" : "85d962fb-f05c-4ccf-9ee1-ac751d0fc07f",
        "fields" : [ ]
      } ]
    },
    "httpStatusCode" : 201,
    "referenceId" : "event2"
  } ]
}
```

Note: The `allOrNone` header in the composite REST request and in SOAP API applies only to platform events defined with the Publish After Commit option. For more information, see Platform Events and Transactions.

SOAP API

To publish a platform event message using SOAP API, use the `create()` call.

Example: This example shows the SOAP message (using Partner API) of a request to create three platform event messages in one call. Each event has one custom field named `Printer_Model__c`.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="urn:sobject.partner.soap.sforce.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:partner.soap.sforce.com">
<SOAP-ENV:Header>
    <ns2:SessionHeader>
       <ns2:sessionId>00DR00000001fWV!AQMAQOshATCQ4fBaYFOTrHVixfEO6l...</ns2:sessionId>

    </ns2:SessionHeader>
    <ns2:CallOptions>
        <ns2:client>ClientApp/34.0.12i</ns2:client>
        <ns2:defaultNamespace xsi:nil="true"/>
        <ns2:returnFieldDataTypes xsi:nil="true"/>
    </ns2:CallOptions>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
    <ns2:create>
        <ns2:sObjects>
            <ns1:type>Low_Ink__e</ns1:type>
            <ns1:fieldsToNull xsi:nil="true"/>
            <ns1:Id xsi:nil="true"/>
            <Printer_Model__c>XZO-600</Printer_Model__c>
        </ns2:sObjects>
        <ns2:sObjects>
            <ns1:type>Low_Ink__e</ns1:type>
            <ns1:fieldsToNull xsi:nil="true"/>
            <ns1:Id xsi:nil="true"/>
            <Printer_Model__c>XYZ-100</Printer_Model__c>
        </ns2:sObjects>
        <ns2:sObjects>
            <ns1:type>Low_Ink__e</ns1:type>
            <ns1:fieldsToNull xsi:nil="true"/>
            <ns1:Id xsi:nil="true"/>
            <Printer_Model__c>XYZ-9000</Printer_Model__c>
        </ns2:sObjects>
    </ns2:create>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The response of the Partner SOAP API request looks something like this response. Headers are deleted for brevity.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="urn:partner.soap.sforce.com">
<soapenv:Header>
 ...
</soapenv:Header>
```

```
<soapenv:Body>
    <createResponse>
        <result>
            <id>e00xx000000000F</id>
            <success>true</success>
            <errors>
               <message>04b8724e-e7e7-4caf-9bcd-0d14c9f97e31</message>
               <statusCode>OPERATION_ENQUEUED</statusCode>
            </errors>
        </result>
        <result>
            <id>e00xx000000000G</id>
            <success>true</success>
            <errors>
               <message>7378b9cc-d381-4150-b093-336e3a0e4018</message>
               <statusCode>OPERATION_ENQUEUED</statusCode>
            </errors>
        </result>
        <result>
            <id>e00xx000000000H</id>
            <success>true</success>
            <errors>
               <message>32da1ef3-6877-485a-8dde-1174f589e31a</message>
               <statusCode>OPERATION_ENQUEUED</statusCode>
            </errors>
        </result>
    </createResponse>
</soapenv:Body>
</soapenv:Envelope>
```

SEE ALSO:

- REST API Developer Guide
- REST API Developer Guide : Using Composite Resources
- SOAP API Developer Guide: `create()` call
- Bulk API 2.0 Bulk API Developer Guide
- Platform Event Error Status Codes

Publish Event Messages with Pub/Sub API

Use Pub/Sub API to publish platform event messages from an external app and get final publish results. Simplify your development by using one API to publish, subscribe, and retrieve the event schema. Based on gRPC and HTTP/2, Pub/Sub API enables efficient publishing of binary event messages in the Apache Avro format.

The Pub/Sub API service is defined in a proto file, with RPC method parameters and return types specified as protocol buffer messages. When an event is published through one of the publish RPC methods, the publish request is serialized based on the protocol buffer message type. For more information, see What is gRPC? and Protocol Buffers in the gRPC documentation, and pubsub_api.proto in the Pub/Sub API GitHub repository.

Publish events by using one of two RPC methods: `Publish` and `PublishStream`.

The `Publish` RPC method is a unary RPC, which means that it sends only one request and receives only one response.

```
rpc Publish (PublishRequest) returns (PublishResponse);
```

The `PublishStream` RPC method uses bidirectional streaming. It can send a stream of publish requests while receiving a stream of publish responses from the server. Use `PublishStream` to achieve a higher publish rate than with `Publish`.

```
rpc PublishStream (stream PublishRequest) returns (stream PublishResponse);
```

The `PublishResponse` holds a `PublishResult` for each event published that indicates the final success or failure of the publish operation, and not the intermediate queueing results. A successful status means that the event was published. A failed status means that the event failed to publish, and the client can retry publishing this event.

To learn more about the RPC methods in Pub/Sub API, see Pub/Sub API RPC Method Reference in the *Pub/Sub API Developer Guide*.

Write a Pub/Sub API client to publish platform event messages. You can use one of the 11 supported programming languages, including Python, Java, Go, and Node. To learn how to write a client in Java or Python, check out Quick Starts in the *Pub/Sub API Developer Guide*. For code examples in other languages, see the Pub/Sub API GitHub repository.

Subscribing to Platform Events

Receive platform events in processes, flows, Apex triggers, Pub/Sub API, or CometD clients.

IN THIS SECTION:

Subscribe to Platform Event Notifications with CometD

Use CometD to subscribe to platform events in an external client.

Group Platform Events into One Stream with a Custom Channel

With a custom channel, you can receive a stream of event messages corresponding to one or more custom platform events, or Real-Time Event Monitoring events. For example, if you've defined platform events corresponding to orders for different regions, one client can subscribe to all those events and process them. Custom channels are supported in Pub/Sub API clients, CometD clients, and event relays only. You can also add filters to custom channels. By using only one client to subscribe to all events and using filters, your subscriptions are optimized.

Filter Your Stream of Platform Events with Custom Channels

Receive only the event messages that match a predefined filter on a custom channel. Create a channel, and configure it with a filter expression. Pub/Sub API and Streaming API (CometD) subscribers to the channel receive a filtered stream of events. With fewer events delivered to subscribers, event processing is optimized. Also, subscribers make more efficient use of the event delivery allocation.

Obtain a Platform Event's Subscribers

View a list of all triggers or processes that are subscribed to a platform event by using the Salesforce user interface or the API.

Identify and Match Event Messages with the EventUuid Field

Delivered platform event messages include the `EventUuid` field, which identifies an event message. Use this field to match published and received event messages by comparing the universally unique identifiers (UUIDs) of the received events with the UUIDs returned in the SaveResult of publish calls. This way, you can find any event messages that aren't delivered and republish them.

SEE ALSO:

Decoupled Publishing and Subscription

Set Up Debug Logs for Event Subscriptions

Debug logs for platform event triggers, event processes, and resumed flow interviews are created by Automated Process and are separate from their corresponding Apex code logs. For a platform event trigger with an overridden running user, debug logs are created by the specified user. The debug logs aren't available in the Developer Console's Log tab.

IN THIS SECTION:

Add a Trace Flag Entry for the Default Automated Process User

To collect logs for an event subscription, add a trace flag entry for the Automated Process entity in Setup.

Add a Trace Flag Entry for the Overridden User

To collect logs for an Apex trigger whose default running user is overridden, add a trace flag entry for the user in Setup.

SEE ALSO:

Salesforce Help: Set Up Debug Logging

Add a Trace Flag Entry for the Default Automated Process User

To collect logs for an event subscription, add a trace flag entry for the Automated Process entity in Setup.

1. From Setup, in the Quick Find box, enter *`Debug Logs`*, then click **Debug Logs**.

2. Click **New**.
3. For Traced Entity Type, select **Automated Process**.
4. Select the time period to collect logs. The start and expiration dates default to the current date and time. To extend the expiration date, click the end date input box, and select the next day from the calendar.
5. For Debug Level, click **New Debug Level**. Enter a name, such as `CustomDebugLevel`, and accept the defaults.
6. Click **Save**.

To collect logs for the user who publishes the events, add another trace flag entry for that user.

Add a Trace Flag Entry for the Overridden User

To collect logs for an Apex trigger whose default running user is overridden, add a trace flag entry for the user in Setup.

1. From Setup, in the Quick Find box, enter `Debug Logs`, then click **Debug Logs**.
2. Click **New**.
3. Keep the Traced Entity Type value of `User`.
4. For Traced Entity Name, click the Lookup button, search for the user in the Lookup window, and select it.
5. Select the time period to collect logs. The start and expiration dates default to the current date and time. To extend the expiration date, click the end date input box, and select the next day from the calendar.
6. For Debug Level, click **New Debug Level**. Enter a name, such as `CustomDebugLevel`, and accept the defaults.
7. Click **Save**.

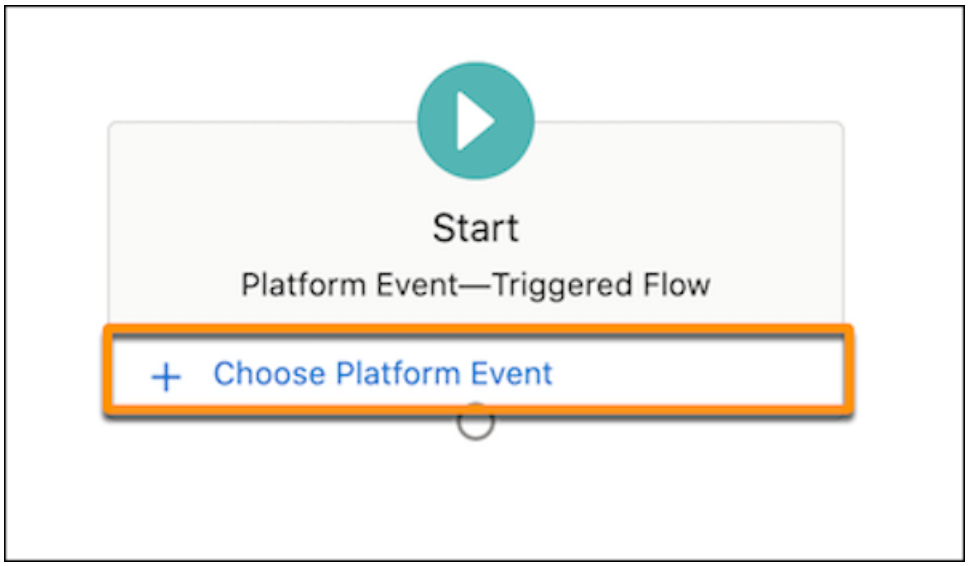
To collect logs for the user who publishes the events, add another trace flag entry for that user.

Subscribe to Platform Event Messages with Flows

Launch flows or resume running instances of flows, called interviews, when platform event messages are received. Subscribed flows and interviews can receive event messages published through Apex, APIs, flows, and other processes. Flows and interviews provide an autosubscription mechanism.

Launch a Flow When a Platform Event Message Is Received

Create a platform event–triggered flow. From the Start element, choose a platform event whose event messages trigger the flow to run.



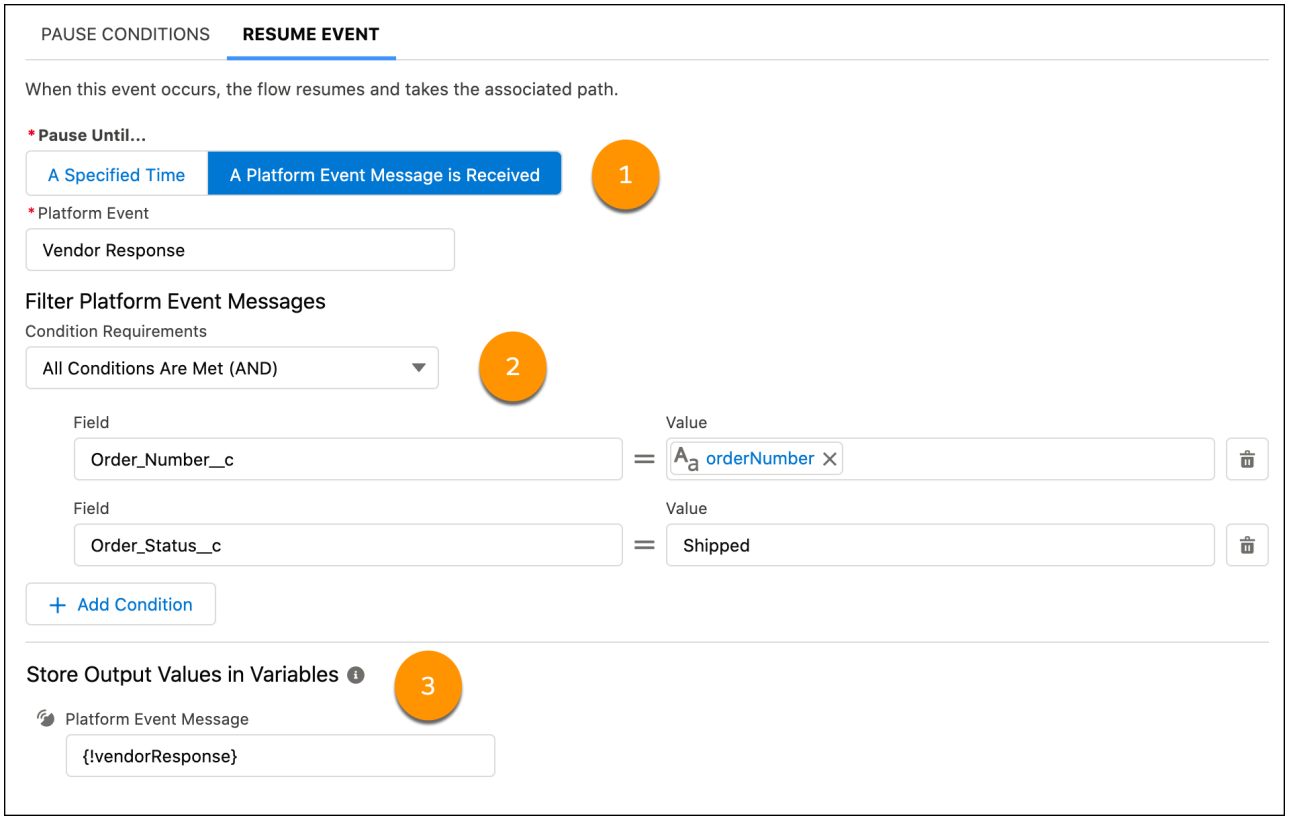
As you build the flow, you can use the field values from the platform event message by referencing the `$Record` global variable.

Resume a Flow When a Platform Event Message Is Received

To configure an autolaunched flow to subscribe to a platform event at run time, add a Pause element and set it up as follows.

- (Optional) Specify conditions that determine whether to pause a flow interview.
- Select the platform event that the flow interview subscribes to.
- Identify the values that a received event message must have to resume the flow interview.
- (Optional) Create a record variable in the flow to store the data from the event message that resumes the flow interview.

Example: This Pause element is set up to resume a flow interview when a vendor response event message is received (1). The order number in the event message must match the flow's `orderNumber` variable value, and the order status must be *`Shipped`* (2). When the flow interview resumes, the `vendorResponse` record variable is populated with the data from the event message (3).



Flow and Platform Event Considerations

If platform event–triggered flows, paused flow interviews, and processes are subscribed to the same platform event, we can't guarantee which one processes each event message first.

Platform event–triggered flows and flow interviews evaluate platform event messages in the order they're received. The order of event messages is based on the event replay ID. A flow can receive a batch of event messages at once, up to a maximum of 2,000 event messages. The order of event messages is preserved within each batch. The event messages in a batch can originate from multiple publishers.

Each platform event–triggered flow or resumed flow interview runs asynchronously in a separate transaction from the transaction that published the event message. As a result, there can be a delay between when an event message is published and when the subscribed flow or interview evaluates the event message.

Debug logs for platform event–triggered flows and resumed flow interviews appear under the Automated Process user. But each flow interview runs in the context of the user who published the event message. So, for example, if a flow interview creates or updates records, system fields like `CreatedById` and `LastModifiedById` reference the user who published the event message.

SEE ALSO:

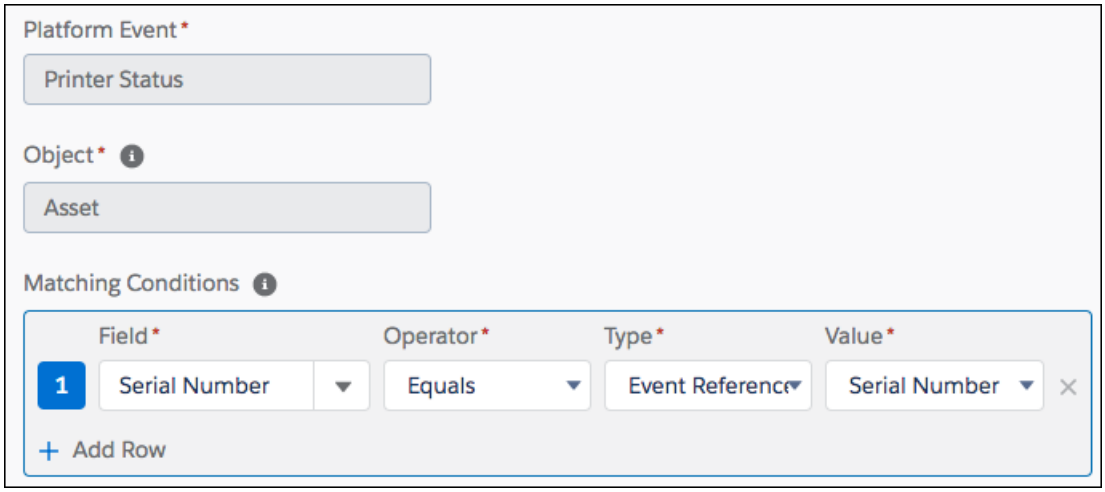
- Considerations for Subscribing to Platform Events with Processes and Flows
- *Salesforce Help*: Flow Limits and Considerations
- *Salesforce Help*: Paused Flow Interview Considerations
- End-to-End Example: Printer Supply Automation

Subscribe to Platform Event Messages with Processes

Processes built in Process Builder can subscribe to platform events and receive event messages published through Apex, APIs, flows, and other processes. Processes provide an autosubscription mechanism.

To subscribe a process to a platform event, build the process to start when it receives a platform event message. In the process's trigger, associate the process with a platform event and an object.

Example: This process starts when it receives a Printer Status event message. When it starts, the process looks for an Asset record whose serial number matches the serial number in the event message.



Process and Platform Event Considerations

If platform event–triggered flows, paused flow interviews, and processes are subscribed to the same platform event, we can't guarantee which one processes each event message first.

A process evaluates platform event messages in the order they're received. The order of event messages is based on the event replay ID. A process can receive a batch of event messages at once, up to a maximum of 2,000 event messages. The order of event messages is preserved within each batch. The event messages in a batch can originate from multiple publishers.

Each event process runs asynchronously in a separate transaction from the transaction that published the event message. As a result, there can be a delay between when an event message is published and when the subscribed flow or interview evaluates the event message.

Debug logs corresponding to the process execution appear under the Automated Process user. But the process actions run in the context of the user who published the event message. So, for example, if a process creates or updates records, system fields like `CreatedById` and `LastModifiedById` reference the user who published the event message.

All processes are subject to entitlements, limits, and other considerations, including Apex governor limits.

SEE ALSO:

Salesforce Help: Process Limits and Considerations

Considerations for Subscribing to Platform Events with Processes and Flows

Set Up Debug Logs for Event Subscriptions

Obtain Processes That Subscribe to a Platform Event in Metadata API

Subscribe to Platform Event Messages with Flows

Subscribe to Platform Event Notifications with Apex Triggers

Use Apex triggers to subscribe to events. You can receive event notifications in triggers regardless of how they were published—through Apex or APIs. Triggers provide an autosubscription mechanism. No need to explicitly create and listen to a channel in Apex.

To subscribe to event notifications, write an `after insert` trigger on the event object type. The `after insert` trigger event corresponds to the time after a platform event is published. After an event message is published, the `after insert` trigger is fired.

Example: This example shows a trigger for the `Low Ink` event. It iterates through each event and checks the `Printer_Model__c` field value. The trigger inspects each received notification and gets the printer model from the notification. If the printer model matches a certain value, other business logic is executed. For example, the trigger creates a case to order a new cartridge for this printer model.

```
// Trigger for catching Low_Ink events.
trigger LowInkTrigger on Low_Ink__e (after insert) {
    // List to hold all cases to be created.
    List<Case> cases = new List<Case>();

    // Get user Id for case owner. Replace username value with a valid value.
    User adminUser = [SELECT Id FROM User WHERE Username='admin@acme.org'];

    // Iterate through each notification.
    for (Low_Ink__e event : Trigger.New) {
        System.debug('Printer model: ' + event.Printer_Model__c);
        if (event.Printer_Model__c == 'MN-123') {
            // Create Case to order new printer cartridge.
            Case cs = new Case();
            cs.Priority = 'Medium';
            cs.Subject = 'Order new ink cartridge for SN ' + event.Serial_Number__c;
            // Optional: Set case owner ID so it is not Automated Process.
            // This step is not needed if the running user is overridden
            // or if using assignment rules.
            cs.OwnerId = adminUser.Id;
            cases.add(cs);
        }
    }

    // Insert all cases in the list.
    if (cases.size() > 0) {
        insert cases;
    }
}
```

An Apex trigger processes platform event notifications sequentially in the order they're received. The order of events is based on the event replay ID. An Apex trigger can receive a batch of events at once. The maximum batch size in a platform event trigger is 2,000 event messages. The order of events is preserved within each batch. The events in a batch can originate from one or more publishers.

Unlike triggers on standard or custom objects, triggers on platform events don't execute in the same Apex transaction as the one that published the event. The trigger runs asynchronously in its own process. As a result, there can be a delay between when an event is published and when the trigger processes the event.

The trigger runs under the Automated Process entity or the user you select in the trigger configuration. If no user is configured, debug logs corresponding to the trigger execution are created by Automated Process. System fields, such as `CreatedById` and `LastModifiedById`, reference the Automated Process entity. You can override the trigger's default running user so that the user for debug logs and records is set to the selected user. For more information, see Configure the User and Batch Size for Your Platform Event Trigger with PlatformEventSubscriberConfig.

Note: The `OwnerId` field of records saved in the trigger is set to the trigger's running user. By default, it's Automated Process. For more information on how to change the OwnerId, see Considerations for Publishing and Subscribing to Platform Events with Apex and APIs.

Event triggers have many of the same limitations of custom and standard object triggers. For example, with some exceptions, you generally can't make Apex callouts from triggers. For more information, see Implementation Considerations for triggers in the *Apex Developer Guide*.

Platform Event Triggers and Uncaught Exceptions

If an uncaught exception occurs during trigger execution, the trigger stops executing and doesn't process the remaining event messages in the current batch. Uncaught exceptions are exceptions that the trigger doesn't handle in a catch block or limit exceptions. As long as the trigger hasn't exceeded the Apex execution-time limit, the DML operations that were carried out before the uncaught exception are committed and aren't rolled back. Committing the DML transactions enables you to use the `setResumeCheckpoint()` method to continue trigger execution from where it left off. With this method, the trigger resumes and picks up the unprocessed event messages from the previous batch. For more information, see Resume a Platform Event Trigger After an Uncaught Exception.

DML transactions are rolled back only when:

- The trigger throws the `EventBus.RetryableException`.
- The trigger exceeds the Apex execution-time limit of 10 minutes. See Maximum execution time for each Apex transaction in Execution Governors and Limits in the *Apex Developer Guide*.

Platform Event Triggers and Apex Governor Limits

Platform event triggers are subject to Apex governor limits.

Synchronous Governor Limits

When governor limits are different for synchronous and asynchronous Apex, the synchronous limits apply to platform event triggers. Asynchronous limits are for long-lived processes, such as Batch Apex and future methods. Synchronous limits are for short-lived processes that execute quickly. Although platform event triggers run asynchronously, they're short-lived processes that execute in batches rather quickly.

Reset Limits

Because a platform event trigger runs in a separate transaction from the one that fired it, governor limits are reset, and the trigger gets its own set of limits.

IN THIS SECTION:

SEE ALSO:

Apex Developer Guide: Execution Governors and Limits
Set Up Debug Logs for Event Subscriptions
View and Manage an Event's Subscribers on the Platform Event's Detail Page
Considerations for Publishing and Subscribing to Platform Events with Apex and APIs

Configure the User and Batch Size for Your Platform Event Trigger with PlatformEventSubscriberConfig

You can override the default running user and batch size of a platform event Apex trigger. By default, the trigger runs as the Automated Process system user with a batch size of 2,000 event messages. Configuring the user and batch size enables you to bypass some limitations that sometimes arise from using the defaults. Use PlatformEventSubscriberConfig in Tooling API or Metadata API to configure the trigger.

Running the trigger as a specific user instead of the default Automated Process entity has these benefits:

- Records are created, modified, and deleted as this user.
- `OwnerId` fields of created records are populated to this user.
- Records are shared with the user when sharing is enabled. For example, when the trigger calls into an Apex class declared with the `with sharing` keywords.

- Debug logs for the trigger execution are created by this user.
- You can send email messages from the trigger, which isn't supported with the default Automated Process user.

You can specify any active user in the Salesforce org. The trigger runs in system context with privileges to access all records regardless of the user's object and field-level permissions. Record sharing is enforced for the running user when the trigger calls into an Apex class declared with the `with sharing` keywords.

In addition to setting a user, you can specify a custom batch size from 1 through 2,000. The batch size is the maximum number of event messages that can be sent to a trigger in one execution. For platform event triggers, the default batch size is 2,000. Setting a smaller batch size can help avoid hitting Apex governor limits.

Note:

- We don't recommend setting the batch size to 1 to process one event at a time. Small batch sizes can slow down the processing of event messages.
- If a trigger is running and subscribed to a platform event, new configuration settings take effect after you suspend and resume the trigger. You can suspend and resume a trigger from the platform event detail page by clicking **Manage** next to the Apex trigger in the Subscriptions related list. For more information, see View and Manage an Event's Subscribers on the Platform Event's Detail Page.

To configure a platform event trigger with Tooling API, see PlatformEventSubscriberConfig in the *Tooling API Developer Guide*. To add a configuration, perform a POST with the PlatformEventSubscriberConfig REST resource, and perform a GET call to retrieve a configuration by ID. Also, you can query the configurations using Tooling API.

To configure a platform event trigger with Metadata API, see PlatformEventSubscriberConfig in the *Metadata API Developer Guide*. You can use Visual Studio Code with the Salesforce Extension pack to deploy and retrieve Metadata API. For more information about installing Visual Studio Code and the extension pack, see Salesforce Extensions for Visual Studio Code. For more information about deploying and retrieving metadata using the CLI, see source Commands and mdapi Commands in the *Salesforce CLI Command Reference*.

SEE ALSO:

Apex Developer Guide: Apex Security and Sharing

Apex Developer Guide: Execution Governors and Limits

Find Uncaught Exceptions in Event Log Files

If an unhandled exception occurs during the execution of your platform event Apex trigger, you can get information about the exception using event log files in Event Monitoring. Examples of an unhandled exception include an uncatchable limit exception or an exception that the trigger doesn't catch.

To retrieve information about the unhandled exception, query `EventLogFile`. The event type for unhandled exceptions is Apex Unexpected Exception Event Type (`ApexUnexpectedException`).

SEE ALSO:

Object Reference for the Salesforce Platform: EventLogFile

Object Reference for the Salesforce Platform: Apex Unexpected Exception Event Type

Trailhead: Event Monitoring

Resume a Platform Event Trigger After an Uncaught Exception

Set a checkpoint in the event stream for where the platform event trigger resumes execution in a new invocation. If an Apex governor limit is hit or another uncaught exception is thrown, the checkpoint is used during the next execution of the trigger. Trigger processing resumes after the last successfully checkpointed event message. You can also set a checkpoint to explicitly control the number of events processed in one trigger execution. However, you can configure the trigger batch size more easily using Metadata API or Tooling API. For more information, see Configure the User and Batch Size for Your Platform Event Trigger with PlatformEventSubscriberConfig.

By processing fewer event messages, your trigger is less likely to hit Apex governor limits. The maximum batch size of a platform event trigger is 2,000, while the maximum of an Apex object trigger is 200. Therefore, platform event triggers are more likely to reach limits and can benefit from this feature.

To set a checkpoint for trigger resumption, set the replay ID of the last successfully processed event message using this method call.

```
EventBus.TriggerContext.currentContext().setResumeCheckpoint(replayId);
```

When the trigger stops its flow of execution, either intentionally or because of an unhandled exception, such as a limit exception, it fires again with a new batch (the sObject list in `Trigger.New`). The new batch starts with the event message after the one with the replay ID that you set. The events are resent in their original order based on the ReplayID field values, which are unchanged. The trigger processes the resent events and later batches sequentially. The `setResumeCheckpoint(replayId)` method doesn't cause the trigger execution to stop, but you can end the execution explicitly. For example, to control the batch size, end the execution flow after some event messages are processed.

If the supplied Replay ID isn't valid, the method throws an `EventBus.InvalidReplayIdException`. An invalid Replay ID is a replay ID that isn't in the current trigger batch of events in the `Trigger.new` list.

Note: Resuming a batch in one trigger doesn't affect another trigger on the same event object. However, having multiple triggers on the same object isn't a best practice because we can't guarantee the order of execution, so we recommend that you add only one trigger per object.

Example: This example trigger sets the replay ID of the last processed event message in each iteration. If a limit exception occurs, the trigger is fired again and resumes processing starting with the event message after the one with the set replay ID.

```
trigger ResumeEventProcessingTrigger on Low_Ink__e (after insert) {
    for (Low_Ink__e event : Trigger.New) {
        // Process the event message.
        // ...

        // Set the Replay ID of the last successfully processed event message.
        // If a limit is hit, the trigger refires and processing starts with the
        // event after the last one processed (the set Replay ID).
        EventBus.TriggerContext.currentContext().setResumeCheckpoint(event.replayId);

    }
}
```

Example: This example controls the platform event trigger batch size and matches it with the 200 batch size of Apex object triggers. The trigger counts the number of event messages processed. The `setResumeCheckpoint(replayId)` is called in each iteration of the loop after each event message that is successfully processed. The loop is exited if you exceed the count of 200 events, and the trigger stops execution. If you have unprocessed event messages, the trigger fires again. The list of event messages sent to the new trigger invocation starts with the event message after the one with the set replay ID.

Note: Starting in API version 51.0, you can configure the trigger batch size by using PlatformEventSubscriberConfig in Metadata API or Tooling API. For more information, see Configure the User and Batch Size for Your Platform Event Trigger with PlatformEventSubscriberConfig.

```
trigger ControlBatchSizeTrigger on Low_Ink__e (after insert) {
    Integer counter = 0;
    for (Low_Ink__e event : Trigger.New) {
      // Increase batch counter.
      counter++;
      // Only process the first 200 event messages
      if (counter > 200) {
        // Resume after the last successfully processed event message
        // after the trigger stops running.
        // Exit for loop.
        break;
      }

      // Process event message.
      // ....

      // Set Replay ID after which to resume event processing
      // in new trigger execution.
      EventBus.TriggerContext.currentContext().setResumeCheckpoint(
          event.ReplayId);
    }
}
```

The `TestBatchSizeTriggerResumption` test class contains a test for the `ControlBatchSizeTrigger`. The test method in the class publishes 201 event messages. Next, it calls the `deliver()` method twice to fire the trigger twice. The first invocation processes 200 event messages. The second invocation processes the last event message. The test verifies that the trigger was invoked by inspecting the `EventBusSubscriber.Position` property, which holds the replay ID of the last processed event message.

```
@isTest
public class TestBatchSizeTriggerResumption {

    @isTest static void testResumingBatchSizeTrigger() {

        Test.startTest();

        // Publish 201 test events
        List<Low_Ink__e> eventList = new List<Low_Ink__e>();
        for(Integer i=0;i<201;i++) {
            Low_Ink__e oneEvent = new Low_Ink__e(Serial_Number__c='X-' + i);
            eventList.add(oneEvent);
        }
        Database.SaveResult[] srs = EventBus.publish(eventList);
        for(Database.SaveResult sr : srs) {
            System.assertEquals(true, sr.isSuccess());
        }


        // Deliver the first 200 test event messages.
        // This will fire the associated event trigger.
```

```
        Test.getEventBus().deliver();

        // Get old position of this subscriber
        EventBusSubscriber subOld =
            [SELECT Name, Position, Topic
             FROM EventBusSubscriber
             WHERE Topic='Low_Ink__e' AND Name='ControlBatchSizeTrigger'];
        System.debug(subOld);

        // Refire the trigger for the last event (201st).
        Test.getEventBus().deliver();

        // VERIFICATION
        // Get new position of this subscriber
        EventBusSubscriber subNew =
            [SELECT Name, Position, Topic
             FROM EventBusSubscriber
             WHERE Topic='Low_Ink__e' AND Name='ControlBatchSizeTrigger'];
        System.debug(subNew);

        System.assertEquals(subOld.Position + 1, subNew.Position);

        Test.stopTest();

    }
}
```

Retry Event Triggers with `EventBus.RetryableException`

Get another chance to process event notifications. Retrying a trigger is helpful when a transient error occurs or when waiting for a condition to change. Retry a trigger if the error or condition is external to the event records and is likely to go away later.

[other]: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

An example of a transient condition: A trigger adds a related record to a master record if a field on the master record equals a certain value. It's possible that in a subsequent try, the field value changes and the trigger can perform the operation.

To retry the event trigger, throw `EventBus.RetryableException`. Events are resent after a small delay. The delay increases in subsequent retries. If the trigger receives a batch of events, retrying the trigger causes all events in the batch to be resent. The events are resent in their original order based on the ReplayID field values, which are unchanged. The trigger processes the resent events and later batches sequentially. Resent events have the same field values as the original events, but the batch sizes of the events can differ. For example, the initial trigger can receive events with replay ID 10 to 20. The resent batch can be larger, containing events with replay ID 10 to 40. When the trigger is retried, the DML operations performed in the trigger before the retry are rolled back and no changes are saved.

Limit the Number of Retry Attempts

You can run a trigger up to 10 times when it's retried (the initial run plus nine retries). After the trigger is retried nine times, it moves to the error state and stops processing new events. Events sent after the trigger moves to the error state and before it returns to the running state aren't resent to the trigger. To resume event processing, fix the trigger and save it.

We recommend limiting the retries to less than nine times. Use the `EventBus.TriggerContext.currentContext().retries` property to check how many times the trigger has been retried. Alternatively, you can query the `EventBusSubscriber.retries` field in API version 43.0 and later.

Example: This example is a skeletal trigger that gives you an idea of how to throw `EventBus.RetryableException` and limit the number of retries. The trigger uses an `if` statement to check whether a certain condition is true. Alternatively, you can use a try-catch block and throw `EventBus.RetryableException` in the catch block.

```
trigger ResendEventsTrigger on Low_Ink__e (after insert) {
    if (condition == true) {
        // Process platform events.
    } else {
        // Ensure we don't retry the trigger more than 4 times
        if (EventBus.TriggerContext.currentContext().retries < 4) {
            // Condition isn't met, so try again later.
            throw new EventBus.RetryableException(
                    'Condition is not met, so retrying the trigger again.');
        } else {
            // Trigger was retried enough times so give up and
            // resort to alternative action.
            // For example, send email to user.
        }
    }
}
```

Email Notifications for Triggers in Error State

When an Apex platform event trigger exceeds the maximum number of retries and is in the error state, you're notified by email. When the trigger subscriber reaches the error state, it disconnects and stops receiving published events.

For more information about the error state and how to resume the trigger, see the Subscription Statessection in View and Manage an Event's Subscribers on the Platform Event's Detail Page on page 80. We recommend limiting the retries to fewer than nine times to avoid reaching this state. See Retry Event Triggers with EventBus.RetryableException on page 48.

The email notification is not sent for general unhandled exceptions, such as uncatchable limit exceptions. Unlike Apex object triggers, platform event triggers don't generate exception emails for unhandled exceptions.

For a platform event trigger in the error state, the notification is sent to the developer specified in the trigger's Last Modified By field. To also send the email to other users, add them on the Apex Exception Email page in Setup. The recipients specified on the Apex Exception Email page also apply to emails sent for Apex object triggers and classes.

To set up more recipients, from Setup, in the Quick Find box, enter *Apex Exception Email*, and then select **Apex Exception Email**.

The users and email addresses entered apply to all managed packages in the customer's org. You can also configure Apex exception emails using the Tooling API object ApexEmailNotification.

Comparing `setResumeCheckpoint()` and `EventBus.RetryableException`

Determine which method is most suitable for resuming a platform event trigger.

`setResumeCheckpoint()` **Method**	`EventBus.RetryableException`
Use `setResumeCheckpoint()` when the trigger has processed event messages successfully before an unhandled exception occurs, such as a limit exception. After the exception, the trigger resumes after the last checkpointed event message.	Throw the `EventBus.RetryableException` to reprocess events when you expect an external condition to change or a transient error to go away.
Trigger execution continues after `setResumeCheckpoint()`.	Trigger execution halts after the `EventBus.RetryableException` is thrown.
DML operations performed are committed.	DML operations performed before the exception is thrown are rolled back and not committed.
When the trigger fires again, only the event messages after the one with the specified replay ID are resent, in addition to any new event messages.	When the trigger fires again, all event messages from the previous batch are resent in the new batch, in addition to any new event messages.
These `TriggerContext` properties don't apply and aren't populated: `retries` and `lastError`.	These `TriggerContext` properties are populated: `retries` and `lastError`.
No limit on how many times you can set a checkpoint with `setResumeCheckpoint()` and how many times the trigger can resume from the checkpoint after an unhandled exception occurs.	You can retry the trigger nine times after its initial run by throwing `EventBus.RetryableException`.

Subscribe to Platform Event Notifications in a Lightning Component

Subscribe to platform events with the `empApi` component in your Lightning web component or Aura component. The `empApi` component provides access to methods for subscribing to a streaming channel and listening to event messages.

The `empApi` component uses a shared CometD-based Streaming API connection, enabling you to run multiple streaming apps in the browser for one user. The connection isn't shared across user sessions.

The event delivery allocation applies for the `empApi` component. The allocation is per channel and per unique browser session. Also, the concurrent CometD client limit applies to the `empApi` component. Each logged-in user using `empApi` counts as one concurrent client. The `empApi` component isn't recommended for apps or sites that are used by a large number of users, such as Experience Cloud sites, because the limit can be reached. This limit is shared with other CometD clients. For more information, see Platform Event Allocations.

Note: As of Spring '19 (API version 45.0), you can build Lightning components using two programming models: the Lightning Web Components model, and the original Aura Components model. Lightning web components are custom HTML elements built using HTML and modern JavaScript. Lightning web components and Aura components can coexist and interoperate on a page.

Subscribe in a Lightning Web Component

To use the `empApi` methods in your Lightning web component, import the methods from the `lightning/empApi` module as follows.

```
import { subscribe, unsubscribe, onError, setDebugFlag, isEmpEnabled }
    from 'lightning/empApi';
```

Then call the imported methods in your JavaScript code.

For an example of how to use the `lightning/empApi` module and a complete reference, see the lightning-emp-api documentation in the *Lightning Component Library*.

Subscribe in an Aura Component

To use the `empApi` methods in your Aura component, add the `lightning:empApi` component inside your custom component and assign an `aura:id` attribute to it.

```
<lightning:empApi aura:id="empApi"/>
```

Then in the client-side controller, add functions to call the component methods.

For an example of how to use the `lightning:empApi` component and a complete reference, see the lightning:empApi documentation in the *Lightning Component Library*.

Subscribe to Platform Event Notifications with Pub/Sub API

Use Pub/Sub API to subscribe to event messages in an external client to integrate your systems. Simplify your development by using one API to publish, subscribe, and retrieve the event schema. Based on gRPC and HTTP/2, Pub/Sub API enables efficient delivery of binary event messages in the Apache Avro format. You can control the volume of event messages received per Subscribe call based on event processing speed in the client.

The Pub/Sub API service is defined in a proto file, with RPC method parameters and return types specified as protocol buffer messages. Pub/Sub API serializes the response of a Subscribe RPC call based on the protocol buffer message type specified in the proto file. For more information, see What is gRPC? and Protocol Buffers in the gRPC documentation, and pubsub_api.proto in the Pub/Sub API GitHub repository.

The `Subscribe` method uses bidirectional streaming, enabling the client to request more events as it consumes events. The client can control the flow of events received by setting the number of requested events in the `FetchRequest` parameter.

```
rpc Subscribe (stream FetchRequest) returns (stream FetchResponse);
```

Salesforce sends platform events to Pub/Sub API clients sequentially in the order they're received. The order of event notifications is based on the replay ID of events. A client can receive a batch of events at once. The total number of events across all batches received in FetchResponses per Subscribe call is equal to the number of events the client requests. The number of events in each individual batch can vary. If the client uses a buffer for the received events, ensure that the buffer size is large enough to hold all event messages in the batch. The buffer size needed depends on the publishing rate and the event message size. We recommend you set the buffer size to 3 MB.

To learn more about the RPC methods in Pub/Sub API, see Pub/Sub API RPC Method Reference in the *Pub/Sub API Developer Guide*.

The platform event channel name is case-sensitive. To subscribe to an event, use this format.

```
/event/Event_Name__e
```

To subscribe to a custom channel, use this format.

```
/event/Channel_Name__chn
```

Example: If you have a platform event named `Low Ink`, provide this channel name when subscribing.

```
/event/Low_Ink__e
```

This example shows the fields in the payload of the received event message. This example prints out the payload only. The received event message also contains the schema ID and the event ID, in addition to the payload.

```
{
  "CreatedDate": 1652978695951,
  "CreatedById": "005SM000000146PYAQ",
  "Printer_Model__c": "XZO-5",
  "Serial_Number__c": "12345",
```

```
  "Ink_Percentage__c": 0.2
}
```

Pub/Sub API is used for system integration and isn't intended for end-user scenarios. The binary event format enables efficient delivery of lightweight messages. As a result, after decoding the event payload, some fields aren't human readable and require additional processing. For example, `CreatedDate` is in Epoch time and can be converted to another date format for readability.

The event schema is versioned—when the schema changes, the schema ID changes as well. For more information about retrieving the event schema, see Get the Event Schema with Pub/Sub API.

Write a Pub/Sub API client to subscribe to platform event messages. You can use one of the 11 supported programming languages, including Python, Java, Go, and Node.

- To learn how to write a client in Java or Python, check out the Python quick start in Quick Starts in the *Pub/Sub API Developer Guide*.
- For code examples in other languages, see the Pub/Sub API GitHub repository.

Subscribe to Platform Event Notifications with CometD

Use CometD to subscribe to platform events in an external client.

Important: Pub/Sub API is newer than the CometD-based Streaming API. You can use Pub/Sub API to publish and subscribe to platform events and change data capture events. Based on gRPC API and HTTP/2, Pub/Sub API efficiently publishes and delivers binary event messages and supports multiple programming languages. For more information about Pub/Sub API, see Subscribe to Platform Event Notifications with Pub/Sub API.

Salesforce sends platform events to CometD clients sequentially in the order they're received. The order of event notifications is based on the replay ID of events. A CometD client can receive a batch of events at once. The number of event messages in a batch can vary. If the client uses a buffer for the received events, ensure that the buffer size is large enough to hold all event messages in the batch. The buffer size needed depends on the publishing rate and the event message size. At a minimum, set the buffer size to 10 MB, and adjust it higher if needed.

The process of subscribing to platform event notifications through CometD is similar to subscribing to PushTopics or generic events. The only difference is the channel name. The platform event channel name is case-sensitive and is in the following format.

```
/event/Event_Name__e
```

To subscribe to a custom channel, use this format.

```
/event/Channel_Name__chn
```

Use this CometD endpoint with the API version appended to it.

```
/cometd/61.0
```

Example: If you have a platform event named `Low Ink`, provide this channel name when subscribing.

```
/event/Low_Ink__e
```

The message of a delivered platform event looks similar to the following example for `Low Ink` events.

```
{
  "data": {
    "schema": "dffQ2QLzDNHqwB8_sHMxdA",
    "payload": {
      "CreatedDate": "2017-04-09T18:31:40.517Z",
      "CreatedById": "005D0000001cSZs",
```

```
      "Printer_Model__c": "XZO-5",
      "Serial_Number__c": "12345",
      "Ink_Percentage__c": 0.2
    },
    "event": {
      "EventUuid": "2ec0e371-1395-457f-9275-be1b527a72f7",
      "replayId": 2
    }
  },
  "channel": "/event/Low_Ink__e"
}
```

The `schema` field in the event message contains the ID of the platform event schema. The schema is versioned—when the schema changes, the schema ID changes as well. For more information about retrieving the event schema, see Get the Event Schema.

Add custom logic to your client to perform some operations after a platform event notification is received. For example, the client can create a request to order a new cartridge for this printer model.

SEE ALSO:

Streaming API Developer Guide: Message Durability

CometD

Pub/Sub API Developer Guide

Considerations for Publishing and Subscribing to Platform Events with Apex and APIs

Group Platform Events into One Stream with a Custom Channel

With a custom channel, you can receive a stream of event messages corresponding to one or more custom platform events, or Real-Time Event Monitoring events. For example, if you've defined platform events corresponding to orders for different regions, one client can subscribe to all those events and process them. Custom channels are supported in Pub/Sub API clients, CometD clients, and event relays only. You can also add filters to custom channels. By using only one client to subscribe to all events and using filters, your subscriptions are optimized.

Types of Events Supported

Custom channels are available for high-volume custom platform events that you define and for Real-Time Event Monitoring events. They aren't supported for standard platform events or legacy standard-volume custom platform events.

PlatformEventChannel and PlatformEventChannelMember Objects in the API

Create a custom channel, and specify the platform events it contains in Tooling API and Metadata API.

PlatformEventChannel represents a custom channel. The `ChannelType` field indicates which members the custom channel can contain. A `ChannelType` value of `event` means that the channel can contain platform events via its channel members. To specify the specific type of events a channel can hold, such as Real-Time Event Monitoring events, use the optional `EventType` field in combination with `ChannelType`.

A custom channel can contain events for only one event product. You can't mix events from different event products in one channel. For example, you can't add platform events and change data capture events to the same channel.

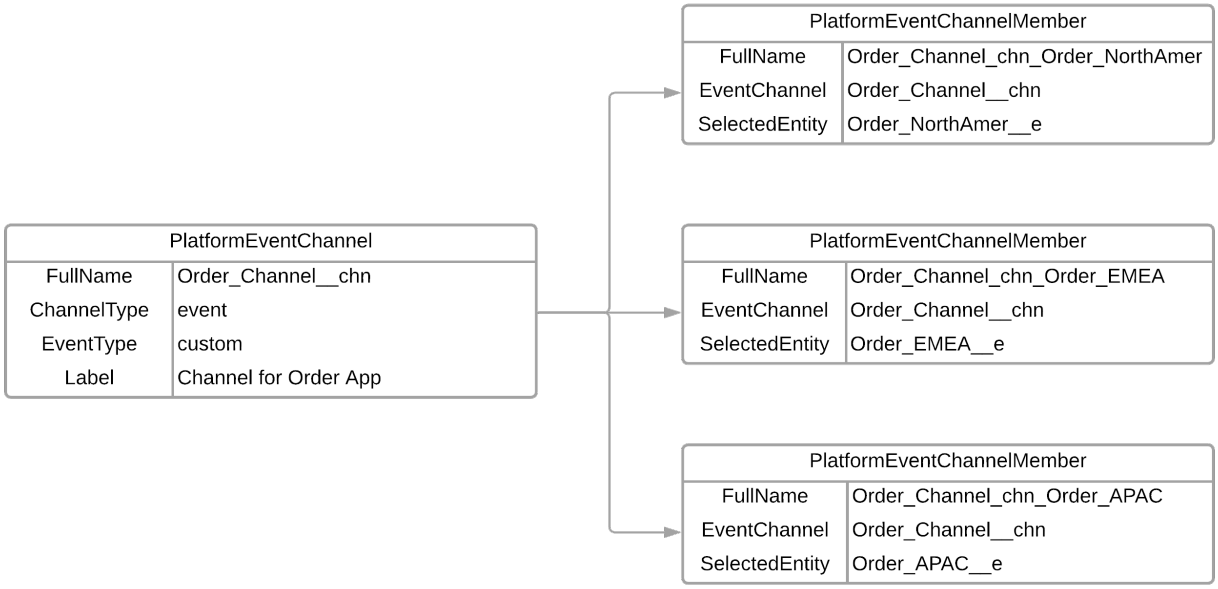
PlatformEventChannelMember represents a member of a channel. It contains a platform event in the `SelectedEntity` field and is associated with the channel via the `EventChannel` field.

Note:

- After you create a channel, you can't change its `ChannelType` or `EventType` field values.
- When you delete a channel by deleting PlatformEventChannel, all its associated members (PlatformEventChannelMember entities) are also deleted.
- You can't add the ApiEventStream and ReportEventStream Real-Time Event Monitoring events to a custom channel via Tooling API because they aren't available in Tooling API. You can add them via Metadata API.

Example Diagram

This diagram shows the object relationships and definitions of the custom channel `Order_Channel__chn` and its members. The channel is set up to receive order events for North America, EMEA, and the APAC regions. A custom event is defined for each region: Order_NorthAmer__e, Order_EMEA__e, Order_APAC__e. Each of these platform events is added to the channel via PlatformEventChannelMember objects. An order management app can subscribe to the custom channel, `Order_Channel__chn`, and receive messages of the three platform events.



Subscribing to a Custom Channel and Getting the Event API Name with Pub/Sub API

When you subscribe to a custom channel, provide the channel name in the format `/event/ChannelName__chn`, such as `/event/Order_Channel__chn`. Your subscriber receives event messages of all events that are part of the channel. If you subscribe to the custom channel with Pub/Sub API, get the API name of the event received through the event schema. To retrieve the event schema, call the `GetSchema` RPC method using the schema ID contained in the received event. The schema name is in the `schema_json` field in the response and identifies the event API name. See GetSchema RPC Method in the Pub/Sub API Developer Guide.

This example shows a received event with Pub/Sub API with the event name taken from the event schema.

```
{
  "CreatedDate": 1711497484289,
  "CreatedById": "0055f000005mc66AAA",
  "Order_Number__c": "2",
  "City__c": "London",
  "Amount__c": 20.0
}
with schema name: Order_EMEA__e
```

Subscribing to a Custom Channel and Getting the Event API Name with CometD via the `EventApiName` Field

When you subscribe to a custom channel, provide the channel name in the format `/event/ChannelName__chn`, such as `/event/Order_Channel__chn`. Your subscriber receives event messages of all events that are part of the channel. In a CometD client, each event message contains the `EventApiName` field, which contains the type of the event. For example, this event message has an `EventApiName` of Order_EMEA__e, which means that it's an Order_EMEA__e event.

```
{
  "schema": "e8jMOnID4xDThlaPBMx5gg",
  "payload": {
    "City__c": "London",
    "CreatedById": "005RM000002Qu16YAC",
    "Amount__c": 20,
    "CreatedDate": "2022-03-29T13:45:19.230Z",
    "Order_Number__c": "2"
  },
  "event": {
    "EventApiName": "Order_EMEA__e",
    "EventUuid": "218544ad-0472-4315-970f-8825a2802de6",
    "replayId": 10306
  }
}
```

The `EventApiName` field is available in event messages received in CometD clients that use a Streaming API endpoint with API version 55.0 and later. It isn't available in event messages received in other subscribers, such as Apex triggers, flows, and Pub/Sub API. It isn't included in change data capture events and events that don't support custom channels. Also, the `EventApiName` field isn't part of the event schema that the REST eventSchema resource or the describe call returns.

Custom Channel Allocations

The maximum number of custom channels and channel members that you can add differ based on the type of events the channel holds. There are separate allocations for custom channels and channel members. See Common Platform Event Allocations.

IN THIS SECTION:

Create a Custom Channel for Custom Platform Events Using the API

Let's walk through the steps to create a channel and add two platform events via channel members. Then, you can subscribe to the channel to validate receiving event messages for platform events.

Create a Custom Channel for Real-Time Event Monitoring Events Using the API

Let's walk through the steps to create a channel and add two Real-Time Event Monitoring events via channel members.

List Custom Channels and Channel Members

You can find which channels and channel members are set up in your Salesforce org by performing SOQL queries through Tooling API.

Create a Custom Channel for Custom Platform Events Using the API

Let's walk through the steps to create a channel and add two platform events via channel members. Then, you can subscribe to the channel to validate receiving event messages for platform events.

IN THIS SECTION:

Prerequisite: Define Platform Events

The custom channel examples depend on a predefined custom platform events called Order_NorthAmer__e and Order_EMEA__e. Before creating the custom channel, define these events in Salesforce.

Create a Custom Channel and Add Platform Events with Tooling API

Create a channel for orders named `Order_Channel__chn`, and add two platform events as members: Order_NorthAmer__e and Order_EMEA__e.

Metadata API Example: Create a Custom Channel and Add Platform Events

Instead of Tooling API, you can use Metadata API to create a channel and channel member. We recommend using Metadata API as part of the application lifecycle management process to develop, test, deploy, and release your apps to production. If you want to only configure the channel, we recommend using Tooling API with REST.

Subscribe to the Channel

After creating a custom channel and its members, subscribe to the channel using a Pub/Sub API Java client, and receive event messages.

Prerequisite: Define Platform Events

The custom channel examples depend on a predefined custom platform events called Order_NorthAmer__e and Order_EMEA__e. Before creating the custom channel, define these events in Salesforce.

1. From Setup, in the Quick Find box, enter *`Platform Events`*, and then select **Platform Events**.
2. Click **New Platform Event**.
3. Provide these values.
   a. Label: *`Order NorthAmer`*
   b. Plural Label: *`Order NorthAmer`*
   c. Select **Starts with a vowel sound**, if available.
   d. Click **Save**.
4. Create four fields. In Custom Fields & Relationships, click **New** for each field, and follow the wizard.
   a. Field type: *`Text`*; Field Label: *`Order Number`*; Length: *`10`*
   b. Field type: *`Text`*; Field Label: *`City`*; Length: *`50`*

EDITIONS

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To define a platform event:
- Customize Application

c. Field type: *Number*; Field Label: *Amount*; Length: *16*; Decimal Places: *2*

5. Repeat these steps for a platform event with the label *Order EMEA* and the same fields.

Create a Custom Channel and Add Platform Events with Tooling API

Create a channel for orders named `Order_Channel__chn`, and add two platform events as members: Order_NorthAmer__e and Order_EMEA__e.

You can use your preferred REST API tool to perform these steps. We recommend using Postman with the Salesforce Platform APIs collection, which contains handy templates for Salesforce API calls. See Quick Start: Connect Postman to Salesforce in Trailhead.

USER PERMISSIONS

To create or update PlatformEventChannel and PlatformEventChannelMember objects:
- Customize Application

To use REST API:
- API Enabled

1. Create the channel using PlatformEventChannel, and set the `channelType` field to `event`. Send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannel
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and then click **Create channel**.

2. Use this request body.

```
{
  "FullName": "Order_Channel__chn",
  "Metadata": {
    "channelType": "event",
    "label": "Custom Channel for Orders"
  }
}
```

You receive a response similar to this response.

```
{
    "id": "0YLRM0000004CEI4A2",
    "success": true,
    "errors": [],
    "warnings": [],
    "infos": []
}
```

3. Add the Order_NorthAmer__e platform event to the channel using PlatformEventChannelMember. The channel member references the channel it's part of (Order_Channel__chn) through the `eventChannel` field. Specify the platform event in the `selectedEntity` field. Send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannelMember
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and then click **Create channel member**.

4. Use this request body.

```
{
 "FullName": "Order_Channel_chn_Order_NorthAmer_e",
 "Metadata": {
  "eventChannel": "Order_Channel__chn",
```

```
  "selectedEntity": "Order_NorthAmer__e"
 }
}
```

You receive a response similar to this response.

```
{
    "id": "0v8RM0000000N6uYAE",
    "success": true,
    "errors": [],
    "warnings": [],
    "infos": []
}
```

5. Add the second channel member that specifies the platform event, Order_EMEA__e. Send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannelMember
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and then click **Create channel member**.

6. Use this request body.

```
{
 "FullName": "Order_Channel_chn_Order_EMEA_e",
 "Metadata": {
  "eventChannel": "Order_Channel__chn",
  "selectedEntity": "Order_EMEA__e"
 }
}
```

You receive a response similar to this response.

```
{
    "id": "0v8RM0000004VPJYA2",
    "success": true,
    "errors": [],
    "warnings": [],
    "infos": []
}
```

SEE ALSO:

Tooling API Developer Guide: PlatformEventChannel

Tooling API Developer Guide: PlatformEventChannelMember

Metadata API Example: Create a Custom Channel and Add Platform Events

Instead of Tooling API, you can use Metadata API to create a channel and channel member. We recommend using Metadata API as part of the application lifecycle management process to develop, test, deploy, and release your apps to production. If you want to only configure the channel, we recommend using Tooling API with REST.

In this example, you create a channel for orders named Order_Channel__chn, and you add two platform events as members: Order_NorthAmer__e and Order_EMEA__e.

To create a channel and channel member with Metadata API, you can use tools such as Visual Studio Code with the Salesforce Extension pack or Salesforce CLI. For more information, see Metadata API Developer Tools and Quick Start: Metadata API in the *Metadata API Developer Guide*.

This sample custom channel definition is for the `Order_Channel__chn` channel. The file name is `Order_Channel__chn.platformEventChannel`. To have this channel accept platform events, `event` is specified for `channelType`.

USER PERMISSIONS

To deploy and retrieve metadata types:
- Customize Application

To update metadata types:
- Modify Metadata Through Metadata API Functions

To use Metadata API:
- API Enabled

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <channelType>event</channelType>
    <label>Custom Channel for Orders</label>
</PlatformEventChannel>
```

The sample channel member definition associates the custom platform event to the channel. This channel member specifies the platform event, Order_NorthAmer__e, and the channel, Order_Channel__chn. The file name is `Order_Channel_chn_Order_NorthAmer_e.platformEventChannelMember`.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>Order_Channel__chn</eventChannel>
    <selectedEntity>Order_NorthAmer__e</selectedEntity>
</PlatformEventChannelMember>
```

This channel member specifies the custom platform event, Order_EMEA__e, and the channel, Order_Channel__chn. The file name is `Order_Channel_chn_Order_EMEA_e.platformEventChannelMember`.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>Order_Channel__chn</eventChannel>
    <selectedEntity>Order_EMEA__e</selectedEntity>
</PlatformEventChannelMember>
```

This `package.xml` file references the channel and its two channel members.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Order_Channel__chn</members>
        <name>PlatformEventChannel</name>
    </types>
    <types>
        <members>Order_Channel_chn_Order_NorthAmer_e</members>
        <name>PlatformEventChannelMember</name>
    </types>
    <types>
        <members>Order_Channel_chn_Order_EMEA_e</members>
```

```
        <name>PlatformEventChannelMember</name>
    </types>
    <version>61.0</version>
</Package>
```

SEE ALSO:

Metadata API Developer Guide: PlatformEventChannel

Metadata API Developer Guide: PlatformEventChannelMember

Subscribe to the Channel

After creating a custom channel and its members, subscribe to the channel using a Pub/Sub API Java client, and receive event messages.

In this example, you subscribe to the `Order_Channel__chn` channel you created for two custom platform events. Only Pub/Sub API and CometD clients support custom channels. Other subscribers, such as Apex triggers, flows, and processes, don't support custom channels.

1. To set up the Pub/Sub API Java client, follow the steps in Java Quick Start for Publishing and Subscribing to Events in the Pub/Sub API Developer Guide.
2. In Step 3: Configure Client Parameters, supply the configuration parameters in `arguments.yaml`. Also, make sure you supply this value:

   ```
   TOPIC: /event/Order_Channel__chn
   ```

3. In a Terminal window, navigate to the top-level `java` folder.
4. To run the Subscribe RPC example, enter: `./run.sh genericpubsub.Subscribe`.
5. After subscribing to the filtered channel, publish an Order_NorthAmer__e event and an Order_EMEA__e event.
   a. Open another Terminal window.
   b. Navigate to your local `pub-sub-api` folder.
   c. In an IDE, such as Visual Studio Code, open `java/src/main/java/utility/CommonContext.java`.
   d. Modify the `createEventMessage` method that takes one parameter. Replace the method with this snippet. Replace the `CreatedById` value with your Salesforce user ID. See Find the Salesforce ID for a User or Profile.

      ```
      public GenericRecord createEventMessage(Schema schema) {
          // Update CreatedById with the appropriate User Id from your org.
          return new GenericRecordBuilder(schema)
                  .set("CreatedDate", System.currentTimeMillis())
                  .set("CreatedById", "<User_Id>")
                  .set("Order_Number__c", "1")
                  .set("City__c", "Los Angeles")
                  .set("Amount__c", 35.0).build();
      }
      ```

   e. For the configuration parameters in `java/src/main/resources/arguments.yaml`, supply this value.

      ```
      TOPIC: /event/Order_NorthAmer__e
      ```

   f. Build the client from the top-level `java` folder with this command: `mvn clean install`.
   g. To run the Publish RPC example from the `java` folder, enter `./run.sh genericpubsub.Publish`.

h. Repeat the previous steps to publish an Order_EMEA__e event. Both events have the same fields, so modify the field values in the `createEventMessage` method in `CommonContext.java`.

```
            .set("Order_Number__c", "2")
            .set("City__c", "London")
            .set("Amount__c", 20.0).build();
```

i. For the configuration parameters in `arguments.yaml`, supply this value.

```
TOPIC: Order_EMEA__e
```

j. Build the client from the top-level `java` folder with this command: `mvn clean install`.

k. To run the Publish RPC example from the `java` folder, enter `./run.sh genericpubsub.Publish`.

Because `Order_Channel__chn` includes both the Order_NorthAmer__e and Order_EMEA__e event types, you receive the event messages of both events in the terminal where your subscriber client is running. This example shows the two received event messages after subscribing to the filtered channel, `/event/Order_Channel__chn`. The Pub/Sub API client output includes the event name, which is obtained from the schema on the received event.

```
2024-03-08 16:58:09,730 [pool-3-thread-2]
java.lang.Class - Received event with payload:
{
  "CreatedDate": 1711497250504,
  "CreatedById": "0055f000005mc66AAA",
  "Order_Number__c": "1",
  "City__c": "Los Angeles",
  "Amount__c": 35.0
}
with schema name: Order_NorthAmer__e

2024-03-08 16:58:09,730 [pool-3-thread-1]
java.lang.Class - Received event with payload:
{
  "CreatedDate": 1711497484289,
  "CreatedById": "0055f000005mc66AAA",
  "Order_Number__c": "2",
  "City__c": "London",
  "Amount__c": 20.0
}
with schema name: Order_EMEA__e
```

If you use a Streaming API (CometD) to subscribe to the channel, the event messages contain the `EventApiName` as seen in this example.

```
{
  "schema": "LofZQqy_2SpDbzzZptVpxQ",
  "payload": {
    "City__c": "Los Angeles",
    "CreatedById": "0055f000005mc66AAA",
    "Amount__c": 35.0,
    "CreatedDate": "2024-03-09T00:27:54.115Z",
    "Order_Number__c": "1"
  },
  "event": {
    "EventUuid": "7fffe7b9-106f-4708-bd8d-00a33aa6ab84",
```

```
    "replayId": 12319972,
    "EventApiName": "Order_NorthAmer__e"
  }
}

{
  "schema": "LofZQqy_2SpDbzzZptVpxQ",
  "payload": {
    "City__c": "London",
    "CreatedById": "0055f000005mc66AAA",
    "Amount__c": 20.0,
    "CreatedDate": "2024-03-09T00:27:54.087Z",
    "Order_Number__c": "2"
  },
  "event": {
    "EventUuid": "4a502e4c-8cc4-4a6e-818f-45f91efac9a4",
    "replayId": 12319973,
    "EventApiName": "Order_EMEA__e"
  }
}
```

Create a Custom Channel for Real-Time Event Monitoring Events Using the API

Let's walk through the steps to create a channel and add two Real-Time Event Monitoring events via channel members.

IN THIS SECTION:

Create a Custom Channel and Add Real-Time Event Monitoring Events with Tooling API

Create a channel for Real-Time Event Monitoring events named `Event_Monitoring_Channel__chn` by using the `eventType` field in PlatformEventChannel. Also, add two Real-Time Event Monitoring events as members: ApiAnomalyEvent and FileEvent.

You can use your preferred REST API tool to perform these steps. We recommend using Postman with the Salesforce Platform APIs collection, which contains handy templates for Salesforce API calls. To set up Postman, see Quick Start: Connect Postman to Salesforce in Trailhead.

USER PERMISSIONS

To create or update PlatformEventChannel and PlatformEventChannelMember objects:
- Customize Application

To use REST API:
- API Enabled

To use Real-Time Event Monitoring events:
- Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

1. Create the channel using PlatformEventChannel, set the `channelType` field to `event` and the `eventType` field to `monitoring`. Send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannel
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and then click **Create channel**.

2. Use this request body.

```
{
  "FullName": "Event_Monitoring_Channel__chn",
  "Metadata": {
    "channelType": "event",
    "eventType": "monitoring",
    "label": "Custom Channel for Real-Time Event Monitoring events"
  }
}
```

You receive a response similar to this response.

```
{
  "id": "0YLZM000000CaR74AK",
  "success": true,
  "errors": [],
  "warnings": [],
  "infos": []
}
```

3. Add the Order_NorthAmer__e platform event to the channel using PlatformEventChannelMember. The channel member references the channel it's part of (Order_Channel__chn) through the `eventChannel` field. Specify the platform event in the `selectedEntity` field. Send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannelMember
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and then click **Create channel member**.

4. Use this request body.

```
{
 "FullName": "Event_Monitoring_Channel_chn_ApiAnomalyEvent",
 "Metadata": {
  "eventChannel": "Event_Monitoring_Channel__chn",
  "selectedEntity": "ApiAnomalyEvent"
 }
}
```

You receive a response similar to this response.

```
{
  "id": "0v8ZM000000CaeZYAS",
  "success": true,
  "errors": [],
  "warnings": [],
  "infos": []
}
```

5. Add the second channel member that specifies the platform event, Order_EMEA__e. Send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannelMember
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and click **Create channel member**.

6. Use this request body.

```
{
 "FullName": "Event_Monitoring_Channel_chn_FileEvent",
 "Metadata": {
  "eventChannel": "Event_Monitoring_Channel__chn",
  "selectedEntity": "FileEvent"
 }
}
```

You receive a response similar to this response.

```
{
  "id": "0v8ZM0000008OPVYA2",
  "success": true,
  "errors": [],
  "warnings": [],
  "infos": []
}
```

Before you subscribe to the `Event_Monitoring_Channel__chn` channel, enable Real-Time Event Monitoring and streaming for the specific events. See Enable Access to Real-Time Event Monitoring and Manage Real-Time Event Monitoring Events in Salesforce Help.

To subscribe to the channel, see Subscribe to the Channel and use `/event/Event_Monitoring_Channel__chn` for the `TOPIC` parameter. After you start the subscription, perform some actions in the Salesforce org to cause some events to be generated. For example, to receive FileEvent events on the channel, download a document.

SEE ALSO:

Platform Events Developer Guide: ApiAnomalyEvent

Platform Events Developer Guide: FileEvent

Tooling API Developer Guide: PlatformEventChannel

Tooling API Developer Guide: PlatformEventChannelMember

Metadata API Example: Create a Custom Channel and Add Real-Time Event Monitoring Events

You can use Metadata API to create a channel and channel member. We recommend using Metadata API as part of the application lifecycle management process to develop, test, deploy, and release your apps to production. If you want to only configure the channel, we recommend using Tooling API with REST.

In this example, you create a channel for Real-Time Event Monitoring events named `Event_Monitoring_Channel__chn` by using the `eventType` field in PlatformEventChannel. Also, you add two Real-Time Event Monitoring events as members: ApiAnomalyEvent and FileEvent.

To create a channel and channel member with Metadata API, you can use tools such as Visual Studio Code with the Salesforce Extension pack or Salesforce CLI. See Metadata API Developer Tools and Quick Start: Metadata API in the *Metadata API Developer Guide*.

This sample custom channel definition is for the `Event_Monitoring_Channel__chn` channel. The file name is `Event_Monitoring_Channel__chn.platformEventChannel`. To have this channel accept Real-Time Event Monitoring events, `event` is specified for `channelType` and `monitoring` is specified for `eventType`.

USER PERMISSIONS

To deploy and retrieve metadata types:
- Customize Application

To update metadata types:
- Modify Metadata Through Metadata API Functions

To use Metadata API:
- API Enabled

To use Real-Time Event Monitoring events:
- Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <channelType>event</channelType>
    <eventType>monitoring</eventType>
    <label>Custom Channel for Real-Time Event Monitoring events</label>
</PlatformEventChannel>
```

This channel member definition associates ApiAnomalyEvent to the channel. The file name is `Event_Monitoring_Channel_chn_ApiAnomalyEvent.platformEventChannelMember`.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>Event_Monitoring_Channel__chn</eventChannel>
    <selectedEntity>ApiAnomalyEvent</selectedEntity>
</PlatformEventChannelMember>
```

This sample channel member definition associates FileEvent to the channel. The file name is `Event_Monitoring_Channel_chn_FileEvent.platformEventChannelMember`.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>Event_Monitoring_Channel__chn</eventChannel>
    <selectedEntity>FileEvent</selectedEntity>
</PlatformEventChannelMember>
```

This `package.xml` file references the channel and its two channel members.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Event_Monitoring_Channel__chn</members>
        <name>PlatformEventChannel</name>
    </types>
    <types>
        <members>Event_Monitoring_Channel_chn_ApiAnomalyEvent</members>
        <name>PlatformEventChannelMember</name>
    </types>
    <types>
        <members>Event_Monitoring_Channel_chn_FileEvent</members>
        <name>PlatformEventChannelMember</name>
    </types>
    <version>61.0</version>
</Package>
```

Before you subscribe to the `Event_Monitoring_Channel__chn` channel, enable Real-Time Event Monitoring and streaming for the specific events. See Enable Access to Real-Time Event Monitoring and Manage Real-Time Event Monitoring Events in Salesforce Help.

To subscribe to the channel, see Subscribe to the Channel and use `/event/Event_Monitoring_Channel__chn` for the `TOPIC` parameter. After you start the subscription, perform some actions in the Salesforce org to cause some events to be generated. For example, to receive FileEvent events on the channel, download a document.

SEE ALSO:

Platform Events Developer Guide: ApiAnomalyEvent

Platform Events Developer Guide: FileEvent

Metadata API Developer Guide: PlatformEventChannel

Metadata API Developer Guide: PlatformEventChannelMember

List Custom Channels and Channel Members

You can find which channels and channel members are set up in your Salesforce org by performing SOQL queries through Tooling API.

To perform SOQL queries, make a REST `query` call or use the Query Editor in the Developer Console with the **Tooling API** option selected. To make REST calls using Postman, set up Postman with the Salesforce Platform APIs collection. See Quick Start: Connect Postman to Salesforce in Trailhead.

Perform a GET request to this endpoint with the SOQL query appended.

USER PERMISSIONS

To query PlatformEventChannel and PlatformEventChannelMember Tooling objects:
- View Setup and Configuration

To use REST with Tooling API:
- API Enabled

```
/services/data/v61.0/tooling/query?q=<query>
```

This query returns all the custom channels. The query results in this page are based on the `Order_Channel__chn` channel.

```
SELECT Id, DeveloperName, ChannelType, MasterLabel FROM PlatformEventChannel
```

If you're using Postman, expand **Event Platform** > **Custom Channels**, and then click **List event channels**.

Sample result:

Id
0YLRM0000004CEI4A2

DeveloperName
Order_Channel

ChannelType
event

MasterLabel
Custom Channel for Orders

And this query returns all the channel members.

```
SELECT Id, DeveloperName,EventChannel, SelectedEntity FROM PlatformEventChannelMember
```

If you're using Postman, expand **Event Platform** > **Custom Channels**, and then click **List channel members**.

For example, the query returns the two channel members created earlier. The `SelectedEntity` field references the ID of the custom platform event.

First channel member:

Id
0v8RM0000000N6uYAE

DeveloperName
Order_Channel_chn_Order_NorthAmer_e

EventChannel
0YLRM0000004CEI4A2

SelectedEntity
01IRM0000006w522AA

Second channel member:

Id
0v8RM0000004VPJYA2

DeveloperName
Order_Channel_chn_Order_EMEA_e

EventChannel
0YLRM0000004CEI4A2

SelectedEntity
01IRM0000006w572AA

Filter Your Stream of Platform Events with Custom Channels

Receive only the event messages that match a predefined filter on a custom channel. Create a channel, and configure it with a filter expression. Pub/Sub API and Streaming API (CometD) subscribers to the channel receive a filtered stream of events. With fewer events delivered to subscribers, event processing is optimized. Also, subscribers make more efficient use of the event delivery allocation.

Note:

- This feature is supported for high-volume custom platform events that you define. It isn't supported for legacy standard-volume custom platform events or standard platform events such as real-time event monitoring events.
- This feature is supported in Pub/Sub API and Streaming API (CometD) subscribers but not in other types of subscribers, such as Apex triggers, flows, and processes.
- If you use Government Cloud and your org was created before January 14, 2022, contact Salesforce to enable this feature. Government Cloud orgs created on or after January 14, 2022 have this feature enabled. This feature is available in all other clouds.

IN THIS SECTION:

Platform Event Filters

Using Tooling API or Metadata API, an administrator with the Customize Application permission can configure a complex filter expression that contains multiple fields.

Filter Expressions in Channel Members

Add a filter expression in a channel member that's associated with a custom channel. The channel member associates a custom platform event with the channel and specifies the filter expression. The channel holds the filtered stream of event messages that match the filter expression for the specified custom platform event.

Subscribe to the Channel and Receive the Filtered Event Stream

After configuring the filter, subscribe to the channel and receive the event messages that match the filter expression. Only Pub/Sub API and CometD clients support stream filtering. Because Apex triggers, flows, and processes don't support custom channels, you can't use them to subscribe to filtered event streams.

Get Custom Channels and Channel Members

You can find which channels and channel members are set up in your Salesforce org by performing SOQL queries through Tooling API.

SEE ALSO:

Subscribe to Platform Event Notifications with CometD

Subscribe to Platform Event Notifications with Pub/Sub API

Platform Event Filters

Using Tooling API or Metadata API, an administrator with the Customize Application permission can configure a complex filter expression that contains multiple fields.

The filter expression is associated with a custom channel and is included in a channel member. You can add one or more filter expressions via channel members to a custom channel. For more information about channels and their allocations, see Group Platform Events into One Stream with a Custom Channel.

IN THIS SECTION:

Filter Expression Format

The filter expression format is based on SOQL and supports a subset of SOQL operators and field types. The filter expression can contain one or more field expressions, joined by a logical operator.

Single-field expression:

```
<FieldName> <Comparison Operator> <Value>
```

Example of multiple-field expressions joined by logical operators:

```
<FieldName> <Comparison Operator> <Value> AND (<FieldName> <Comparison Operator> <Value>
  OR <FieldName> <Comparison Operator> <Value>) ...
```

Text field values are included within single quotes. Examples of a single-field expression filtering on a Text field:

```
City__c = 'San Francisco'
City__c LIKE 'San F%'
```

Example of a single-field expression filtering on a Date field:

```
Delivery_Date__c > 2022-07-14T09:30:11-08:00
```

Examples of a multiple-field expression:

```
City__c = 'San Francisco' AND Amount__c > 22.34 AND Has_Shipped__c = true
City__c = 'San Francisco' OR City__c = 'New York'
```

Example of a multiple-field expression using parentheses and the AND and OR logical operators:

```
Amount__c > 22.34 AND (City__c = 'San Francisco' OR City__c = 'New York')
```

Spaces within each field expression are optional. For the entire filter expression, if you use parentheses around each field expression, spaces are optional between the field expression pairs and the logical operator. Otherwise, include a space between the logical operator and the field expressions.

Supported Field Types

All field types supported for custom platform event fields are supported in filter expressions.

- Checkbox
- Date
- Date/Time
- Number
- Text
- Text Area (Long)

Supported Comparison Operators

These comparison operators are supported in filter expressions.

- =
- !=
- >
- <
- >=
- <=
- LIKE

Considerations for the LIKE Operator

The `LIKE` operator is supported for Text fields. The text string value must be enclosed in single quotes. The `LIKE` operator can match partial text string values when used with the `%` and `_` wildcards. The `%` wildcard matches zero or more characters. The `_` wildcard matches exactly one character.

For example, this expression matches messages with City__c values that start with 'San F', such as 'San Francisco' and 'San Fernando'. But it doesn't match the city value of 'San Mateo'.

```
City__c LIKE 'San F%'
```

This expression matches messages with City__c values that start with 'Bake' and end with any character, such as 'Baker'.

```
City__c LIKE 'Bake_'
```

Supported Logical Operators

These logical operators are supported in filter expressions.

- AND
- OR
- NOT

Considerations for the NOT Operator

Use the NOT operator to negate an expression. For example, this expression states that the city isn't New York.

```
NOT City__c = 'New York'
```

In this next expression, the NOT operator negates two conditions evaluated with the AND operator. The filter matches events that have the city set to a value other than New York or the Amount set to a value other than 100. If an event has both the city set to New York and the Amount set to 100, it doesn't match the filter criteria and isn't delivered.

```
NOT(City__c='New York' AND Amount__c=100)
```

If there's more than one expression, including the expression with the NOT operator, parentheses around NOT and its expression are required. In this example, two field expressions are joined by the AND operator, and NOT is used only for the first expression. It must be enclosed within parentheses because there are two expressions. The entire filter expression states that the city isn't New York and the Amount value is greater than 10.50.

```
(NOT(City__c='New York')) AND (Amount__c>10.50)
```

This example also requires enclosing the NOT operator in parentheses. This filter expression matches events that have a delivery date greater than 2021-10-21T09:30:11 in the Pacific time zone and whose city isn't New York or amount isn't 100.

```
Delivery_Date__c>2021-10-21T09:30:11-08:00 AND (NOT(City__c='New York' AND Amount__c=100))
```

Filter Expression Allocations

- You can add up to 10 fields in a filter expression.
- The filter expression's maximum length is 131,072 characters.
- A filter expression is part of a channel member. The maximum number of filter expressions you can add per channel depends on the number of channel members you can create. For more information, see Custom Channel Allocations.

SEE ALSO:

Salesforce Object Query Language (SOQL) Reference

Field Considerations

Keep these considerations in mind for the fields in a filter expression.

Text Field Considerations

- Enclose Text field values in single quotes. For example, `MyTextField__c='Hello'` is valid, but `MyTextField__c=Hello` isn't valid.
- Text values are case-insensitive. For example, `MyTextField__c='ABC'` and `MyTextField__c='abc'` are considered the same. Events with any combination of uppercase and lowercase letters of the field value match the filter and are delivered.
- A Text value can contain spaces and tabs between words. Because leading and trailing spaces and tabs in Text field values are stripped in the received event messages, don't include them in the filter string. If you do, the filter comparison fails.
- Text fields support all comparison operators. Comparisons of Text fields using `<`, `<=`, `>`, and `>=` are lexicographic, similar to SOQL.
- If a Text field value includes special characters such as a double quote (`"`), you can escape the characters, with some exceptions. You can't escape the backslash (`\`), underscore (`_`), and percent (`%`) characters. For more information, see Quoted String Escape Sequences in the *SOQL and SOSL Reference*.

Checkbox Field Considerations

- Checkbox fields support only the `=` and `!=` comparison operators. Using another operator causes an error.
- Comparing a Checkbox field to null is equivalent to comparing it to a value of false.

Date Field Considerations

- For Date/Time fields, the supported formats include the time zone offset preceded by `+` or `-`: `YYYY-MM-DDThh:mm:ss+hh:mm` and `YYYY-MM-DDThh:mm:ss-hh:mm`, and the UTC time zone designator `Z`: `YYYY-MM-DDThh:mm:ssZ`.
- You can compare Date and Date/Time fields to hardcoded date values only, such as `2021-07-09` or `2021-07-09T10:30:11-08:00`. You can't compare them to date literals such as `TOMORROW`. For more information, see Date Formats and Date Literals in the *SOQL and SOSL Reference*.

Number Field Considerations

- If a filter expression contains a Number field with a value greater than 2147483647, when you attempt to save the channel member containing the filter expression you get a `FIELD_INTEGRITY_EXCEPTION` with an error message that starts with `"A number format error occurred"`. The error is due to a limitation in SOQL, which is described in this known issue. To save the filter expression, append `.0` to the value so that it becomes a decimal value. For example: `"filterExpression" : "MyNumberField__c = 1657093404000.0"`

Null Field Considerations

- When comparing a field to null, only the `=` and `!=` operators are supported.

Platform Event Field Considerations

- Deleting event fields—If a field is referenced in a filter expression, you can't delete it. If you delete it, you get an error.
- Deleting a custom platform event—If a custom platform event is referenced in a filter expression in a channel member, you can't delete the custom event definition.
- Renaming event fields—If you rename a field that's referenced in a filter expression, the filter continues to be applied correctly. The system maps the old field name to the renamed field. It's not necessary to update the field name in the filter expression. If you rename a field label, the field name doesn't change, and filtering continues to work correctly.

  Note: If a filter expression was created before Winter '23, renamed fields work only after you update the filter expression and save the channel member again.

- Namespace prefix—If a filter expression was created before an org had a namespace and the filter expression didn't contain the namespace prefix in the field names, the filter expression is automatically updated with the namespace prefix and continues to work.
- Changing field types—You can't change the type of a field that's referenced in a filter expression. If you change it, you get an error.
- Field name case in the filter expression—The names of fields used in a filter expression are case-insensitive. The case of field names in the filter expression and the platform event schema can differ.
- Missing event fields—If a filter expression references a valid field that isn't part of a published event message, the field is evaluated as null or false for a Checkbox field.

Event Delivery Usage for Filtered Streams

The event delivery allocation applies to the number of events delivered after the filter is applied and not before filtering. Because a filter can reduce the number of events delivered to a subscriber, using a filter helps lower a subscriber's usage of the event delivery allocation.

For example, a client subscribes to a channel to receive order events, and the event bus contains 100 such events to deliver. But the channel member for Order_Event__e has a filter that matches only order events with certain values for the city and amount fields. Out of the 100 order events, 15 match the field values and are delivered. The event delivery usage is in this case 15 events and not 100. For more information about the event delivery allocation, see Platform Event Allocations.

Filter Expressions in Channel Members

Add a filter expression in a channel member that's associated with a custom channel. The channel member associates a custom platform event with the channel and specifies the filter expression. The channel holds the filtered stream of event messages that match the filter expression for the specified custom platform event.

Let's walk through the steps to create a channel, a channel member, and a filter expression. Then we can subscribe to the channel to validate receiving the filtered event stream.

IN THIS SECTION:

Prerequisite: Define a Platform Event

The examples in this section depend on a predefined custom platform event called Order_Event__e. To define this event in Salesforce, complete these steps.

1. From Setup, in the Quick Find box, enter `Platform Events`, and then select **Platform Events**.
2. Click **New Platform Event**.
3. Provide these values.
   a. Label: `Order Event`
   b. Plural Label: `Order Events`
   c. Select **Starts with a vowel sound**, if available.
   d. Click **Save**.
4. Create four fields. In Custom Fields & Relationships, click **New** for each field, and follow the wizard.
   a. Field type: `Text`; Field Label: `Order Number`; Length: `18`
   b. Field type: `Text`; Field Label: `City`; Length: `50`
   c. Field type: `Number`; Field Label: `Amount`; Length: `16`; Decimal Places: `2`

EDITIONS

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To define a platform event:
- Customize Application

Add a Filter with Tooling API

Before you can add a filter, create a channel. Use PlatformEventChannel in Tooling API, and specify API version 56.0 or later.

You can use your preferred REST API tool to perform these steps. We recommend using Postman with the Salesforce Platform APIs collection, which contains handy templates for Salesforce API calls. To set up Postman, see Quick Start: Connect Postman to Salesforce in Trailhead.

USER PERMISSIONS

To create or update PlatformEventChannel and PlatformEventChannelMember objects:
- Customize Application

To use REST API:
- API Enabled

1. To create a channel, send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannel
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and click **Create channel**.

2. Use this example request body. To have this channel accept platform events, `event` is specified for `channelType`.

```
{
  "FullName": "MyChannel__chn",
  "Metadata": {
    "channelType": "event",
    "label": "Custom Channel for Platform Events"
  }
}
```

You receive a response similar to this example response.

```
{
  "id" : "0YLRM000000004X4AQ",
  "success" : true,
  "errors" : [ ],
  "warnings" : [ ],
  "infos" : [ ]
}
```

3. Add a channel member that specifies the custom platform event and filter expression. This example references the custom platform event, Order_Event__e. Send a POST request to this URI.

```
/services/data/v61.0/tooling/sobjects/PlatformEventChannelMember
```

If you're using Postman, expand **Event Platform** > **Custom Channels** > **Platform Event**, and click **Create channel member**.

4. Use this example request body.

```
{
 "FullName": "MyChannel_chn_Order_Event_e",
 "Metadata": {
  "eventChannel": "MyChannel__chn",
  "filterExpression": "(City__c LIKE 'S%' OR City__c='New York') AND Amount__c>10.50",
  "selectedEntity": "Order_Event__e"
 }
}
```

You receive a response similar to this example response.

```
{
  "id" : "0v8RM0000000MnNYAU",
  "success" : true,
  "errors" : [ ],
  "warnings" : [ ],
  "infos" : [ ]
}
```

To update a filter expression, perform a PATCH request to `/services/data/v61.0/tooling/sobjects/PlatformEventChannelMember/<ChannelMemberID>`, and pass in the entire request body with the new filter expression. You can update only the `filterExpression` field of a channel member. All other fields aren't updateable.

If your Salesforce org has a namespace, prepend the namespace prefix to each field used in `filterExpression` and the `selectedEntity` value in the PlatformEventChannelMember request body. For example, if the namespace is `ns`, the request body in this example becomes:

```
{
  "FullName": "MyChannel_chn_Order_Event_e",
  "Metadata": {
    "eventChannel": "MyChannel__chn",
    "filterExpression": "(ns__City__c LIKE 'S%' OR ns__City__c='New York') AND
ns__Amount__c>10.50",
    "selectedEntity": "ns__Order_Event__e"
  }
}
```

SEE ALSO:

Tooling API Developer Guide: PlatformEventChannel

Tooling API Developer Guide: PlatformEventChannelMember

Add a Filter with Metadata API

We recommend using Metadata API as part of the application lifecycle management process to develop, test, deploy, and release your apps to production. If you want to create the channel and filter expression, we recommend that you use Tooling API with REST.

Before you add a filter, create a channel. Use PlatformEventChannel in Metadata API, and specify API version 56.0 or later.

To create a channel and channel member with Metadata API, you can use tools such as Visual Studio Code with the Salesforce Extension pack or Salesforce CLI. For more information, see Metadata API Developer Tools and Quick Start: Metadata API in the *Metadata API Developer Guide*.

This sample custom channel definition is for the `MyChannel__chn` channel. The file name is `MyChannel__chn.platformEventChannel`. To have this channel accept platform events, `event` is specified for `channelType`.

USER PERMISSIONS

To deploy and retrieve metadata types:
- Customize Application

To update metadata types:
- Modify Metadata Through Metadata API Functions

To use Metadata API:
- API Enabled

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <channelType>event</channelType>
```

```
    <label>Custom Channel for Platform Events</label>
</PlatformEventChannel>
```

This channel member specifies the custom platform event and filter expression. This sample channel member definition associates the custom platform event to the channel and adds a filter expression. The file name is `MyChannel_chn_Order_Event_e.platformEventChannelMember`.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>MyChannel__chn</eventChannel>
    <filterExpression><![CDATA[(City__c LIKE 'S%' OR City__c='New York') AND
Amount__c>10.50]]>
    </filterExpression>
    <selectedEntity>Order_Event__e</selectedEntity>
</PlatformEventChannelMember>
```

Note: If the filter expression contains the `<` and `&` special characters, they aren't allowed in XML data in their literal form. Escape those characters as `<` and `&`, or enclose the entire filter expression value within the `<![CDATA[...]]>` section. Although no special characters are present in the previous example, `<![CDATA[...]]>` is included for convenience. For more information, see CData sections in the Extensible Markup Language (XML) specification.

If your Salesforce org has a namespace, prepend the namespace prefix to each field used in `filterExpression` and the `selectedEntity` value in the PlatformEventChannelMember request body. For example, if the namespace is `ns`, the request body in this example becomes:

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>MyChannel__chn</eventChannel>
    <filterExpression><![CDATA[(ns__City__c LIKE 'S%' OR ns__City__c='New York') AND
ns__Amount__c>10.50]]>
    </filterExpression>
    <selectedEntity>ns__Order_Event__e</selectedEntity>
</PlatformEventChannelMember>
```

This package.xml file references the channel and channel member.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyChannel__chn</members>
        <name>PlatformEventChannel</name>
    </types>
    <types>
        <members>MyChannel_chn_Order_Event_e</members>
        <name>PlatformEventChannelMember</name>
    </types>
    <version>61.0</version>
</Package>
```

To update a filter expression, redeploy the package with an updated value for the `filterExpression` field in the PlatformEventChannelMember component. You can update only the `filterExpression` field of a channel member. All other fields aren't updateable.

SEE ALSO:

Metadata API Developer Guide: PlatformEventChannel

Metadata API Developer Guide: PlatformEventChannelMember

Subscribe to the Channel and Receive the Filtered Event Stream

After configuring the filter, subscribe to the channel and receive the event messages that match the filter expression. Only Pub/Sub API and CometD clients support stream filtering. Because Apex triggers, flows, and processes don't support custom channels, you can't use them to subscribe to filtered event streams.

Note: Before subscribing to the channel, follow the steps in the previous sections to create the `MyChannel__chn` channel, and configure a filter expression for Order_Event__e with Tooling API or Metadata API.

In this example, we use a Pub/Sub API Java client sample. See Java Quick Start for Publishing and Subscribing to Events in the Pub/Sub API Developer Guide. If you prefer to use a tool in the Salesforce UI, you can use the empApi Lightning component or the Streaming Monitor app from AppExchange.

1. Set up the Pub/Sub API Java client and subscribe to the channel.
   a. Follow the steps in Java Quick Start for Publishing and Subscribing to Events in the Pub/Sub API Developer Guide.
   b. In Step 3: Configure Client Parameters, for the `TOPIC` argument, provide the custom channel that you created: `/event/MyChannel__chn`. The channel name format is `/event/`***`ChannelName__chn`***.
   c. From the `java` folder, run: `./run.sh genericpubsub.Subscribe`.
2. Now that you're subscribed to the custom channel, publish event messages.
   a. Open another Terminal window.
   b. Navigate to your local `pub-sub-api` folder.
   c. In an IDE, such as Visual Studio Code, open `java/src/main/java/utility/CommonContext.java`.
   d. Modify the `createEventMessages` method. Replace the code with this snippet. Replace the `CreatedById` value with your Salesforce user ID. See Find the Salesforce ID for a User or Profile.

```
public List<GenericRecord> createEventMessages(Schema schema, final int numEvents)
{

    String[] orderNumbers = {"99","100","101","102","103"};
    String[] cities = {"Los Angeles", "New York", "San Francisco", "San Jose",
                       "Boston"};
    Double[] amounts = {35.0, 20.0, 2.0, 123.0, 180.0};

    // Update CreatedById with the appropriate User Id from your org.
    List<GenericRecord> events = new ArrayList<>();
    for (int i=0; i<numEvents; i++) {
        events.add(new GenericRecordBuilder(schema)
                .set("CreatedDate", System.currentTimeMillis())
                .set("CreatedById", "<User_Id>")
                .set("Order_Number__c", orderNumbers[i % 5])
                .set("City__c", cities[i % 5])
```

```
                .set("Amount__c", amounts[i % 5]).build());
    }

    return events;
}
```

e. For the configuration parameters in `java/src/main/resources/arguments.yaml`, supply these values.

- `TOPIC: /event/Order_Event__e`
- `NUMBER_OF_EVENTS_TO_PUBLISH: 5`
- `SINGLE_PUBLISH_REQUEST: true`

f. Build the client from the top-level `java` folder with this command: `mvn clean install`.

g. To run the PublishStream RPC example and publish five event messages from the `java` folder, enter `./run.sh genericpubsub.PublishStream`.
As a refresher, here's the filter expression that we set in the previous section.

```
(City__c LIKE 'S%' OR City__c='New York') AND Amount__c>10.50
```

From the event messages published, only the second and fourth event messages match the filter criteria given in the previous example and are delivered to the client. The other event messages only partially match the criteria, so they aren't delivered to the client.

This example shows the two event messages received after subscribing to the filtered channel, `/event/MyChannel__chn`.

```
2024-03-26 17:04:38,347 [pool-3-thread-1]
java.lang.Class - Received event with payload:
{
  "CreatedDate": 1711497876867,
  "CreatedById": "0055f000005mc66AAA",
  "Order_Number__c": "100",
  "City__c": "New York",
  "Amount__c": 20.0
}
with schema name: Order_Event__e

2024-03-26 17:04:38,347 [pool-3-thread-2]
java.lang.Class - Received event with payload:
{
  "CreatedDate": 1711497876867,
  "CreatedById": "0055f000005mc66AAA",
  "Order_Number__c": "102",
  "City__c": "San Jose",
  "Amount__c": 123.0
}
with schema name: Order_Event__e
```

Get Custom Channels and Channel Members

You can find which channels and channel members are set up in your Salesforce org by performing SOQL queries through Tooling API.

To perform SOQL queries, make a REST `query` call or use the Query Editor in the Developer Console with the **Tooling API** option selected. To make REST calls using Postman, set up Postman with the Salesforce Platform APIs collection. See Quick Start: Connect Postman to Salesforce in Trailhead.

Perform a GET request to this endpoint with the SOQL query appended.

USER PERMISSIONS

To query PlatformEventChannel and PlatformEventChannelMember Tooling objects:
- View Setup and Configuration

To use REST with Tooling API:
- API Enabled

```
/services/data/v61.0/tooling/query?q=<query>
```

This query returns all the custom channels.

```
SELECT Id, DeveloperName, ChannelType, MasterLabel FROM PlatformEventChannel
```

If you're using Postman, expand **Event Platform** > **Custom Channels**, and then click **List event channels**.

Sample result:

Id
0YLRM000000004X4AQ

DeveloperName
MyChannel

ChannelType
event

MasterLabel
Custom Channel for Platform Events

And this query returns all the channel members.

```
SELECT Id, DeveloperName,EventChannel,FilterExpression, SelectedEntity FROM
PlatformEventChannelMember
```

If you're using Postman, expand **Event Platform** > **Custom Channels**, and then click **List channel members**.

Sample result (the `SelectedEntity` field references the ID of the custom platform event):

Id
0v8RM0000000MnNYAU

DeveloperName
MyChannel_chn_Order_Event_e

EventChannel
0YLRM000000004X4AQ

FilterExpression

```
(City__c LIKE 'S%' OR City__c='New York') AND Amount__c>10.50
```

SelectedEntity
01IRM0000006mrs2AA

Obtain a Platform Event's Subscribers

View a list of all triggers or processes that are subscribed to a platform event by using the Salesforce user interface or the API.

Note: CometD and Pub/Sub API subscribers to a platform event channel aren't exposed in the user interface or the API. Flow Pause element subscribers to a platform event aren't returned in Metadata API.

IN THIS SECTION:

View and Manage an Event's Subscribers on the Platform Event's Detail Page

View the triggers, flows, and processes that are subscribed to a platform event in the Subscriptions related list. Manage subscriptions for Apex triggers.

View Event Subscribers

View a list of all triggers, processes, and platform event–triggered flows that are subscribed to a platform event in the Subscriptions related list. CometD subscribers, such as your own CometD client or the `empApi` Lightning component, and Pub/Sub API subscribers aren't listed in this page.

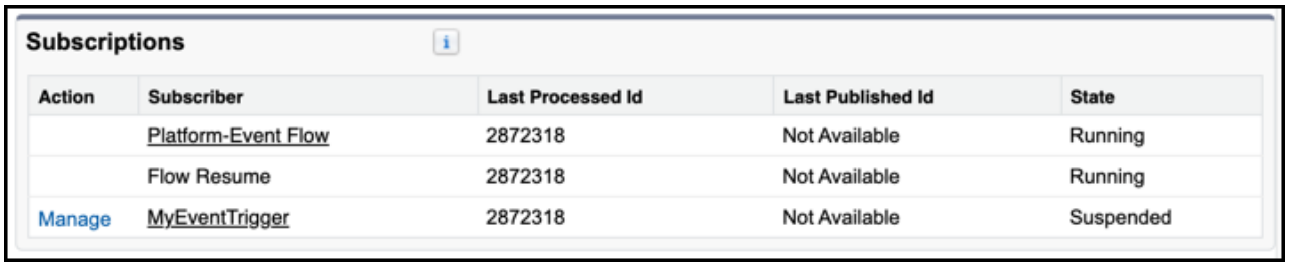
1. From Setup, enter *`Platform Events`* in the Quick Find box, then select **Platform Events**.
2. Click your event's name.

   On the event's definition page, the Subscriptions related list shows all the active triggers, processes, and platform event–triggered flows that are subscribed to the platform event.

   Note: Only one "Flow Resume" subscriber appears in the Subscriptions related list for all paused flow interviews that are subscribed to the platform event. Processes and platform event–triggered flows are listed individually.

3. To access a subscriber's definition, click the subscriber name in the Subscriptions related list. For a trigger, details include its implementation and API version. For a process, details include its version number and API name.

   Note: Why are you seeing flow version details when you click a process? Similar to a flow, a running instance of a process is a flow interview. The information that you see on the Flow Version page is about the process. You can click the flow API name of the process to view the list of processes for your org.



Subscriptions

Action	Subscriber	Last Processed Id	Last Published Id	State
	Platform-Event Flow	2872318	Not Available	Running
	Flow Resume	2872318	Not Available	Running
Manage	MyEventTrigger	2872318	Not Available	Suspended

The list shows the replay ID of the event that the system last processed (Last Processed Id field) and the event last published (Last Published Id field). Knowing which replay ID was last processed is useful when there's a gap in the events published and processed. For example, if a trigger contains complex logic that causes a delay in processing large batches of incoming events.

Note: For high-volume platform events, the Last Published Id value isn't available and is always shown as Not Available.

Subscription States

Also, the Subscriptions list shows the state of each subscriber, which can be one of the following.

- `Running`—The subscriber is actively listening to events. If you modify the subscriber, the subscription continues to process events.
- `Error`— The subscriber was disconnected and stopped receiving published events. A trigger reaches this state when it exceeds the number of maximum retries with the `EventBus.RetryableException`. Trigger assertion failures and unhandled exceptions don't cause the error state. We recommend limiting the retries to fewer than nine times to avoid reaching this state. When you fix and save the trigger, or for a managed package trigger, if you redeploy the package, the trigger resumes automatically from the tip, starting from new events. Also, you can resume a trigger subscription in the subscription detail page that you access from the platform event page.
- `Suspended`—The subscriber is disconnected and can't receive events because a Salesforce admin suspended it or due to an internal error. You can resume a trigger subscription in the subscription detail page that you access from the platform event page. To resume a process, deactivate it and then reactivate it. If you modify the subscriber, the subscription resumes automatically from the tip, starting from new events.

Suspend or Resume an Apex Trigger Subscription

Resume a suspended trigger subscription where it left off, starting from the earliest event message that is available in the event bus. If you want to bypass event messages that are causing errors or are no longer needed, you can resume the subscription from the tip, starting from new event messages.

To manage a trigger subscription:

1. In the Subscriptions related list, click **Manage** next to the Apex trigger.
2. In the subscription detail page, choose the appropriate action.
   - To suspend a running subscription, click **Suspend**.
   - To resume a suspended subscription, starting from the earliest event message that is available in the event bus, click **Resume**.
   - To resume a suspended subscription, starting from new event messages, click **Resume from Tip**.

You can't manage subscriptions for flows and processes through the Subscriptions related list.

Note:

- After you click **Resume** or **Resume from Tip**, there can be a delay of a few minutes before the subscription resumes.

- After you modify a subscriber, the subscription resumes automatically. For more information, see the Subscription States section.
- If you click **Resume** for a trigger that's in the error state, the trigger skips the events that were retried with `EventBus.RetryableException`. The subscription starts with the unprocessed events sent after the error state was reached and that are within the retention window.

Obtain Processes That Subscribe to a Platform Event in Metadata API

Use Metadata API to retrieve all processes subscribed to a platform event.

1. Retrieve all event subscriptions in your org with this sample package manifest.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>*</members>
      <name>EventSubscription</name>
   </types>
   <version>45.0</version>
</Package>
```

2. In each .subscription file, look at the `referenceData` parameter. The value is the API name of a process.

Example: In this .subscription file, referenceData points to version 4 of the Printer_Management process.

```
<?xml version="1.0" encoding="UTF-8"?>
<EventSubscription xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <eventType>Printer_Status__e</eventType>
    <referenceData>Printer_Management_4</referenceData>
</EventSubscription>
```

Obtain an Event's Subscribers by Querying EventBusSubscriber

The EventBusSubscriber standard object contains information about the trigger and process subscribers of all platform events. You can query this object using SOQL.

For more information, see EventBusSubscriber in the *Object Reference for the Salesforce Platform*.

Identify and Match Event Messages with the `EventUuid` Field

Delivered platform event messages include the `EventUuid` field, which identifies an event message. Use this field to match published and received event messages by comparing the universally unique identifiers (UUIDs) of the received events with the UUIDs returned in the SaveResult of publish calls. This way, you can find any event messages that aren't delivered and republish them.

Note: If you use Apex to publish events, use Apex publish callbacks to track the final result of the `EventBus.publish` call. For more information, see Get the Result of Asynchronous Platform Event Publishing with Apex Publish Callbacks.

The `EventUuid` field is a universally unique identifier (UUID) that identifies a platform event message. The system populates the `EventUuid` field, and you can't overwrite its value. The `EventUuid` field is available in CometD clients using API version 52.0 and later. The API version corresponds to the version that an Apex trigger is saved with or the version specified in a CometD subscriber

endpoint. The `EventUuid` field isn't part of the event schema, which is returned by the REST `eventSchema` resource or the describe call result. The `EventUuid` field is available for high-volume and standard-volume platform events.

For Pub/Sub API clients, the event `id` field contains the event UUID value. There's no field named `EventUuid`. The `id` field is present in all Pub/Sub API clients, isn't versioned, and can be overwritten. For more information, see the PublishStream RPC Method in the Pub/Sub API documentation.

For all publishing methods except for Pub/Sub API, event publishing is asynchronous. A success status in an immediately returned SaveResult means that the publish operation is queued in Salesforce. The operation is carried out later when system resources are available. Some failures, such as validation or limit errors, are returned in the SaveResult, but not asynchronous errors. In rare cases, enqueued publish operations can fail due to a system error, and the event message isn't delivered. You can use the `EventUuid` field to determine which enqueued event messages failed to publish and then republish them.

Publishing with Pub/Sub API is synchronous and the returned response contains the final publishing status.

Get the Event UUID of Published Event Messages

Before you can compare the UUIDs of published and received event messages, first save the UUID of published event messages. Also, save the corresponding event field values so that you can republish the events if needed.

If you publish the event using Salesforce APIs, the SaveResult returned contains the UUID in the Error `message` field. This example contains the save result of an event inserted using a REST API POST request.

```
{
  "id" : "e01xx0000000001AAA",
  "success" : true,
  "errors" : [ {
    "statusCode" : "OPERATION_ENQUEUED",
    "message" : "e981b488-81f3-4fcc-bd6f-f7033c9d7ac3",
    "fields" : [ ]
  } ]
}
```

If you publish the event in Apex, you can obtain the UUID by calling this method: `EventBus.getOperationId(saveResult)`.

Note: If the event publish request fails to be enqueued in Salesforce, no event UUID is returned in the SaveResult. As a result, the `EventBus.getOperationId(result)` Apex method returns null. This behavior applies even to events that have a prepopulated UUID and that were created via `SObjectType.newSObject(recordTypeId, loadDefaults)`.

This example gets the UUID from the event publish call using Apex.

Prerequisites: Before you can run this example, define a platform event with the label of Order Event and these fields: Order Number of type Text(10) and Has Shipped of type Checkbox.

```
// Publish a high-volume event message
Order_Event__e evt = new Order_Event__e(
    Order_Number__c='17',
    Has_Shipped__c = false);
Database.SaveResult sr = EventBus.publish(evt);
// Inspect immediate result
if (sr.isSuccess() == true) {
    System.debug('Successfully enqueued event for publishing.');
    // Get the UUID that uniquely identifies this event publish
    System.debug('UUID=' + EventBus.getOperationId(sr));
} else {
   for(Database.Error err : sr.getErrors()) {
       System.debug('Error returned: ' +
```

```
                    err.getStatusCode() +
                    ' - ' +
                    err.getMessage());
    }
}

// Debug message output:
//|DEBUG|Successfully enqueued event for publishing.
//|DEBUG|UUID=6ba5db7e-c27b-4a67-a3c5-cf425ffcaf53
```

Get the Event UUID from Received Event Messages in a CometD Client

In a CometD client, the received event message contains the event UUID in the `EventUuid` field in the event subsection, as shown in this JSON event example.

```
{
  "schema": "UIovjRagY-xEDIJ1Ehzafg",
  "payload": {
    "CreatedDate": "2021-03-04T18:31:40.517Z",
    "CreatedById": "005RM00000231cZYAQ",
    "Order_Number__c": "17",
    "Has_Shipped__c": false
  },
  "event": {
     "EventUuid": "e981b488-81f3-4fcc-bd6f-f7033c9d7ac3",
     "replayId": 617
  }
}
```

Get the Event UUID from Received Event Messages in a Pub/Sub API Client

In a Pub/Sub API client, the received event message contains the event UUID value in the `id` field of the event instance. For example, you can retrieve the UUID value in your code by accessing the id field on the event instance as follows.

```
event.id
```

The returned value is a UUID similar to this example.

```
4c45a27a-5d86-47ed-881a-878b9a9c0dcc
```

Get the Event UUID from Received Event Messages in an Apex Trigger

In an Apex trigger, extract the event UUID by accessing the `EventUuid` field on the event object.

```
trigger OrderEventTrigger on Order_Event__e (after insert) {
    for(Order_Event__e evt: Trigger.New) {
        // Get the event UUID
        String EventUuid = evt.EventUuid;
        System.debug('Received event UUID=' + EventUuid);

        // Store the event UUID for matching with published event UUID
        // . . .
    }
```

```
}

// Debug message output:
//|DEBUG|Received event UUID=6ba5db7e-c27b-4a67-a3c5-cf425ffcaf53
```

Match UUIDs of Published and Received Event Messages

After you obtain the event UUIDs for both published and received event messages, match the UUIDs. Any UUIDs that don't match can indicate that the event hasn't been delivered. You can attempt to republish the unmatched event messages.

SEE ALSO:

Publish Event Messages with Apex

Testing Your Platform Event in Apex

Add Apex tests to test platform event subscribers. Before you can package or deploy Apex code, including triggers, to production, it must have tests and sufficient code coverage. Add Apex tests to provide code coverage for your triggers.

IN THIS SECTION:

Event and Event Bus Properties in Test Context

In test context, event messages and the event bus have different properties. State information of events and subscribers is reset and isn't persisted.

Deliver Test Event Messages

Deliver test event messages after the `Test.stopTest()` statement. Alternatively, deliver test event messages at any time with the `Test.getEventBus().deliver()` method. Fail test event messages with the `Test.getEventBus().fail()` method.

Test Retried Event Messages

An Apex trigger can retry processing of an event message by throwing `EventBus.RetryableException`. In API version 43.0 and later, you can test retried event messages by calling `Test.EventBus.deliver()` and inspecting `EventBusSubscriber` fields.

SEE ALSO:

Apex Developer Guide : Testing and Code Coverage

Event and Event Bus Properties in Test Context

In test context, event messages and the event bus have different properties. State information of events and subscribers is reset and isn't persisted.

Test Events and the Test Event Bus

When an Apex test publishes an event message, it's published to a test event bus that is separate from the Salesforce event bus. In an Apex test, state information of events and subscribers is reset, as follows.

- The event replay ID value is reset to 0 and starts from 1 for the first test event message.

- Event state information in EventBusSubscriber is reset. The last processed replay ID (`EventBusSubscriber.Position`) and the last published replay ID (`EventBusSubscriber.Tip`) are reset to 0.
- When test events are published and processed in subscribers, event state information is updated.
- Subscriber status is reset to `Running` (`EventBusSubscriber.Status`).
- You can query EventBusSubscriber to get event state. For example, the following SOQL query gets some information about all trigger subscribers to the Order_Event__e event.

```
SELECT Name, Position, Retries, LastError
FROM EventBusSubscriber
WHERE Topic='Order_Event__e' AND Type='ApexTrigger'
```

After an Apex test finishes executing, state information of events and subscribers reverts to the non-test values.

Test Events and Limits

Event allocations don't apply to test events, which have their own publishing limit of 500 event messages in a test method. If the number of event messages published from an Apex test context exceeds the limit, an error is returned with the `LIMIT_EXCEEDED` status code. The error is in the `SaveResult` that the `EventBus.publish` Apex method returns.

Testing Event Subscribers

Use an Apex test to test publishing and subscribing to a platform event. When you publish an event message in an Apex test, event subscribers are notified and start execution, including:

- Apex triggers
- Processes (when using an Apex test class saved with API version 43.0 or later)
- Flows (when using in an Apex test class saved with API version 43.0 or later)

Apex tests don't cause CometD-based or Pub/Sub API subscribers to run.

SEE ALSO:

Event-Driven Software Architecture

Object Reference for the Salesforce Platform: EventBusSubscriber

Deliver Test Event Messages

Deliver test event messages after the `Test.stopTest()` statement. Alternatively, deliver test event messages at any time with the `Test.getEventBus().deliver()` method. Fail test event messages with the `Test.getEventBus().fail()` method.

Deliver Test Event Messages After `Test.stopTest()`

To publish platform event messages in an Apex test, enclose the publish statements within `Test.startTest()` and `Test.stopTest()` statements. Call the `EventBus.publish()` method within the `Test.startTest()` and `Test.stopTest()` statements. In test context, the `EventBus.publish()` method enqueues the publish operation. The `Test.stopTest()` statement causes the event publishing to be carried out and event messages to be delivered to the test event

bus. Include your validations after the `Test.stopTest()` statement. For example, you can validate that a subscribed Apex trigger or a subscribed flow Pause element has performed the expected actions, like creating a Salesforce record.

```
// Create test events
Test.startTest();
// Publish test events with EventBus.publish()
Test.stopTest();
// Perform validations
```

Example: This sample test class contains two test methods. The `testValidEvent()` method checks that the event was successfully published and fires the associated trigger. The `testInvalidEvent()` method verifies that publishing an event with a missing required field fails, and no trigger is fired. The `testValidEvent()` method creates one Low_Ink__e event. After `Test.stopTest()`, it executes a SOQL query to verify that a case record is created, which means that the trigger was fired. The second test method follows a similar process but for an invalid test.

This example requires that the Low_Ink__e event and the associated trigger are defined in the org.

```
@isTest
public class EventTest {
    @isTest static void testValidEvent() {

        // Create a test event instance
        Low_Ink__e inkEvent = new Low_Ink__e(Printer_Model__c='MN-123',
                                             Serial_Number__c='10013',
                                             Ink_Percentage__c=0.15);

        Test.startTest();

        // Publish test event
        Database.SaveResult sr = EventBus.publish(inkEvent);

        Test.stopTest();

        // Perform validations here

        // Verify SaveResult value
        System.assertEquals(true, sr.isSuccess());

        // Verify that a case was created by a trigger.
        List<Case> cases = [SELECT Id FROM Case];
        // Validate that this case was found
        System.assertEquals(1, cases.size());
    }

    @isTest static void testInvalidEvent() {

        // Create a test event instance with invalid data.
        // We assume for this test that the Serial_Number__c field is required.
        // Publishing with a missing required field should fail.
        Low_Ink__e inkEvent = new Low_Ink__e(Printer_Model__c='MN-123',
                                             Ink_Percentage__c=0.15);

        Test.startTest();

        // Publish test event
```

```
        Database.SaveResult sr = EventBus.publish(inkEvent);

        Test.stopTest();

        // Perform validations here

        // Verify SaveResult value - isSuccess should be false
        System.assertEquals(false, sr.isSuccess());

        // Log the error message
        for(Database.Error err : sr.getErrors()) {
            System.debug('Error returned: ' +
                        err.getStatusCode() +
                        ' - ' +
                        err.getMessage()+' - '+err.getFields());
        }

        // Verify that a case was NOT created by a trigger.
        List<Case> cases = [SELECT Id FROM Case];
        // Validate that this case was found
        System.assertEquals(0, cases.size());
    }
}
```

Deliver Test Event Messages on Demand with `Test.getEventBus().deliver()`

You can control when test event messages are delivered to subscribers by calling `Test.getEventBus().deliver()`. Use `Test.getEventBus().deliver()` to deliver test event messages multiple times, and verify that subscribers have processed the test events each step of the way. Delivering event messages multiple times is useful for testing sequential processing of events. For example, you can verify sequential actions of a subscriber in a loop within the same test.

Enclose `Test.getEventBus().deliver()` within the `Test.startTest()` and `Test.stopTest()` statement block.

```
Test.startTest();
// Create test events
// ...
// Publish test events with EventBus.publish()
// ...
// Deliver test events
Test.getEventBus().deliver();
// Perform validations
// ...
Test.stopTest();
```

Also, you can call `Test.getEventBus().deliver()` in an Apex test method outside the `Test.startTest()` and `Test.stopTest()` statement block. Doing so enables you to test event messages with asynchronous Apex.

```
Test.startTest();
// Do some tests
Test.stopTest();

// Deliver test events
Test.getEventBus().deliver();
```

Deliver Test Event Messages Published from Asynchronous Apex

When testing a batch Apex job that publishes BatchApexErrorEvent on failure, use the `Test.startTest()` and `Test.stopTest()` statement block with `Test.getEventBus().deliver()`. The `Test.stopTest()` call ensures that the asynchronous Apex job executes after this statement. Next, `Test.getEventBus().deliver()` delivers the event message that the failed batch job published.

This snippet shows how to execute a batch Apex job and deliver event messages. It executes the batch job after `Test.stopTest()`. This batch job publishes a BatchApexErrorEvent message when a failure occurs through the implementation of `Database.RaisesPlatformEvents`. After `Test.stopTest()` runs, a separate `Test.getEventBus().deliver()` statement is added so that it can deliver the BatchApexErrorEvent.

```
try {
    Test.startTest();
    Database.executeBatch(new SampleBatchApex());
    Test.stopTest();
    // Batch Apex job executes here
} catch(Exception e) {
    // Catch any exceptions thrown in the batch job
}

// The batch job fires BatchApexErrorEvent if it fails, so deliver the event.
Test.getEventBus().deliver();
```

Asynchronous Apex also includes queueable Apex and future methods. If a platform event message is published from within those async Apex jobs, they're delivered after `Test.stopTest()`. It's not necessary to add `Test.getEventBus().deliver();`. The next example shows how to deliver a platform event message that a queueable Apex job publishes. After `Test.stopTest()`, the queueable job is executed and the event message is delivered.

```
Test.startTest();
System.enqueueJob(new SampleQueueableApex());
Test.stopTest();
// Queueable Apex job executes here.
// The platform event message published by the job is delivered too.
```

Note: If further platform events are published by downstream processes, add `Test.getEventBus().deliver();` to deliver the event messages for each process. For example, if a platform event trigger, which processes the event from the Apex job, publishes another platform event, add a `Test.getEventBus().deliver();` statement to deliver the event message.

Example: Deliver Event Messages Individually

This test class publishes an `Order_Event__e` event message and delivers it using `Test.getEventBus().deliver()`. It verifies that the trigger processed the event message and created a task. A duplicate event message (an event with the same `Event_ID__c` custom field value) is published and delivered. The test verifies that the trigger didn't create a task for the duplicate event.

Before you can run this test class, define a platform event with the name of `Order_Event__e` and these fields: `Event_ID__c` of type Text, `Order_Number__c` of type Text, `Has_Shipped__c` of type Checkbox. Also, in the `OrderTrigger` trigger, replace the user `Name` field value in the SOQL query with a user's full name in your Salesforce org, such as `John Smith`.

```
@isTest
public class MyTestClassDeliver {

    @isTest static void doSomeTesting() {
```

```
        Test.startTest();

        // Publish a test event
        Order_Event__e event = new Order_Event__e(
              Event_ID__c='123AB', Order_Number__c='12346', Has_Shipped__c=true);
        Database.SaveResult sr = EventBus.publish(event);

        // Verify that the publish was successful
        System.assertEquals(true, sr.isSuccess());

        // Deliver the test event before Test.stopTest()
        Test.getEventBus().deliver();

        // Check that the case that the trigger created is present.
        List<Task> tasks = [SELECT Id FROM Task];
        // Validate that this task was found.
        // There is only one test task in test context.
        Integer taskCount = tasks.size();
        System.assertEquals(1, taskCount);

        // Publish a duplicate event
        Order_Event__e dupEvent = new Order_Event__e(
              Event_ID__c='123AB', Order_Number__c='12346', Has_Shipped__c=true);
        Database.SaveResult sr2 = EventBus.publish(dupEvent);

        // Verify that the publish was successful.
        System.assertEquals(true, sr2.isSuccess());

        Test.getEventBus().deliver();

        // Get all tasks in test context
        List<Task> tasksNew = [SELECT Id FROM Task];
        // Validate that no task was created and
        // the number of tasks should not have changed.
        System.assertEquals(taskCount, tasksNew.size());

        Test.stopTest();

    }
}
```

This example trigger processes `Order_Event__e` event messages that the test class publishes.

Note: Because this trigger performs a SOQL query for each event notification received, the Apex governor limit for SOQL queries can be hit.

```
trigger OrderTrigger on Order_Event__e (after insert) {
    // List to hold all cases to be created.
    List<Task> tasks = new List<Task>();

    // Get user Id for case owner
    User usr = [SELECT Id FROM User WHERE Name='<Replace with FirstName LastName>' LIMIT
1];
```

```
    // Iterate through each notification.
    for (Order_Event__e event : Trigger.New) {
        if (event.Has_Shipped__c == true) {
            // Create task only if it doesn't exist yet for the same order
            String eventID = '%' + event.Event_ID__c;
            List<Task> tasksFromQuery =
                [SELECT Id FROM Task WHERE Subject LIKE :eventID];
            if (tasksFromQuery.size() == 0) {
                Task t = new Task();
                t.Priority = 'Medium';
                t.Subject = 'Follow up on shipped order ' + event.Order_Number__c +
                    ' for event ID ' + event.Event_ID__c;
                t.OwnerId = usr.Id;
                tasks.add(t);
            }
        }
    }
    // Insert all tasks in the list.
    if (tasks.size() > 0) {
        insert tasks;
    }

}
```

Fail the Publishing of Event Messages to Test Apex Publish Callbacks

Apex publish callbacks contain the final result of asynchronous `EventBus.publish` calls. To test your Apex publish callback class, you can simulate the failure of a publish call with `Test.getEventBus().fail()`.

In an Apex test, event messages are published synchronously in the test event bus. To can simulate the execution of the callback methods in a test, you can deliver or fail the publishing of the event messages. This section covers the failure of event publishing.

The `Test.getEventBus().fail()` method causes the publishing of events to fail immediately after the call and event messages are removed from the test event bus. This method causes the `onFailure()` method in the callback class to be invoked. When the event messages fail to publish, none of the triggers defined on the platform event receive any failed events.

This example class is a test class for the `FailureAndSuccessCallback` class that is given in Get the Result of Asynchronous Platform Event Publishing with Apex Publish Callbacks. This test class shows how to test the failed delivery of test event messages in the test event bus. Before you run this test class, define a platform event in Setup with the label `Order Event` and a Text field of `Order Number`.

```
@isTest
public class MyCallbackTest {
    @isTest static void testFailedEventsWithFail() {

        // Publish with callback
        FailureAndSuccessCallback cb = new FailureAndSuccessCallback();

        // Create test event with EventUuid field value
       Order_Event__e event = (Order_Event__e)Order_Event__e.sObjectType.newSObject(null,
 true);
        event.Order_Number__c='100';
        System.debug('EventUuid of created event: ' + event.EventUuid);
        // Publish an event with callback
```

```
        EventBus.publish(event, cb);

        // Fail event
        // (invoke onFailure and DO NOT deliver event to subscribers)
        Test.getEventBus().fail();

        // Verify that tasks were created by the onFailure() method
        List<Task> tasksFailed =
            [SELECT Id,Subject,Description FROM Task
             WHERE Subject='Follow up on event publishing failures.'];
        System.Assert.areEqual(1,tasksFailed.size(),
                               'Unexpected number of tasks received for failed publishing');

        System.debug('tasksFailed[0].Description=' + tasksFailed[0].Description);
        System.debug('event.EventUuid=' + event.EventUuid);
        System.Assert.isTrue(tasksFailed[0].Description.contains(event.EventUuid),
                               'EventUuid was not found in the Description field.');

    }
}
```

To deliver event messages successfully, check out these sections.

- Deliver Test Event Messages After Test.stopTest()
- Deliver Test Event Messages on Demand with Test.getEventBus().deliver()

SEE ALSO:

Apex Developer Guide : Using Limits, `startTest`, and `stopTest`

Test Retried Event Messages

An Apex trigger can retry processing of an event message by throwing `EventBus.RetryableException`. In API version 43.0 and later, you can test retried event messages by calling `Test.EventBus.deliver()` and inspecting `EventBusSubscriber` fields.

To force redelivery of a retried event message in an Apex test, call `Test.EventBus.deliver()`. This method also delivers other event messages that have been published after the last `deliver()` call.

In API version 43.0 or later, you can check these new EventBusSubscriber fields to test retried triggers.

- `Retries`
- `LastError`

The `EventBusSubscriber.Retries` field indicates how many times a trigger was retried.

`EventBusSubscriber.LastError` indicates the error message that was passed to the `throw` statement that executed last (`throw new EventBus.RetryableException('Error Message')`).

Note: When `EventBus.RetryableException` is thrown, `EventBusSubscriber.Position` isn't incremented because the trigger didn't successfully process the event message.

Example: This test method delivers a test event message that fires a trigger. The associated event trigger throws `EventBus.RetryableException` twice. The test verifies that the trigger was retried twice by querying `EventBusSubscriber` and checking the `Retries` field value.

Before you can run this test class, define a platform event with the name of `Order_Event__e` and the following fields: `Order_Number__c` of type Text and `Has_Shipped__c` of type Checkbox. This test class assumes there is an associated trigger called `OrderTriggerRetry` that retries the event. The trigger is not provided in this example.

```
@isTest
public class MyTestClassRetryDoc {

    @isTest static void doSomeTesting() {

        Test.startTest();

        // Publish a test event
        Order_Event__e event = new Order_Event__e(
              Order_Number__c='12345', Has_Shipped__c=true);
        Database.SaveResult sr = EventBus.publish(event);
        // Deliver the initial event message.
        // This will fire the associated event trigger.
        Test.getEventBus().deliver();

        // Trigger retries event twice, so loop twice
        for(Integer i=0;i<2;i++) {
            // Get info about all subscribers to the event
            EventBusSubscriber[] subscribers =
                [SELECT Name, Type, Position, Retries, LastError
                 FROM EventBusSubscriber WHERE Topic='Order_Event__e'];

            for (EventBusSubscriber sub : subscribers) {
                System.debug('sub.Retries=' + sub.Retries);
                System.debug('sub.lastError=' + sub.lastError);
                if (sub.Name == 'OrderTriggerRetry') {
                    System.assertEquals(i+1, sub.Retries);
                }
            }

            // Deliver the retried event
            Test.getEventBus().deliver();
        }

        Test.stopTest();

    }
}
```

SEE ALSO:

Retry Event Triggers with EventBus.RetryableException

Encrypting Platform Event Messages at Rest in the Event Bus

For increased security, you can enable encryption of platform event messages while they're stored in the event bus in a Shield Encryption org.

When you enable encryption of platform events in a Shield Encryption org, event messages are encrypted using the key that is based on the event bus tenant secret type. The encrypted event messages are stored in the event bus for up to 3 days (or 1 day for standard-volume events). The encryption applies to all custom and standard platform events, including Salesforce Event Monitoring streamed events.

To enable encryption and delivery of platform events, first create an event bus tenant secret on the Key Management page in Setup. Then enable encryption of platform events on the Encryption Policy page.

If you don't enable encryption of platform events in a Shield Encryption org, event messages are stored in clear text in the event bus.

Decrypting Platform Event Messages Before Delivery

Before delivering a platform event message to a subscribed client, the event payload is decrypted using the encryption key. The platform event message is sent over a secure channel using HTTPS and TLS, which ensures that the data is protected and encrypted while in transit. If the encryption key was rotated and a new key is issued, stored event messages are not re-encrypted, but they are decrypted before delivery using the archived key. If a key is destroyed, stored event messages can't be decrypted and aren't delivered.

Note: Classic Encryption is not supported.

Error Status Code

If you enable encryption and an event message could not be published due to an encryption failure, the publish operation returns the `PLATFORM_EVENT_ENCRYPTION_ERROR` status code. For more information, see Platform Event Error Status Codes.

Enable Encryption of Platform Events

To enable encryption of platform event messages at rest, generate an event bus tenant secret and then enable encryption.

USER PERMISSIONS

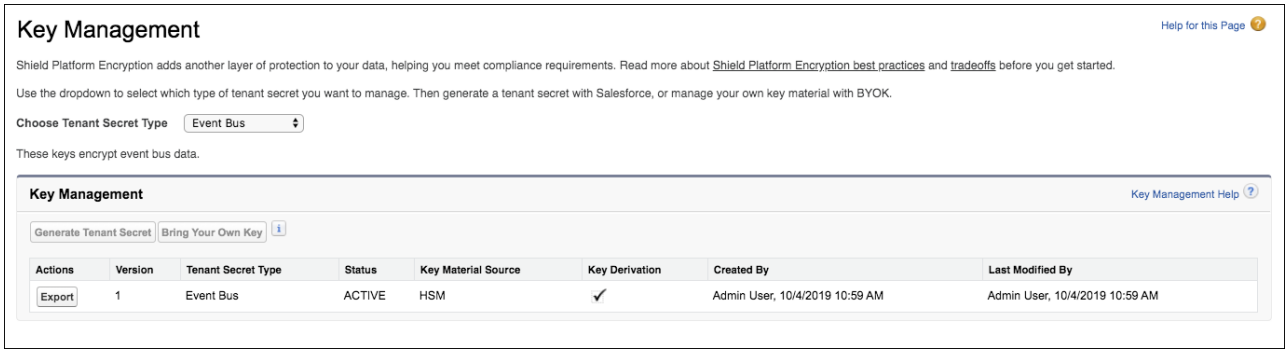
To manage tenant secrets:
- Manage Encryption Keys

Prerequisites:

- A Shield Platform Encryption org.
- Only authorized users can generate tenant secrets from the Platform Encryption page. Ask your Salesforce admin to assign the Manage Encryption Keys permission to you.

Steps:

1. To generate an event bus tenant secret, from Setup, in the Quick Find box, enter *`Platform Encryption`*, and then select **Key Management**.
2. In the Choose Tenant Secret Type dropdown list, choose **Event Bus**.
3. Click **Generate Tenant Secret** or, to upload a customer-supplied tenant secret, click **Bring Your Own Key**.



Note:

- If your org has no tenant secrets, perform Step 3 before Step 2.
- You can generate or rotate an event bus tenant secret once every 7 days.
- You can also generate a tenant secret through SOAP API or REST API using the TenantSecret object and the Type field value of EventBus. For more information, see TenantSecret in the *Object Reference for Salesforce and Lightning Platform*.

4. To enable encryption, from Setup, in the Quick Find box, enter `Platform Encryption`, and then select **Encryption Policy**.
5. Select **Encrypt and deliver change data capture events and platform events**.

   Note: You can access and control this setting in Metadata API, in PlatformEncryptionSettings. Ensure that the event bus tenant secret is created before setting `enableEventBusEncryption` to true.

6. Click **Save**.

When you enable encryption for platform events, you also enable it for change data capture events. For more information, see Change Events for Encrypted Salesforce Data in the *Change Data Capture Developer Guide*.

Monitor Platform Event Publishing and Delivery Usage

To get usage data for event publishing and delivery to Pub/Sub API and CometD clients, `empApi` Lightning components, and event relays, query the PlatformEventUsageMetric object. In API 58.0 and later, enable and use Enhanced Usage Metrics to get granular usage data for various time segments. If Enhanced Usage Metrics isn't enabled, usage data is available for the last 24 hours, ending at the last hour, and for historical daily usage. PlatformEventUsageMetric is available in API version 50.0 and later.

Important: We recommend that you use Enhanced Usage Metrics. With Enhanced Usage Metrics, you can query usage data at a granular level. You can break down usage metrics by event name, client ID, event type, and usage type. And you can get usage data by various time segments, including daily, hourly, and 15-minute periods. See Enhanced Usage Metrics.

Use PlatformEventUsageMetric to get visibility into your event usage and usage trends. The usage data gives you an idea of how close you are to your allocations and when you need more allocations. The usage metrics stored in PlatformEventUsageMetric are separate from the REST API limits values. Use the REST API limits to track your monthly delivery and publishing usage against your allocations. The event delivery usage that the limits API returns is common for platform events and change data capture events in CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays. PlatformEventUsageMetric breaks down usage of platform events and change data capture events so that you can track their usage separately.

Because dates are stored in Coordinated Universal Time (UTC), convert your local dates and times to UTC for the query. For the date format to use, see Date Formats and Date Literals in the *SOQL and SOSL Reference*.

Usage data is stored for at least 45 days. Usage data is updated hourly and is available only when usage is nonzero for a 24-hour period. Usage data isn't available for 1-hour intervals or any other arbitrary interval. The only supported intervals are the last 24 hours and daily data. Also, usage data isn't available for standard-volume platform events.

After a Salesforce major upgrade, usage data can be inaccurate for the day and the last 24 hours within the upgrade window. New usage data overwrites the data for the hour that the 5-minute upgrade occurs in. The new usage data includes metrics that start after the upgrade for that hour. For more information about Salesforce upgrades, see Salesforce Upgrades and Maintenance in *Help* and Salesforce Status.

For platform events, you can query usage data for these metrics. The first value is the metric name value that you supply in the query.

- `PLATFORM_EVENTS_PUBLISHED`—Number of platform events published
- `PLATFORM_EVENTS_DELIVERED`—Number of platform events delivered to CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays

For change data capture events, you can query usage data for these metrics. The first value is the metric name value that you supply in the query.

- `CHANGE_EVENTS_PUBLISHED`—Number of change data capture events published
- `CHANGE_EVENTS_DELIVERED`—Number of change data capture events delivered to CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays

Note: Even though usage data is available for the number of change events published through the `CHANGE_EVENTS_PUBLISHED` metric, no event publishing limit is enforced for change events. Publishing allocations apply only to platform events.

This table lists which events, publishers, and subscribers are included for each metric.

Metric	Events	Publishers and Subscribers
Event publishing metrics: `PLATFORM_EVENTS_PUBLISHED` and `CHANGE_EVENTS_PUBLISHED`	All types of events including: • Custom and standard platform events • Change events	All publishing methods including: • Apex • Salesforce APIs including Pub/Sub API • Flows • Process Builder processes
Event delivery metrics: `PLATFORM_EVENTS_DELIVERED` and `CHANGE_EVENTS_DELIVERED`	• Custom and standard platform events that count towards the delivery allocation. Delivery metrics don't include Event Monitoring real-time events and some standard events. To check which standard platform event is counted against the delivery allocation, see Standard Platform Event Object List. • Change events	Subscription methods: • CometD, Pub/Sub API, `empApi` Lightning components, and event relays • Delivery metrics don't include event deliveries to Apex triggers, flows, and processes.

Obtain Usage Metrics for the Last 24 Hours

To get usage metrics for the last 24 hours, ending at the last hour, perform a query by specifying the start and end date and time in UTC, and the metric name.

For the last 24-hour period, the end date is the current date in UTC, with the time rounded down to the previous hour. The start date is 24 hours before the end date. Dates have hourly granularity.

Example: Based on the current date and time of August 4, 2020 11:23 in UTC, the last hour is 11:00. The query includes these dates.

- Start date in UTC format: 2020-08-03T11:00:00.000Z
- End date in UTC format: 2020-08-04T11:00:00.000Z

This query returns the usage for the number of platform events delivered between August 3, 2020 at 11:00 and August 4, 2020 at 11:00.

```
SELECT Name, StartDate, EndDate, Value FROM PlatformEventUsageMetric
WHERE Name='PLATFORM_EVENTS_DELIVERED'
AND StartDate=2020-08-03T11:00:00.000Z AND EndDate=2020-08-04T11:00:00.000Z
```

The query returns this result for the last 24-hour usage.

Name	StartDate	EndDate	Value
PLATFORM_EVENTS_DELIVERED	2020-08-03T11:00:00.000+0000	2020-08-04T11:00:00.000+0000	575

The time span between StartDate and EndDate is 24 hours for the stored 24-hour usage. You can specify either StartDate or EndDate in the query and get the same result.

Obtain Historical Daily Usage Metrics

To get daily usage metrics for 1 or more days, perform a query by specifying the start date and end date in UTC and the metric name.

Example: To get usage metrics for a period of 3 days, from July 19 to July 22, 2020, use these start and end dates. Time values are 0.

- Start date for the query: 2020-07-19T00:00:00.000Z
- End date for the query: 2020-07-22T00:00:00.000Z

This query selects usage metrics for the number of platform events delivered for a 3-day period.

```
SELECT Name, StartDate, EndDate, Value FROM PlatformEventUsageMetric
WHERE Name='PLATFORM_EVENTS_DELIVERED'
AND StartDate>=2020-07-19T00:00:00.000Z and EndDate<=2020-07-22T00:00:00.000Z
```

The query returns these results for the specified date range.

Name	StartDate	EndDate	Value
PLATFORM_EVENTS_DELIVERED	2020-07-19T00:00:00.000+0000	2020-07-20T00:00:00.000+0000	575
PLATFORM_EVENTS_DELIVERED	2020-07-20T00:00:00.000+0000	2020-07-21T00:00:00.000+0000	899
PLATFORM_EVENTS_DELIVERED	2020-07-21T00:00:00.000+0000	2020-07-22T00:00:00.000+0000	1,035

General Considerations

If you query the `Id` of PlatformEventUsageMetric, the `Id` value returned isn't a valid record ID. For example, this query returns an Id field value of `000000000000000AAA`.

```
SELECT Id, Name, StartDate, EndDate, Value FROM PlatformEventUsageMetric WHERE
Name='PLATFORM_EVENTS_DELIVERED'
```

As a result, you can't use PlatformEventUsageMetric in batch Apex with QueryLocator because QueryLocator requires valid record IDs to be passed in to the `execute` method. Using PlatformEventUsageMetric with batch Apex and QueryLocator causes unexpected results. Instead, use an iterable with batch Apex and PlatformEventUsageMetric. For more information, see Using Batch Apex in the *Platform Events Developer Guide*.

IN THIS SECTION:

Enhanced Usage Metrics

In API version 58.0 and later, enable Enhanced Usage Metrics to get granular usage data and time segments in PlatformEventUsageMetric queries. You can break down usage metrics by event name, client ID, event type, and usage type. And you can get usage data by granular time segments, including daily, hourly, and 15-minute periods.

SEE ALSO:

Object Reference for Salesforce and Lightning Platform: PlatformEventUsageMetric

Enhanced Usage Metrics

In API version 58.0 and later, enable Enhanced Usage Metrics to get granular usage data and time segments in PlatformEventUsageMetric queries. You can break down usage metrics by event name, client ID, event type, and usage type. And you can get usage data by granular time segments, including daily, hourly, and 15-minute periods.

Note: Enhanced Usage Metrics isn't available in Government Cloud.

USER PERMISSIONS

To query the PlatformEventUsageMetric `Client`, `EventName`, `EventType`, and `UsageType` fields:

- View All Data

Enable Enhanced Usage Metrics

Before you can get more usage metrics, enable Enhanced Usage Metrics in Metadata API. Set the `enableEnhancedUsageMetrics` field to true in PlatformEventSettings. See PlatformEventSettings in the *Metadata API Developer Guide*.

Query Example: Get Usage Metrics for the Last 24 Hours Aggregated by Event Name

If you don't specify the `StartDate` and `EndDate` in your query, the query returns data for the last 24 hours by default. The example query aggregates the results per event because the `EventName` field is specified in the SELECT statement. Also, the query aggregates the data per hour as specified by the `TimeSegment` field. The query also includes the event type and the usage type.

Tip: Before you run this query, use Metadata API to enable Enhanced Usage Metrics. In the PlatformEventSettings metadata type, set the `enableEnhancedUsageMetrics` field to `true`. See PlatformEventSettings in the *Metadata API Developer Guide*.

```
SELECT EventName, EventType, UsageType, Value, StartDate, EndDate
FROM PlatformEventUsageMetric
WHERE TimeSegment='Hourly'
```

In this sample result, usage data for published and delivered events is returned for all events: Order_Event__e and AccountChangeEvent. The query aggregates usage data per hour.

EventName	EventType	UsageType	Value	StartDate	EndDate
Order_Event__e	CUSTOM_PLATFORM_EVENT	DELIVERY	1154	2023-04-01T00:00:00.000+0000	2023-04-01T01:00:00.000+0000
Order_Event__e	CUSTOM_PLATFORM_EVENT	DELIVERY	1316	2023-04-01T01:00:00.000+0000	2023-04-01T02:00:00.000+0000
Order_Event__e	CUSTOM_PLATFORM_EVENT	PUBLISH	577	2023-04-01T00:00:00.000+0000	2023-04-01T01:00:00.000+0000
Order_Event__e	CUSTOM_PLATFORM_EVENT	PUBLISH	658	2023-04-01T01:00:00.000+0000	2023-04-01T02:00:00.000+0000
AccountChangeEvent	CHANGE_EVENT	PUBLISH	15	2023-04-01T01:00:00.000+0000	2023-04-01T02:00:00.000+0000
AccountChangeEvent	CHANGE_EVENT	DELIVERY	15	2023-04-01T01:00:00.000+0000	2023-04-01T02:00:00.000+0000

For valid `TimeSegment` values, check the `TimeSegment` field of PlatformEventUsageMetric in the *Object Reference for the Salesforce Platform*.

You can refine the query results by adding fields in the WHERE clause. To view usage data for only a specific usage type, add the UsageType field in the WHERE clause. For example, to query for only delivered event usage, add this condition.

```
WHERE UsageType='DELIVERY'
```

Or add this condition for published events.

```
WHERE UsageType='PUBLISH'
```

You can narrow the query by event type. For example, to query for custom platform events only, add this condition.

```
WHERE EventType='CUSTOM_PLATFORM_EVENT'
```

Or add this condition for change events.

```
WHERE UsageType='CHANGE_EVENT'
```

You can query usage for one event only. For example:

```
WHERE EventName='Order_Event__e'
```

Query Rules

- If `StartDate` and `EndDate` aren't specified in the WHERE clause, the query defaults to the last 24-hour period.
- You must specify the `StartDate` and `EndDate` field values in the WHERE clause or neither. If only `StartDate` or `EndDate` are specified, you get an error.
- The maximum time span between `StartDate` and `EndDate` is 30 days.
- The minimum time span between `StartDate` and `EndDate` is 15 minutes.
- The `StartDate` field can refer to a date that is no more than 60 days old.
- The `TimeSegment` field must always be specified in the query's WHERE clause. Optionally, it can also be part of the SELECT statement.
- Make sure that the time span between `StartDate` and `EndDate` in the WHERE clause is valid for the `TimeSegment` value chosen. Check the `TimeSegment` field of PlatformEventUsageMetric in the *Object Reference for the Salesforce Platform*.

- A query must have at least one of `Name`, `EventType`, or `EventName` fields in either the SELECT or WHERE clause.
- A query that uses `EventName` or `EventType` must also specify the `UsageType` in either the SELECT or WHERE clause.

Query Considerations

- We recommend that you include `StartDate` and `EndDate` in the query's SELECT statement. Including these fields helps you interpret the query results and map each result with its corresponding time segment.
- To make sure the query covers all time segments between the start date and end date, use the `>=` and `<=` logical operators with `StartDate` and `EndDate` in the WHERE clause. For example: `StartDate >= DateTime1 AND EndDate <= DateTime2`
- Date fields accept date literals, such as `LAST_WEEK`, in addition to date values. For more information, see Date Formats and Date Literals in WHERE in the *SOQL and SOSL Reference*.
- The maximum number of rows that can be returned in a query for enhanced usage metrics is 2,000. If the query generates more than 2,000 rows, you get an error and the query returns no results.
- The LIMIT clause isn't supported.
- SOQL aggregate functions, such as `SUM()` and `MAX()`, aren't supported. For more information, see Aggregate Functions in the *SOQL and SOSL Reference*.

Drill Into a Time Slot with the Highest Usage

The examples in this section follow a scenario that starts with a multiday time range and drills down into smaller time slots to find the time slot with the highest usage. The first example gets daily usage. The second and third examples drill down into hourly and 15-minute usage.

One of the fields these examples use is the `Client` field. The `Client` field is populated for subscriber clients for event delivery usage. For publisher clients, the `Client` field is populated if the client ID is available. Otherwise, it's empty. The example query results contain placeholder values for the `Client` field for simplicity.

Get Daily Usage Metrics Aggregated by Event Name and Client

This example query gets daily usage metrics for delivered events grouped by event name and client for a period of 2 days. The query aggregates the results per event and client because the `EventName` and `Client` fields are specified in the SELECT statement.

Tip: Before you run this query, use Metadata API to enable Enhanced Usage Metrics. In the PlatformEventSettings metadata type, set the `enableEnhancedUsageMetrics` field to `true`. See PlatformEventSettings in the *Metadata API Developer Guide*.

```
SELECT EventName, EventType, Client, Value, StartDate, EndDate
FROM PlatformEventUsageMetric
WHERE TimeSegment='Daily'
AND UsageType='DELIVERY'
AND StartDate >= 2023-04-01T00:00:00.000Z
AND EndDate <= 2023-04-03T00:00:00.000Z
```

In this sample result, usage data is returned for all events: Order_Event__e and AccountChangeEvent. The query aggregates usage data by client. Two clients receive Order_Event__e events, and the usage data is computed for each. AccountChangeEvent events are received by one client only.

EventName	EventType	Client	Value	StartDate	EndDate
Order_Event__e	CUSTOM_PLATFORM_EVENT	client1	31327	2023-04-01T00:00:00.000+0000	2023-04-02T00:00:00.000+0000

EventName	EventType	Client	Value	StartDate	EndDate
Order_Event__e	CUSTOM_PLATFORM_EVENT	client1	20801	2023-04-02T00:00:00.000+0000	2023-04-03T00:00:00.000+0000
Order_Event__e	CUSTOM_PLATFORM_EVENT	client2	399	2023-04-01T00:00:00.000+0000	2023-04-02T00:00:00.000+0000
Order_Event__e	CUSTOM_PLATFORM_EVENT	client2	27	2023-04-02T00:00:00.000+0000	2023-04-03T00:00:00.000+0000
AccountChangeEvent	CHANGE_EVENT	client3	1009	2023-04-01T00:00:00.000+0000	2023-04-02T00:00:00.000+0000
AccountChangeEvent	CHANGE_EVENT	client3	780	2023-04-02T00:00:00.000+0000	2023-04-03T00:00:00.000+0000

Get Hourly Usage Metrics for One Event

Query hourly usage to view event usage for delivered events by hour for a time period up to 24 hours. This example query gets usage metrics for one event, Order_Event__e. The query aggregates the results into 1-hour intervals as specified by the `TimeSegment` field. Results are grouped per event and client because the `EventName` and `Client` fields are specified in the SELECT statement.

In the previous daily usage example, April 1 has the highest usage. To drill down into the usage for that day for one event, query for that date.

Tip: Before you run this query, use Metadata API to enable Enhanced Usage Metrics. In the PlatformEventSettings metadata type, set the `enableEnhancedUsageMetrics` field to `true`. See PlatformEventSettings in the *Metadata API Developer Guide*.

```
SELECT EventName,Client, Value, StartDate, EndDate
FROM PlatformEventUsageMetric
WHERE TimeSegment='Hourly'
AND UsageType='DELIVERY'
AND EventName='Order_Event__e'
AND StartDate >= 2023-04-01T00:00:00.000Z
AND EndDate <= 2023-04-02T00:00:00.000Z
```

In this sample result, hourly usage data is returned for Order_Event__e on April 1. The query aggregates usage data by client. Two clients receive Order_Event__e events, and the usage data is computed for each. A partial list of results is included for brevity.

EventName	Client	Value	StartDate	EndDate
Order_Event__e	client1	1136	2023-04-01T00:00:00.000+0000	2023-04-01T01:00:00.000+0000
Order_Event__e	client1	1301	2023-04-01T01:00:00.000+0000	2023-04-01T02:00:00.000+0000
Order_Event__e	client1	903	2023-04-01T02:00:00.000+0000	2023-04-01T03:00:00.000+0000
Order_Event__e	client2	17	2023-04-01T00:00:00.000+0000	2023-04-01T01:00:00.000+0000
Order_Event__e	client2	15	2023-04-01T01:00:00.000+0000	2023-04-01T02:00:00.000+0000
Order_Event__e	client2	13	2023-04-01T02:00:00.000+0000	2023-04-01T03:00:00.000+0000

Get Granular Usage Metrics for a 15-Minute Period

Get event usage aggregated into 15-minute periods for a time period up to 1 hour. This example query gets event delivery usage metrics for one event, Order_Event__e. The query aggregates the results into 15-minute intervals as specified by the `TimeSegment` field. Results are grouped per event and client because the `EventName` and `Client` fields are specified in the SELECT statement.

In the previous hourly usage example, the time period between the hours of 01:00:00 and 02:00:00 on April 1 has the highest usage. To drill down into the usage for that day for one event, query for that date.

Tip: Before you run this query, use Metadata API to enable Enhanced Usage Metrics. In the PlatformEventSettings metadata type, set the `enableEnhancedUsageMetrics` field to `true`. See PlatformEventSettings in the *Metadata API Developer Guide*.

```
SELECT EventName,Client, Value, StartDate, EndDate
FROM PlatformEventUsageMetric
WHERE TimeSegment='FifteenMinutes'
AND UsageType='DELIVERY'
AND EventName='Order_Event__e'
AND client='client1'
AND StartDate >= 2023-04-01T01:00:00.000Z
AND EndDate <= 2023-04-01T02:00:00.000Z
```

In this sample result, usage data for every 15 minutes is returned for Order_Event__e for client1 on April 1 between the hours of 01:00:00 and 02:00:00.

EventName	Client	Value	StartDate	EndDate
Order_Event__e	client1	321	2023-04-01T22:00:00.000+0000	2023-04-01T22:15:00.000+0000
Order_Event__e	client1	399	2023-04-01T22:15:00.000+0000	2023-04-01T22:30:00.000+0000
Order_Event__e	client1	265	2023-04-01T22:30:00.000+0000	2023-04-01T22:45:00.000+0000
Order_Event__e	client2	298	2023-04-01T22:45:00.000+0000	2023-04-01T23:00:00.000+0000

SEE ALSO:

Object Reference for the Salesforce Platform: PlatformEventUsageMetric

Platform Event Considerations

Learn about special behaviors related to defining, publishing, and subscribing to platform events. Learn how to test platform events. And get an overview of the various events that Salesforce offers.

IN THIS SECTION:

Considerations for Defining and Publishing Platform Events

Take note of the considerations when defining and publishing platform events.

Considerations for Subscribing to Platform Events with Processes and Flows

Before you use processes or flows to subscribe to platform events, familiarize yourself with these considerations.

Considerations for Publishing and Subscribing to Platform Events with Apex and APIs

Before you use Apex or Salesforce APIs to publish and subscribe to platform events, familiarize yourself with these considerations.

Decoupled Publishing and Subscription

When the publish behavior of a platform event is set to **Publish Immediately**, it's published outside of a Lightning Platform database transaction. As a result, the publishing and subscription processes are decoupled—the subscription process can't assume that an action made by the publishing transaction is committed before an event message is received. Familiarize yourself with some scenarios that can occur from the decoupled behavior.

What's the Difference Between the Salesforce Events?

Salesforce offers various features that use events, some of which are based on standard platform events. Other features are event-like but aren't event notifications.

Considerations for Defining and Publishing Platform Events

Take note of the considerations when defining and publishing platform events.

Considerations for Defining Platform Events

Field-Level Security

All platform event fields are read only by default, and you can't restrict access to a particular field. Field-level security permissions don't apply and the event message contains all fields.

Enforcement of Field Attributes

Platform event records are validated to ensure that the attributes of their custom fields are enforced. Field attributes include the Required and Default attributes, the precision of number fields, and the maximum length of text fields.

Permanent Deletion of Event Definitions

When you delete an event definition, it's permanently removed and can't be restored. Before you delete the event definition, delete the associated triggers. Published events that use the definition are also deleted.

Renaming Event Objects

Before you rename an event, delete the associated triggers. If the event name is modified after clients have subscribed to this event, the subscribed clients must resubscribe to the updated topic. To resubscribe to the new event, add your trigger for the renamed event object.

No Associated Tab

Platform events don't have an associated tab because you can't view event records in the Salesforce user interface.

No SOQL Support

You can't query event messages using SOQL.

No Record Page Support in Lightning App Builder

When creating a record page in Lightning App Builder, platform events that you defined show up in the list of objects for the page. However, you can't create a Lightning record page for platform events because event records aren't available in the user interface.

Platform Events in Package Uninstall

When uninstalling a package with the option **Save a copy of this package's data for 48 hours after uninstall** enabled, platform events aren't exported.

Event Volume in Package Installations and Upgrades

Installing a managed or unmanaged package that contains a standard-volume platform event causes the event type to be saved as high volume in the subscriber org. Upgrading a managed package doesn't change the event volume in the subscriber org.

No Support in Professional and Group Editions

Platform events aren't supported in Professional and Group Edition orgs. Installation of a package that contains platform event objects fails in those orgs.

Permissions for Defining and Using Platform Events

To define a custom platform event, you must have the Customize Application permission. For the permissions for publishing and subscribing to platform events, see Platform Event Permissions.

Considerations for Publishing Platform Events

Publishing Events in Read-Only Mode

During read-only mode, publishing standard-volume platform events results in an exception, and the events aren't published. Publishing high-volume platform events in read-only mode sometimes fails when the event schema isn't up to date in Salesforce. Your org is in read-only mode during Salesforce maintenance activities.

High-Volume Platform Event Persistence

Platform events are temporarily persisted to and served from an industry-standard distributed system during the retention period. A distributed system doesn't have the same semantics or guarantees as a transactional database. As a result, we can't provide a synchronous response for an event publish request. Events are queued and buffered, and Salesforce attempts to publish the events asynchronously. In rare cases, the event message might not be persisted in the distributed system during the initial or subsequent attempts. This means that the events aren't delivered to subscribers, and they aren't recoverable.

At-Least-Once Event Publishing and Duplicate Events

Asynchronous platform event publishing uses the at-least-once pub/sub model, not the exactly-once model. With the at-least-once model, there's a small chance of duplicate events, because if the system encounters an internal error when publishing the queued event, it retries the publishing. In rare cases, the system doesn't receive a publish acknowledgment, so the same event is published more than once. However, if publish acknowledgments are received as expected, no duplicate events are published.

A duplicate event is an event with the same `EventUuid` field value, a different `ReplayId` field value, and the same payload. We recommend you handle any issues that result from duplicate events in your subscriber. For example, you can implement deduplication logic using the `EventUuid` value and avoid processing duplicate events.

For more information about asynchronous event publishing, see Asynchronous Publishing.

Considerations for Subscribing to Platform Events with Processes and Flows

Before you use processes or flows to subscribe to platform events, familiarize yourself with these considerations.

Supported Platform Events

Processes and flows can subscribe to custom platform events and these standard platform events.

- AIPredictionEvent
- BatchApexErrorEvent
- FlowExecutionErrorEvent
- FOStatusChangedEvent
- OrderSummaryCreatedEvent
- OrderSumStatusChangedEvent
- PlatformStatusAlertEvent

Infinite Loops and Limits

Be careful when publishing events from processes or flows because you can get into an infinite loop and exceed limits. For example, a process is associated with the Printer Status platform event. The same process includes an action that creates a Printer Status event message. The process would trigger itself.

To avoid creating an endless loop in an event process, make sure that the new event message's field values don't meet the filter criteria for the associated criteria node.

Subscriptions Related List

On the platform event's detail page, the Subscriptions related list shows which entities are waiting to receive that platform event's messages. The related list includes a link to each subscribed process. If flow interviews are waiting for that platform event's messages, one "Process" subscriber appears in the Subscriptions related list.

Uninstalling Events

Before you uninstall a package that includes a platform event:

- Delete interviews that are waiting for that platform event's messages
- Deactivate processes that reference the event

Einstein Predictions

AIPredictionEvents are sent for every Einstein prediction result. To trigger your process or flow only by predictions on a specific object, use event condition filters. For example, if your process acts only on predictions written to Lead records, add a matching condition to check that the Lead ID field equals the AI Predicted Object ID event reference.

If your process or flow updates a field that is used by an Einstein prediction, Einstein will run the prediction again and write back new results. The new results generate a new AIPredictionEvent that could trigger your process or flow again, resulting in a loop. Avoid creating potential loops by only updating fields that aren't used in Einstein predictions.

Event Processes

These considerations apply only to event processes.

Apex Actions

You can't use an event reference to set an sObject variable in the Apex class.

Email Alerts Actions

Email alerts can't use values from platform event messages. For the process to send an email that contains values from the platform event message that starts the process, use this workaround:

1. Create an autolaunched flow.
2. In the flow, create a variable for each field in the platform event. Be sure to use compatible data types and make the variables available for input.
3. In the flow, add a Send Email action, and set the action's input variables with the flow variables.
4. In the process, add a Flows action and specify the autolaunched flow. Use event references to assign each platform event field to its corresponding flow variable.

Flows Actions

You can't use an event reference to set a record variable in the flow, even when the platform event is specified as the record variable's object. To pass values into the flow from the platform event message that starts the process, use this workaround:

1. In the flow, create a variable for each field in the platform event. Be sure to use compatible data types and make the variables available for input.
2. In the process, when you add the Flows action, use event references to assign each platform event field to its corresponding flow variable.

Packaging Event Processes

When you package an event process, the associated object isn't included automatically. Advise your subscribers to create the object, or manually add the object to your package.

Resumed Flow Interviews

These considerations apply only to flow interviews that resume when a platform event message is received.

Formulas

To reference a platform event in a flow formula, pass the event data into a record variable in the Pause element. Then reference the appropriate field in that record variable.

Event Condition Values

When you filter platform event messages, only the first 765 bytes of the condition value are used for filtering. Note that the number of characters will be smaller if you use multi-byte characters.

SEE ALSO:

Decoupled Publishing and Subscription

Considerations for Publishing and Subscribing to Platform Events with Apex and APIs

Before you use Apex or Salesforce APIs to publish and subscribe to platform events, familiarize yourself with these considerations.

Support Only for `after insert` Triggers

Only `after insert` triggers are supported for platform events because event notifications can't be updated. They're only inserted (published).

Infinite Trigger Loop and Limits

Be careful when publishing events from triggers because you could get into an infinite trigger loop and exceed limits. For example, if you publish an event from a trigger that's associated with the same event object, the trigger is fired in an infinite loop.

Publishing Events in Apex with Text Fields Set to Empty Strings

If you publish an event in Apex with a Text field set to an empty string, the field value in the delivered event message is null instead of empty string. The Text field value of empty string is preserved when publishing through other methods, including APIs, flows, and processes.

Platform Event Triggers: `OwnerId` Fields of New Records

In platform event triggers, if you create a Salesforce record that contains an `ownerId` field, the system populates the field with `Automated Process` by default. To set this field to another value, you can configure the trigger to run as another user. That way, the `OwnerId` field references the selected user. For more information, see Configure the User and Batch Size for Your Platform Event Trigger with PlatformEventSubscriberConfig. Alternatively, if you don't change the running user, you can set the `ownerId` field explicitly to the appropriate user when you create the record. This example explicitly populates the `ownerId` field for an opportunity with an ID obtained from another record.

```
Opportunity newOpp = new Opportunity(
    OwnerId = customerOrder.createdById,
    AccountId = acc.Id,
    StageName = 'Qualification',
    Name = 'A ' + customerOrder.Product_Name__c + ' opportunity for ' + acc.name,
    CloseDate = Date.today().addDays(7));
```

For cases and leads, you can alternatively use assignment rules for setting the owner. For more information, see AssignmentRuleHeader for SOAP API or Setting DML Options for Apex.

Platform Event Triggers: Changing the Opportunity OwnerId Field

If a platform event trigger updates the opportunity OwnerId field when opportunity splits are enabled, the trigger runs as the default Automated Process system user. A set of opportunity splits is created that totals 0%. The 0% split is invalid and must be 100% when an opportunity owner is changed. The 0% split causes validation errors when users attempt to update some opportunity fields, such as the Amount and Owner fields. To avoid these issues, configure the platform event trigger so that it runs as a different user. For more information, see Configure the User and Batch Size for Your Platform Event Trigger with PlatformEventSubscriberConfig

No Email Support from a Platform Event Trigger

With the default Automated Process running user, sending an email message from a platform event trigger using the `Messaging.SingleEmailMessage` class isn't supported. The email can't be sent because the sender is the Automated

Process entity, which has no email address. To send an email, change the running user of the trigger. For more information, see Configure the User and Batch Size for Your Platform Event Trigger with PlatformEventSubscriberConfig.

Replaying Past Events

You can replay platform events that were sent in the past. You can replay platform events through Pub/Sub API or Streaming API (CometD) but not Apex and other subscribers. For more information, see these resources.

- Java Quick Start in the *Pub/Sub API Developer Guide*.
- Event Message Durability in the *Pub/Sub API Developer Guide*.

Salesforce Maintenance Activities

On rare occasions, some Salesforce maintenance activities, such as an org migration to a new data center or an instance refresh, reset the stream of retained high-volume platform events. Because of the stream reset, the events are no longer available for replay. Also, the Replay ID of events published before the maintenance activity has no relation to the Replay ID of events after the activity. For more information, see How to Prepare for an Org Migration and Instance Refresh Maintenance.

Using MuleSoft's Salesforce Connector after a Hyperforce Migration

When a Salesforce instance is migrated to Hyperforce, the migration can result in an invalid Replay ID error in the Mule app. This error can be returned if the Mule app tries to access a Replay ID from the object store that's no longer valid on the new Hyperforce instance. See the knowledge article about steps to take to avoid the error.

Millisecond Time Precision in DateTime Fields

For event messages delivered to CometD clients in JSON format, the DateTime fields include the number of milliseconds. The date format, which is in the ISO 8601 standard, is `YYYY-MM-DDTHH:mm:ss.sssZ`. In API version 42.0 and earlier, DateTime fields don't include the millisecond part of the time, and the DateTime format is `YYYY-MM-DDTHH:mm:ssZ`.

For event messages delivered to Apex triggers, DateTime fields don't include millisecond precision, like DateTime fields of Salesforce objects.

Apex Trigger Subscriptions Disabled in Inactive Salesforce Orgs

If an org becomes inactive, all Apex trigger subscriptions are stopped and disabled. Triggers no longer process incoming event messages and can't process missed event messages. After the org is reactivated, new Apex trigger subscriptions are started when a platform event message is published.

SEE ALSO:

Platform Event Allocations

Decoupled Publishing and Subscription

When the publish behavior of a platform event is set to **Publish Immediately**, it's published outside of a Lightning Platform database transaction. As a result, the publishing and subscription processes are decoupled—the subscription process can't assume that an action made by the publishing transaction is committed before an event message is received. Familiarize yourself with some scenarios that can occur from the decoupled behavior.

Note: This decoupled behavior doesn't apply to platform events whose publish behavior is set to **Publish After Commit**.

Publisher Does Not Respect Transaction Boundaries

If an event is defined with a publish behavior of **Publish Immediately**, the publishing of the platform event message isn't transactional. As a result, a Salesforce record that an event publisher creates after publishing might not be committed to the database before the subscriber receives the event message. If the subscriber looks up the record, it might not be found because it hasn't been committed yet. For example, consider this scenario.

1. A Process Builder process publishes an event and creates a task record.
2. A trigger on the Task object runs some logic, which delays the commit of the task record.
3. A second Process Builder process, which is subscribed to the event, receives the event and looks up the newly created task. The process returns the following error because the trigger hasn't finished executing, and the record is not yet committed.

   ```
   "MyProcess process is configured to start when a MyEvent platform event message
   occurs. A MyEvent message occurred, but the process didn't start because no records
   in your org match the values specified in the process's Object node."
   ```

The example uses Process Builder, but the scenario applies to other methods of publishing and subscribing, such as the API and triggers.

Conversely, if a subscriber creates a Salesforce record after receiving an event message, the new record might not be found immediately after publishing. The reason is that the event is not processed synchronously after publishing, or the event processing might take a long time if the logic is complex.

Solution

The solution is to change the publishing behavior of the event to **Publish After Commit**. With this behavior, the event message is published after the first process creates the task record and the transaction finishes. The second process is able to find the task record.

Event Published from a Trigger

Consider an `after insert` trigger on a Salesforce object that publishes an event defined with a publish behavior of **Publish Immediately**. The event can be processed before the Salesforce record in the trigger is committed to the database. For example, consider this scenario.

1. An `after insert` trigger on a custom object publishes an event message.
2. A Process Builder process is subscribed to the event. The process is fired before the trigger finishes execution and before it commits the new custom object record.
3. The process tries to look up the record to match the event and fails because the record is not found.

Solution

The solution is to change the publishing behavior of the event to **Publish After Commit**. With this behavior, the event message is published after the trigger creates the custom object record and the transaction commits. The second process that receives the published event message is able to find the new record that the first process created.

What's the Difference Between the Salesforce Events?

Salesforce offers various features that use events, some of which are based on standard platform events. Other features are event-like but aren't event notifications.

Custom Events

You can use the following types of events to generate and deliver custom messages.

Custom Platform Events
: Use custom platform events to deliver secure, scalable, and customizable event notifications within Salesforce or from external sources. Custom platform event fields are defined in Salesforce and determine the data that you send and receive. Apps can publish and subscribe to platform events on the Lightning Platform or in external systems.

Generic Events
: Generic events are custom events that contain arbitrary payloads. With a generic event, you can't define the schema of the event.

Data Events

The following types of events are tied to Salesforce records.

Change Data Capture Events
Salesforce publishes Change Data Capture events for record and field changes.

PushTopic Events
PushTopic events track field changes in Salesforce records and are tied to Salesforce records.

Custom and Data Event Comparison

For a comparison of custom and data events, see Streaming Event Features in the Streaming API Developer Guide.

Standard Events: Security, Apex, and Monitoring

Salesforce publishes the following examples of standard platform events. These predefined events enable monitoring of security-related actions and user actions in Salesforce.

Asset Token Events
Subscribe to an AssetTokenEvent stream to monitor OAuth 2.0 authentication activity. Salesforce publishes an asset token event upon successful completion of an OAuth 2.0 asset token flow for a connected device.

Batch Apex Error Events
Subscribe to an BatchApexErrorEvent stream to catch errors that occur during batch Apex job execution. You can receive all types of errors and exceptions, including uncatchable exceptions, such as Apex limit exceptions.

Real-Time Event Monitoring
Real-Time Event Monitoring provides standard platform events that you can subscribe to for monitoring user activity in real time, such as logins and running reports. For example, you can subscribe to the event channel for LoginEventStream to receive notifications when users log in.

Event-Like Features

The following features can trick you into being streaming events, but they're not.

Event Monitoring Log
Like Real-Time Event Monitoring, you can use Event Monitoring to track user activity, such as logins and running reports. Unlike Real-Time Events, Event Monitoring doesn't send real-time notifications. Instead, it stores user activity in a log that you can query.

Transaction Security Policies
A transaction security policy evaluates user activity, such as logins and data exports, and trigger actions in real time. When a policy is triggered, notifications are sent through email or in-app notifications. You can use standard actions, such as blocking an operation, or custom actions defined in Apex.

Calendar Events
A calendar event is an appointment or meeting that you create and view in the user interface. In SOAP API, the Event object represents a calendar event. These events are calendar items and not notifications that software systems send.

SEE ALSO:

Standard Platform Event Objects

Examples

Check out platform event apps—an end-to-end example using flows, a Java client, and a sample app that covers a business scenario.

IN THIS SECTION:

End-to-End Example: Printer Supply Automation

This example demonstrates how to make sure that your office printers always have enough paper and ink by using two platform events and two flows.

Your company just received a shipment of "smart" printers. You configure the printers to send information to Salesforce. You build a flow that uses the received information to decide whether to order more ink or paper from the vendor. Also, you build another flow to schedule installation of the new supplies the day after they're delivered.

IN THIS SECTION:

Platform Events: Printer Status and Vendor Response

This example uses two platform events: one to hold the information coming from the printer (Printer Status) and one to hold the information coming from the vendor (Vendor Response).

The Printer Status platform event includes these custom fields.

API Name	Field Label	Data Type	Description
Serial_Number	Serial Number	Text	The printer's unique identifier. This value is used to locate the corresponding asset record.
Ink_Status	Ink Status	Text	Values: Full, Medium, Low, or Empty.
Paper_Level	Paper Level	Number	Paper level in percentage.
Total_Print_Count	Total Print Count	Number	Aggregate number of pages printed.

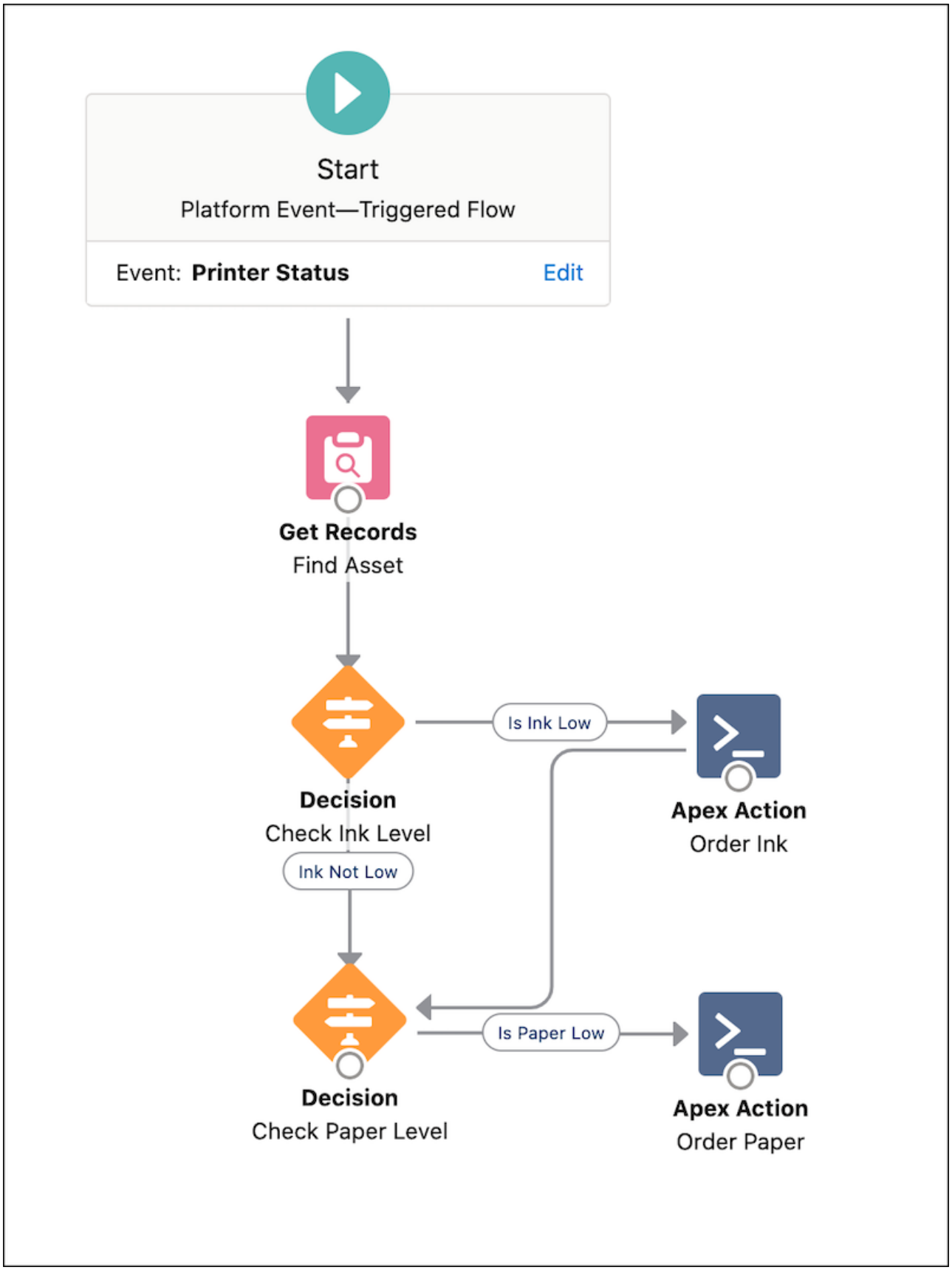
The Vendor Response platform event includes these custom fields.

API Name	Field Label	Data Type	Description
Order_Number	Order Number	Text	The order's unique identifier.
Expected_Delivery_Date	Expected Delivery Date	Date	The date when the vendor expects the order to be delivered
Order_Status	Order Status	Text	Values: Ordered, Confirmed, Shipped, Delivered, Delayed, Canceled.
Part_Label	Part Label	Text	The label of the part to order.
Part_Number	Part Number	Text	The part number of the part to order.
Serial_Number	Serial Number	Text	The printer's unique identifier. This value is sent in the order request and returned in the vendor response. It's used to locate the corresponding asset record.

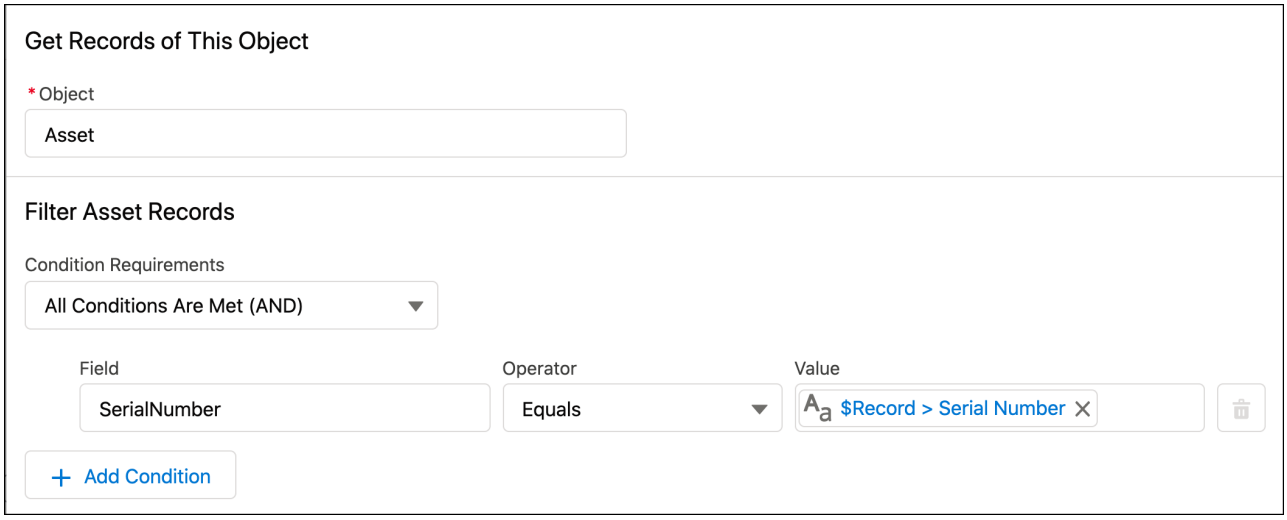
Flow: Automation for Printer Status Events

When the platform event–triggered flow receives a Printer Status event, the flow finds the asset record that's associated with the printer. The flow evaluates whether the printer has low ink or paper, and if so, calls an Apex action to order ink or another action to order paper.

The flow starts when it receives a Printer Status platform event message.

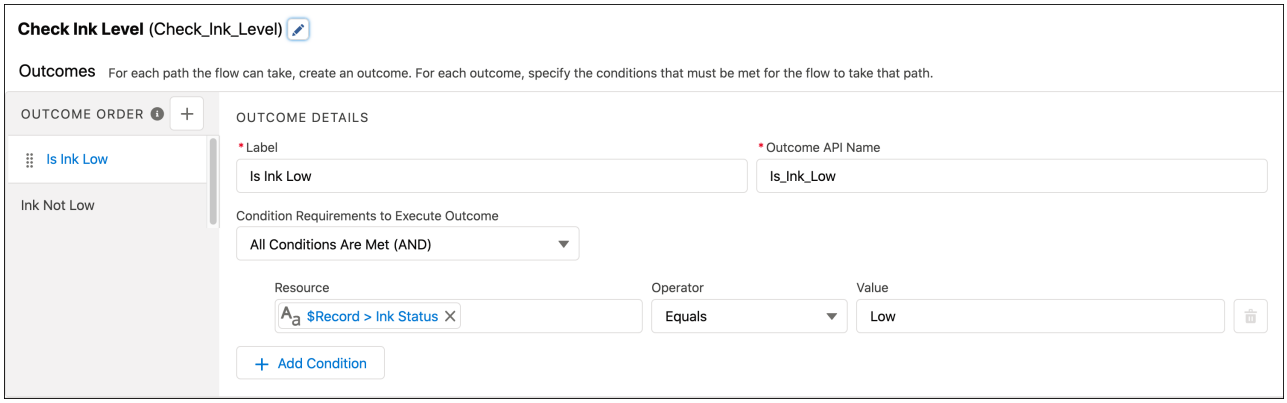

Start
Platform Event—Triggered Flow
Event: Printer Status
Edit
Get Records
Find Asset
Decision
Check Ink Level
Is Ink Low
Apex Action
Order Ink
Ink Not Low
Decision
Check Paper Level
Is Paper Low
Apex Action
Order Paper

The Get Records element finds the related asset record by matching the asset's serial number with that of the incoming event message. The Get Records element provides us with the asset record fields that we use later in the flow.

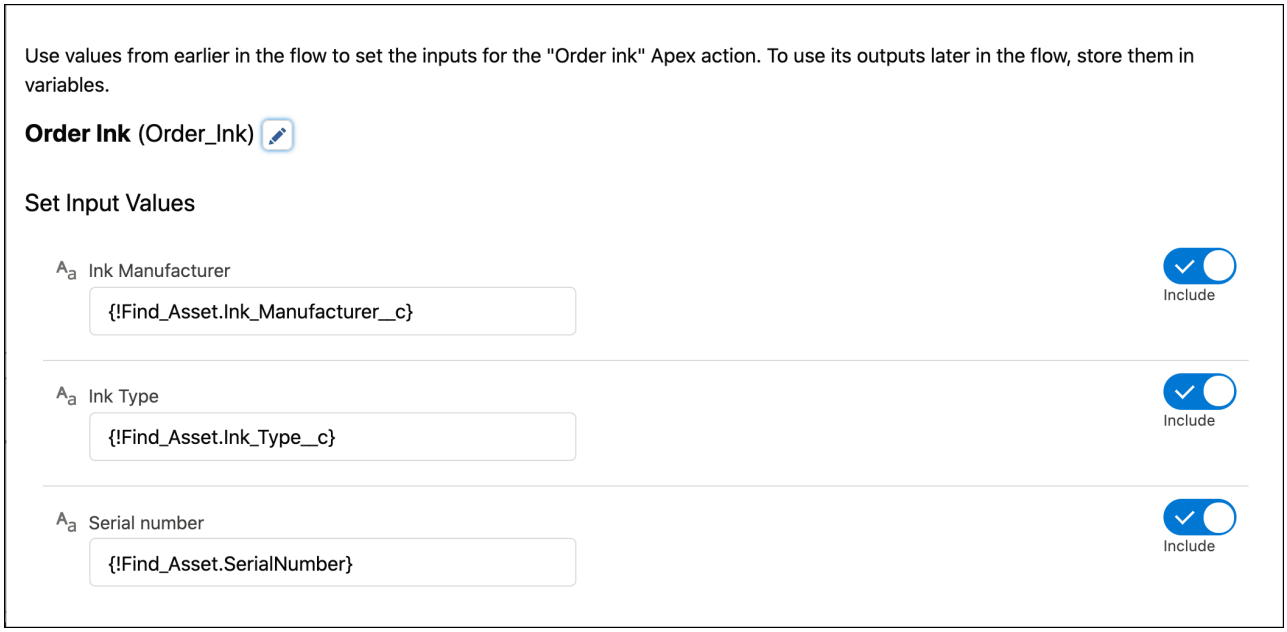


Order Ink or Paper

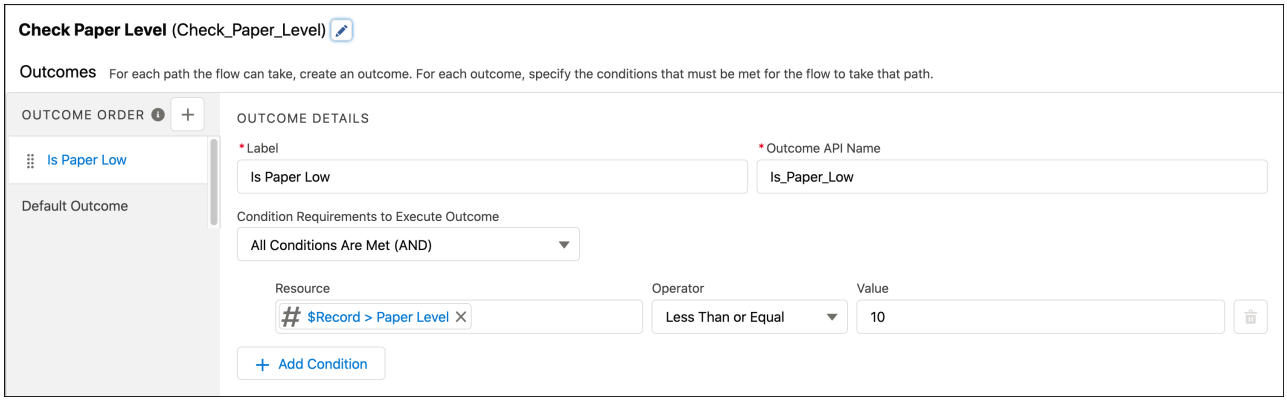
A Decision element evaluates whether the ink level is low. It checks whether the Ink_Level__c field value in the event message is equal to 'Low'.



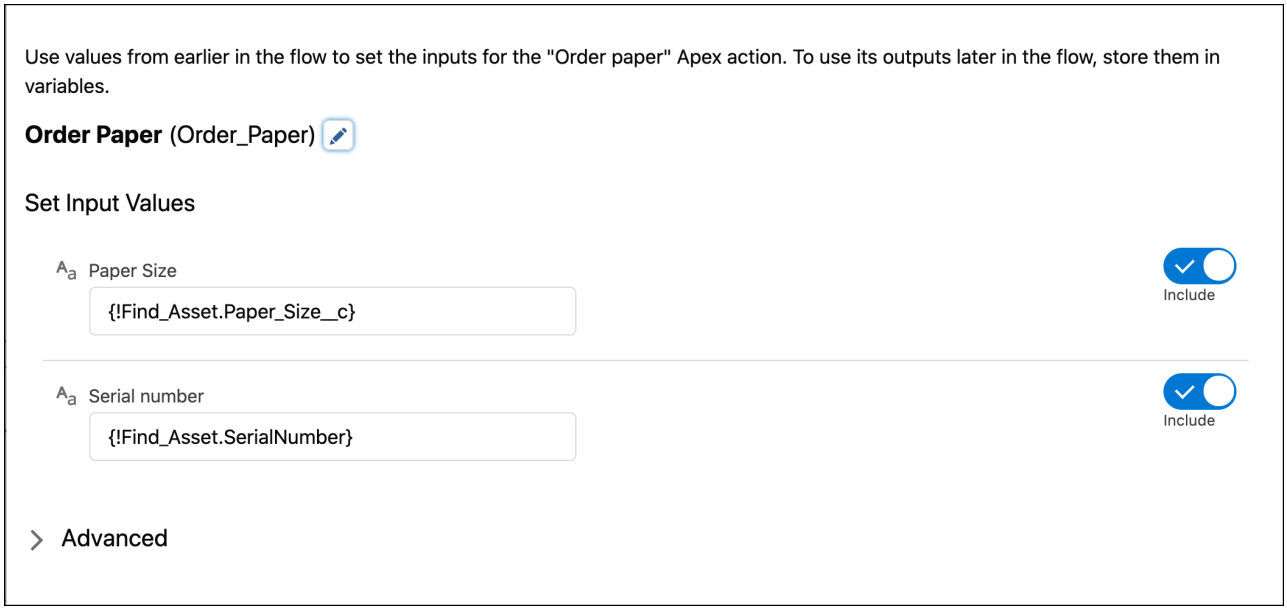
If the ink level is low, the flow calls an Apex action that orders ink. The Apex action calls an invocable method and passes information about the ink type and the printer serial number as invocable variables.



After the ink level is evaluated, another Decision element evaluates the paper level.



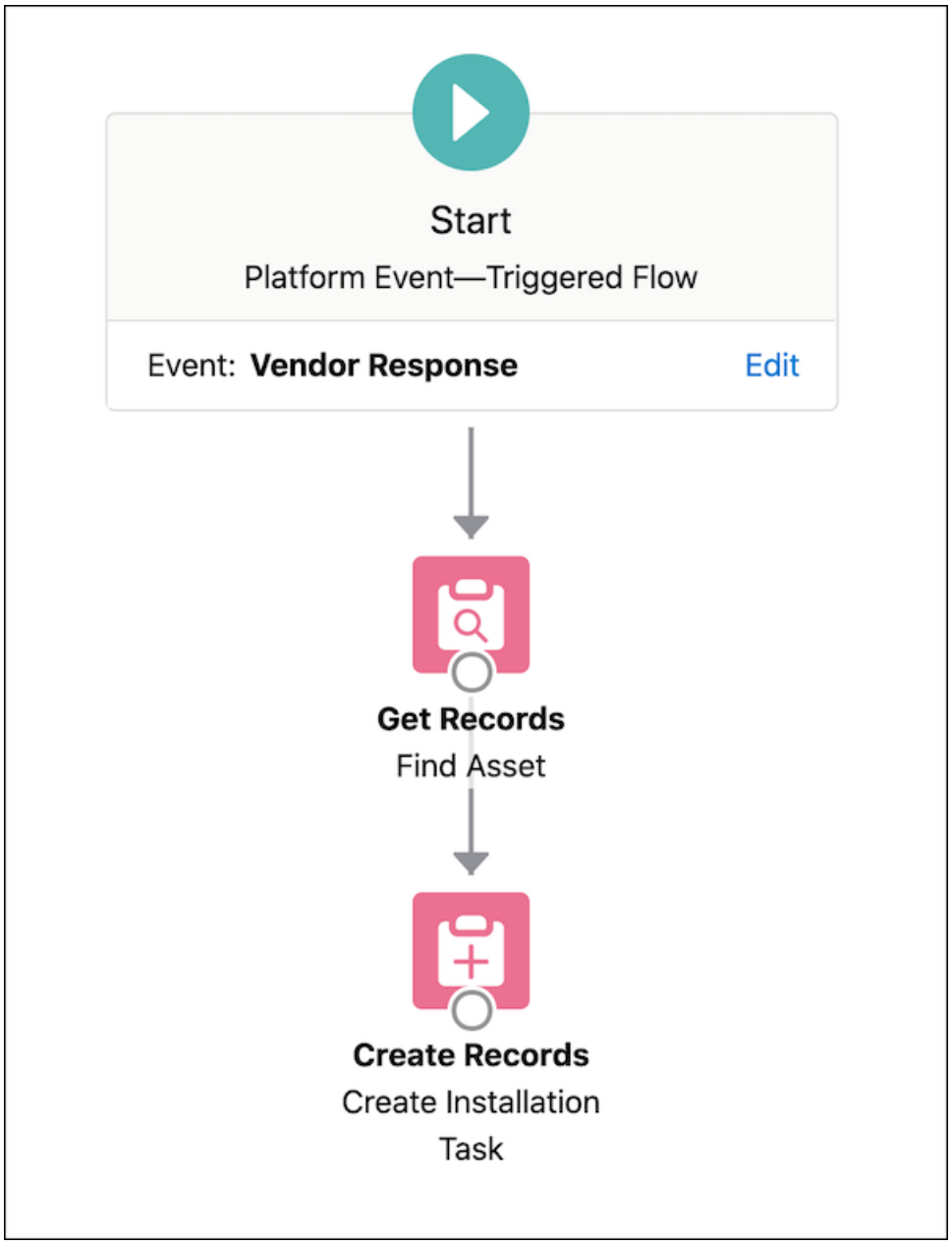
If the paper level is lower than 10%, the flow calls an Apex action to order paper. The Apex action calls an invocable method and passes the paper size and serial number as invocable variables.



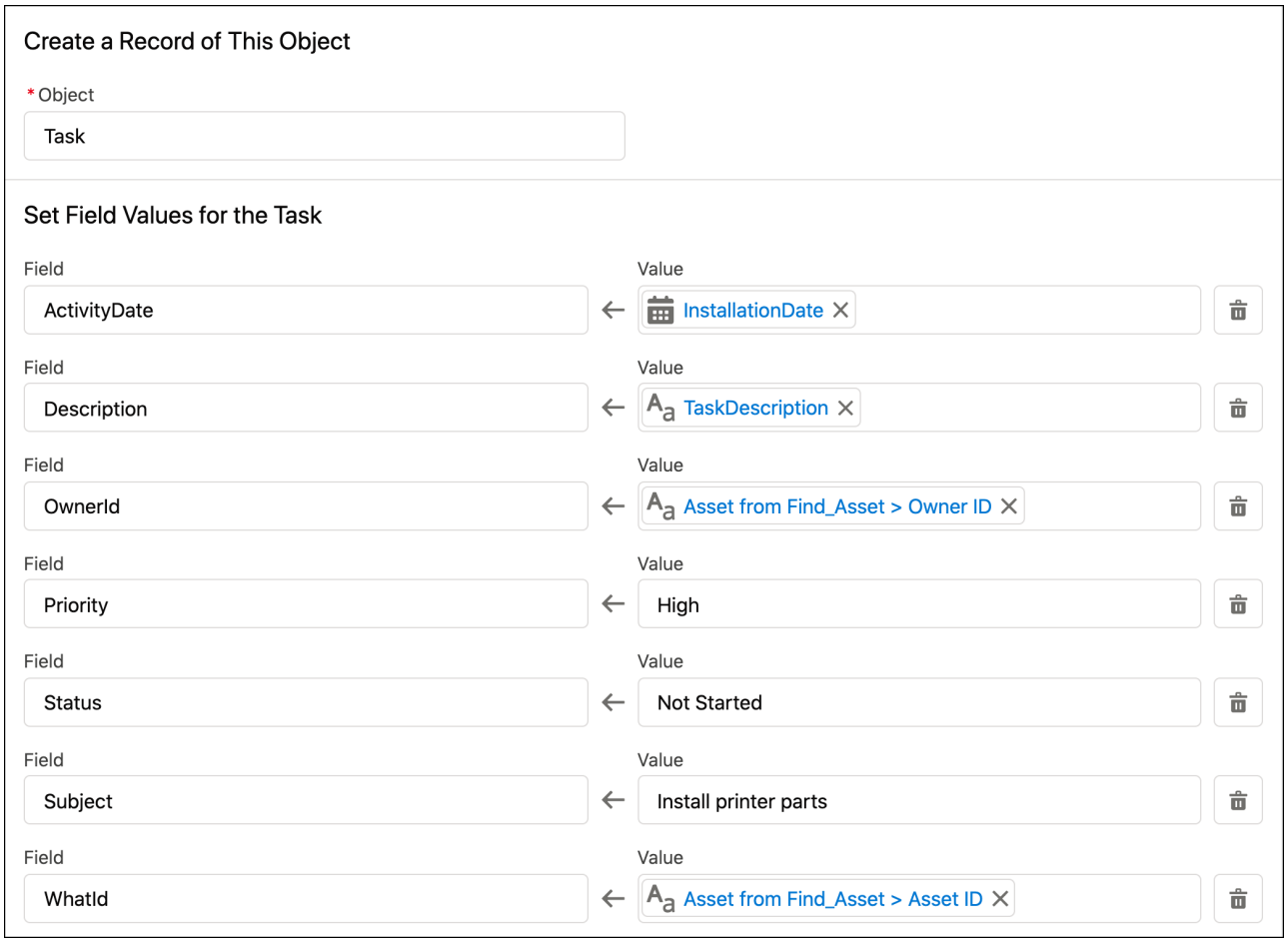
The implementation of Apex actions isn't covered in this example. For more information about invocable Apex actions, see InvocableMethod Annotation and InvocableVariable Annotation in the *Apex Developer Guide*. Typically, you call an external service to place an order. To do so from an Apex action, you use Apex callouts. For more information, see Invoking Callouts Using Apex in the *Apex Developer Guide*.

Flow: Automation for Vendor Response Events

The Install Printer Supplies flow is a platform event–triggered flow that subscribes to the Vendor Response platform event. When the vendor ships the printer part, they publish the Vendor Response platform event to notify their customer. This flow starts when it receives the Vendor Response event message. It creates a task for the asset owner to install the new printer part.

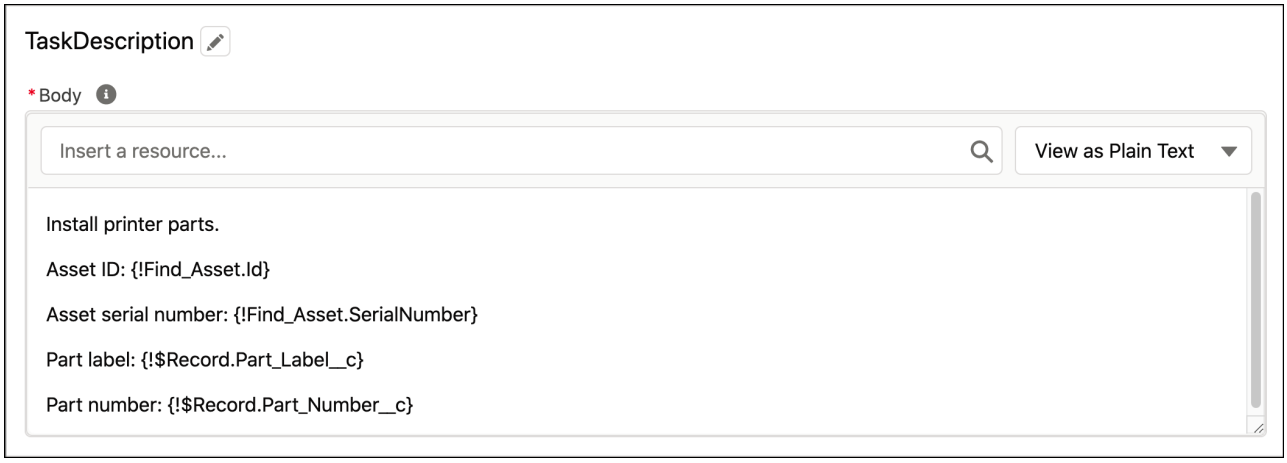


The Get Records element finds the related asset by matching the asset's serial number with that of the received event message. Next, the Create Records element creates the installation task for the part.



In this example, some task fields reference flow resources that are created separately. The InstallationDate is a formula resource and is defined as follows.

TaskDescription is a text template resource with the following body.



Java Client

The sample Java client uses Pub/Sub API to publish and subscribe to platform events. Pub/Sub API provides a single interface to publish and subscribe to event messages. Based on gRPC and HTTP/2, Pub/Sub API enables efficient delivery of binary event messages in the Apache Avro format.

Important: The sample Java client provides enough instructions and code snippets so that you can build your own client. The examples in the Java client are for learning purposes only. The examples aren't intended for production use and haven't undergone thorough functional and performance testing. You can use these examples as a starting point to build your own client.

Check out Java Quick Start for Publishing and Subscribing to Events in the Pub/Sub API Developer Guide.

Platform Event Samples

Check out a sample that covers common business scenarios and uses platform events along with other Lightning Platform features.

Sample App: The E-Bikes App and the Pub/Sub API Demo

E-Bikes is a fictitious electric bicycle manufacturer. E-Bikes manages its products and reseller orders with the E-Bikes app, which offers a rich user experience. Another app, the E-Bikes Manufacturing app, receives orders sent from the E-Bikes app. The E-Bikes Manufacturing app is a Node app that uses Pub/Sub API to subscribe to Order__ChangeEvent, the change event that is generated for orders, when a reseller order is placed in the E-Bikes app. After the manufacturer receives the change event and approves the order, the manufacturing app publishes a platform event, Manufacturing_Event__e, back to Salesforce.

The Pub/Sub API demo represents the E-Bikes Manufacturing app and is built using the Lightning Web Runtime. The demo is an add-on to the E-Bikes sample app. The E-Bikes sample app uses Lightning Web Components and integrates with Salesforce Experiences.

Install the E-Bikes app from the ebikes-lwc GitHub repository. After you install the E-Bikes app, install the Pub/Sub API demo from the ebikes-manufacturing GitHub repository.

Reference

The reference documentation for platform events covers an API object, Apex methods, limits, error codes, and standard platform events.

Check out this object in the *Object Reference for the Salesforce Platform*.

- EventBusSubscriber

Check out these Apex resources in the *Apex Reference Guide*.

- EventBus Class in the `System` namespace
- TriggerContext Class in the `EventBus` namespace

Check out these resources about limits, error codes, and standard platform events.

IN THIS SECTION:

Platform Event Allocations

Learn about the allocations available for platform event definitions, publishing and subscribing to platform events, and event delivery in Pub/Sub API clients, CometD clients, `empApi` Lightning components, and event relays.

Platform Event Error Status Codes

When publishing an event message results in an error, a status code is returned in the SaveResult.

Standard Platform Event Objects

Check out the standard platform events that Salesforce publishes.

SEE ALSO:

Salesforce Help: Configure the Process Trigger

Salesforce Help: Flow element: Pause

Platform Event Allocations

Learn about the allocations available for platform event definitions, publishing and subscribing to platform events, and event delivery in Pub/Sub API clients, CometD clients, `empApi` Lightning components, and event relays.

IN THIS SECTION:

Which Type of Platform Events Do Allocations Apply to?

Platform events can be custom events, which are platform events that you define, or standard events, which are the events that Salesforce defines, including Real-Time Event Monitoring events.

Common Platform Event Allocations

Common allocations include allocations for platform event definitions, concurrent CometD clients, and allocations for processes and flows. The common allocations apply to standard-volume and high-volume platform events.

Default Platform Event Allocations

If your org has no add-on licenses, default allocations apply for event publishing and delivery that can't be exceeded. The default allocations are enforced to ensure fair sharing of resources in the multitenant environment and to protect the service.

Platform Event Add-On License and Usage-Based Entitlement

To increase your event delivery allocation for Pub/Sub API, CometD, `empApi` Lightning components, and event relays, purchase an add-on for additional platform events. The add-on moves your event delivery usage to a monthly model and allows for spikes in usage. To purchase an add-on, contact your Salesforce Account Representative.

Monitor Event Usage Against Your Allocations

Check your event publishing and delivery usage and maximum allocation in Setup, or using REST API or Apex.

Monitor Hourly Event Delivery Usage with REST API

Get the hourly delivery usage by periodically retrieving the daily event delivery usage using REST API.

Monitor Event Usage with PlatformEventUsageMetric

Perform a SOQL query on PlatformEventUsageMetric to get visibility into your event usage and usage trends. With enhanced usage metrics, you can view separate and combined metrics for platform events and change data capture events. Break down usage metrics by event name, client ID, event type, usage type, and get usage data by granular time segments. PlatformEventUsageMetric data is available for CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays.

Standard-Volume Platform Event Allocations

These allocations are for standard-volume events defined in API version 44.0 and earlier.

SEE ALSO:

Considerations for Publishing and Subscribing to Platform Events with Apex and APIs

Apex Publish Callback Limits

Change Data Capture Developer Guide: Change Data Capture Allocations

Which Type of Platform Events Do Allocations Apply to?

Platform events can be custom events, which are platform events that you define, or standard events, which are the events that Salesforce defines, including Real-Time Event Monitoring events.

- The event publishing and delivery allocations apply to custom events.
- For standard events, including Real-Time Event Monitoring events, the event publishing allocation isn't enforced.
- For standard events, to find out whether the event delivery allocation applies, check the event reference documentation in Standard Platform Event Object List.
- For Real-Time Event Monitoring events, the event delivery allocation isn't enforced.
- When allocations aren't enforced, system protection limits apply.

Common Platform Event Allocations

Common allocations include allocations for platform event definitions, concurrent CometD clients, and allocations for processes and flows. The common allocations apply to standard-volume and high-volume platform events.

Description	Performance and Unlimited Editions	Enterprise Edition	Developer Edition	Professional Edition (with API Add-On)
Maximum number of platform event definitions that can be created in an org	100	50	5	5
Maximum number of concurrent CometD clients (subscribers) across all channels and for all event types	2,000	1,000	20	20
Maximum number of Process Builder processes and flows that can subscribe to a platform event	4,000	4,000	4,000	5
Maximum number of active Process Builder processes and flows that can subscribe to a platform event	2,000	2,000	2,000	5
Maximum number of custom channels that can be created for all events except Real-Time Event Monitoring events	100	100	100	100

Description	Performance and Unlimited Editions	Enterprise Edition	Developer Edition	Professional Edition (with API Add-On)
This allocation is separate from the one for custom change data capture channels.				
Maximum number of custom channels that can be created for Real-Time Event Monitoring events This allocation is separate from the one for custom change data capture channels.	3	3	3	3
Maximum number of distinct custom platform events that can be added to a channel as part of channel members If the same platform event is added to multiple channels, it's counted once toward the allocation.	50	50	5	5
Maximum number of Real-Time Event Monitoring events that can be added to a channel as part of channel members If the same event is added to multiple channels, it's counted once toward the allocation.	10	10	10	10
Maximum event message size that you can publish If your event object has hundreds of custom fields or many long text area fields, you can hit this limit. In this case, the publishing call gets an error.	1 MB	1 MB	1 MB	1 MB

Note:

- The concurrent client allocation applies to CometD and to all types of events: platform events, change events, PushTopic events, and generic events. It doesn't apply to non-CometD clients, such as Apex triggers, flows, and Process Builder processes. Flows and Process Builder processes apply only to platform events and not to change events. The `empApi` Lightning component uses CometD and consumes the concurrent client allocation like any other CometD client. Each logged-in user using `empApi` counts as one concurrent client. If the user has multiple browser tabs using `empApi`, the streaming connection is shared and is counted as one client for that user. A client that exceeds the concurrent client allocation receives an error and can't subscribe. When one of the clients disconnects and a connection is available, the new client can subscribe. For more information, see Streaming API Error Codes in the Streaming API Developer Guide.
- Platform events that originate from an installed managed package share the org's allocation for the maximum number of platform event definitions.

SEE ALSO:

Enterprise Messaging Platform Events

Default Platform Event Allocations

If your org has no add-on licenses, default allocations apply for event publishing and delivery that can't be exceeded. The default allocations are enforced to ensure fair sharing of resources in the multitenant environment and to protect the service.

- The event publishing allocation is how many event messages you can publish per hour using any method, including Apex, Pub/Sub API and other APIs, flows, and processes.
- The event delivery allocation is how many event messages can be delivered in a 24-hour period to Pub/Sub API and CometD subscribers, `empApi` Lightning components, and event relays. It excludes non-API subscribers, such as Apex triggers, flows, and Process Builder processes. Published event messages that are delivered to non-API subscribers, such as Apex triggers, flows, and Process Builder processes, don't count against the delivery allocation.
- The event delivery allocation is shared between high-volume platform events and Change Data Capture events.

Note: Even though Apex triggers, flows, and Process Builder processes don't count against the event delivery limit, their event processing rate depends on the subscriber processing time and volume of events received. A higher processing time and event volume means that it takes longer for the subscriber to reach the tip of the event stream.

Event Delivery Usage Combined for All Subscribers

The number of delivered events to clients is counted for each subscribed client, including event relays. If you have multiple client subscribers, your usage is added across all subscribers. For example, you have an Unlimited Edition org with a default allocation of 50,000 events in a 24-hour period. Within a few hours, 20,000 event messages are delivered to two subscribed clients. So you consumed 40,000 events and are still entitled to 10,000 events within the 24-hour period.

How Is Event Publishing and Delivery Usage Calculated?

The event hourly publishing and daily delivery limits are rolling limits. The hourly publishing usage is calculated for the number of publishes in the last hour. Similarly, the daily delivery usage is calculated for the number of delivered events in the last 24 hours. As time goes by, the usage is updated. The event publishing limit is checked when a new event is published. The event delivery limit is checked when a new event is received.

To learn more about how event usage is calculated against your event allocations, see Learn About Daily Rate Limits in the App Development Without Limits Trailhead module.

Default Allocations

Description	Performance and Unlimited Editions	Enterprise Edition and Professional Edition (with API Add-On)	Developer Edition
Event Delivery: maximum number of delivered event messages in the last 24 hours, shared by all clients. (Applies to CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays only.)	50,000	25,000	10,000
Event Delivery for Salesforce Order Management: maximum number of delivered event messages in the last 24 hours, shared by all clients. (Applies to CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays only.) This allocation is provided with the purchase of a Salesforce Order Management license.	100	100	100

Description	Performance and Unlimited Editions	Enterprise Edition and Professional Edition (with API Add-On)	Developer Edition
Event Publishing: maximum number of event messages published per hour. (Applies to all publishing methods, including Apex, Pub/Sub API and other APIs, flows, and Process Builder processes.)	250,000	250,000	50,000

How to Avoid Exceeding Event Allocations

Proactively monitor your event usage. For more information, see Monitor Event Usage Against Your Allocations and Monitor Event Usage with PlatformEventUsageMetric. When your event publishing usage gets close to the allocation, try these methods to reduce the consumption of delivered events.

- Use stream filtering to reduce the amount of events delivered to the subscriber and receive only relevant events. For more information, see Filter Your Stream of Platform Events with Custom Channels.
- Make sure you don't have unnecessary subscribers. Each event delivered to a subscriber counts against the event delivery allocation.

What to Do If You Exceed the Event Delivery Allocation

If you exceed the default event delivery allocation, an error is returned and the subscription is disconnected.

- The error you receive in a CometD client is: `403::Organization total events daily limit exceeded`. The error is returned in the Bayeux `/meta/connect` channel when a CometD subscriber first connects or in an existing subscriber connection. For more information, see Streaming API Error Codes in the Streaming API Developer Guide.
- The error code that you receive in a Pub/Sub API client is: `sfdc.platform.eventbus.grpc.subscription.limit.exceeded`. And the error message is: `You have exceeded the event delivery limit for your org`.

When the client reaches the event delivery allocation, perform one of these steps.

- Keep the subscriber disconnected for a temporary time. While the subscriber is disconnected, the event usage for the last 24 hours decreases after some time. The events received in Salesforce during the disconnected state are stored for the retention period of 72 hours. After usage decreases, resume the subscription from where it left off and receive events. You can retrieve stored event messages with Pub/Sub API and CometD using the Replay ID.

- If you reach the event delivery limit often and your event volume is high, consider purchasing an add-on to increase your event allocations by contacting your Salesforce Account Representative. The add-on that you purchase moves your event delivery usage to a monthly model and allows for spikes in usage.

SEE ALSO:

Platform Event Add-On License and Usage-Based Entitlement

Monitor Event Usage Against Your Allocations

Change Data Capture Developer Guide: Change Data Capture Allocations

Salesforce Developers Blog: How to Work Within Platform Events Delivery Limits

Pub/Sub API Developer Guide

Salesforce Help: Event Relay

Streaming API Developer Guide

Platform Event Add-On License and Usage-Based Entitlement

To increase your event delivery allocation for Pub/Sub API, CometD, `empApi` Lightning components, and event relays, purchase an add-on for additional platform events. The add-on moves your event delivery usage to a monthly model and allows for spikes in usage. To purchase an add-on, contact your Salesforce Account Representative.

Check out the benefits and facts about an add-on license.

- The add-on increases the 24-hour allocation of delivered event messages by 100,000 per day (3 million a month) as a usage-based entitlement.
- The add-on increases the hourly event publishing allocation by 25,000 events per hour.
- The add-on allows for spikes in usage. The daily delivery usage isn't as strictly enforced as the default allocation. You can can exceed your 24-hour event delivery allocation by a certain amount.
- The entitlement is reset every month after your contract start date.
- Entitlement usage is computed only for production orgs. It isn't available in sandbox or trial orgs. For more information, see Usage-based Entitlement Fields.
- Salesforce monitors event overages based on a calendar month, starting with your contract start date. If you exceed the monthly entitlement, Salesforce contacts you to discuss your event usage needs. The entitlement used for monitoring monthly event overages is the daily allocation multiplied by 30.

Table 1: Example: Entitlement with One High-Volume Platform Event Add-On License

Description	Performance and Unlimited Editions	Enterprise Edition and Professional Edition (with API Add-On)
Event Delivery: entitlement for delivered event messages, shared by all clients. (Applies to CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays only.) You can exceed this entitlement by a certain amount before receiving an error. Salesforce uses the monthly entitlement for event overage monitoring. The monthly entitlement is returned in the `limits` REST API resource.	Last 24 hours: 150,000 (50 K included with org license + 100 K from add-on license) Monthly entitlement: 4.5 million (1.5 million included with org license + 3 million from add-on license)	Last 24 hours: 125,000 (25 K included with org license + 100 K from add-on license) Monthly entitlement: 3.75 million (0.75 million included with org license + 3 million from add-on license)

Description	Performance and Unlimited Editions	Enterprise Edition and Professional Edition (with API Add-On)
Event Publishing: maximum number of event messages published per hour. (Applies to all publishing methods, including Apex, Pub/Sub API and other APIs, flows, and Process Builder processes.)	275,000 (250 K included with org license + 25 K from add-on license)	275,000 (250 K included with org license + 25 K from add-on license)

Monitor Event Usage Against Your Allocations

Check your event publishing and delivery usage and maximum allocation in Setup, or using REST API or Apex.

Check your event publishing and delivery usage in the user interface. From Setup, in the Quick Find box, enter *Platform Events*, and then select **Platform Events**. The usage is shown in the Event Allocations section.

If your org purchased the add-on for additional platform events, the monthly event delivery usage is also displayed in the Event Allocations section. This usage corresponds to the `MonthlyPlatformEvents` REST limits value.

Learn about other ways to check event usage with REST API, Apex, and in the Company Information page.

Allocation	Default Allocations	Add-On License
Event Delivery: number of delivered event messages to CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays	If your org hasn't purchased the add-on, check your usage in one of these ways. • Daily event delivery usage in the last 24 hours using REST API: check the `DailyDeliveredPlatformEvents` value with the REST API `limits` resource. • Daily event delivery usage in the last 24 hours using Apex: use the `System.OrgLimit` class and check the `DailyDeliveredPlatformEvents` value. The daily event delivery usage is updated within a few minutes after event delivery.	If your org has purchased the add-on, check your usage in one of these ways. • Monthly event delivery usage: From Setup, in the Quick Find box, enter *Platform Events*, and then select **Platform Events**. The monthly event delivery usage is displayed in the Event Allocations section. In the REST API `limits` resource, this value corresponds to `MonthlyPlatformEvents` in API version 47.0 and earlier. This value in the UI and API is updated within a few minutes after event delivery. • Usage-based entitlement: From Setup, in the Quick Find box, enter *Company Information*, and then select **Company Information**. The usage is shown under the Usage-based Entitlements related list. In the REST API limits resource, this value corresponds

Allocation	Default Allocations	Add-On License
		to `MonthlyPlatformEventsUsageEntitlement` in API version 48.0 and later. This value in the UI and API is updated once a day.
Event Publishing: number of event messages published per hour	With the REST API `limits` resource: usage information is returned in `HourlyPublishedPlatformEvents`.	With the REST API `limits` resource: usage information is returned in `HourlyPublishedPlatformEvents`.

SEE ALSO:

REST API Developer Guide: Limits

REST API Developer Guide: List Org Limits

Apex Reference Guide: OrgLimit Class

Monitor Hourly Event Delivery Usage with REST API

Get the hourly delivery usage by periodically retrieving the daily event delivery usage using REST API.

To monitor your org's high-volume platform event and change event delivery hourly usage, make a REST API call to the limits resource every hour. The difference between the results obtained in the last 2 hours shows how many events were delivered in the last hour.

For example, you make a call at 12:00 PM and see that you have 40,000 events remaining. Then you run the same call at 1:00 PM and see that you have 38,500 events remaining. The returned responses indicate that 1,500 events were delivered to your API subscribers between 12:00 PM and 1:00 PM.

These results are examples of the responses that a GET request to the `/services/data/v61.0/limits` URI returns.

```
First call result:
{
...
  "DailyDeliveredPlatformEvents" : {
    "Max" : 50000,
    "Remaining" : 40000
  },

...
}

Second call result:
{
...
  "DailyDeliveredPlatformEvents" : {
    "Max" : 50000,
    "Remaining" : 38500
  },
```

```
...
}
```

SEE ALSO:

REST API Developer Guide: Limits

REST API Developer Guide: List Org Limits

Monitor Event Usage with PlatformEventUsageMetric

Perform a SOQL query on PlatformEventUsageMetric to get visibility into your event usage and usage trends. With enhanced usage metrics, you can view separate and combined metrics for platform events and change data capture events. Break down usage metrics by event name, client ID, event type, usage type, and get usage data by granular time segments. PlatformEventUsageMetric data is available for CometD and Pub/Sub API clients, `empApi` Lightning components, and event relays.

For more information, see Enhanced Usage Metrics.

Standard-Volume Platform Event Allocations

These allocations are for standard-volume events defined in API version 44.0 and earlier.

Important: You can no longer define new standard-volume custom platform events. New platform events are high volume by default. Standard-volume custom platform events will be retired in Winter '25. To migrate existing standard-volume events, see Migrate Standard-Volume Platform Events to High-Volume Platform Events Before Retirement.

Description	Performance and Unlimited Editions	Enterprise Edition	Developer Edition and Professional Edition (with API Add-On)
Event Delivery: maximum number of delivered event messages in the last 24 hours, shared by all CometD clients[1]	50,000	25,000	10,000
Event Publishing: maximum number of event messages published per hour	100,000	100,000	1,000

If you exceed the event delivery allocation, you receive this error: `403::Organization total events daily limit exceeded`. The error is returned in the Bayeux `/meta/connect` channel when a CometD subscriber first connects or in an existing subscriber connection. For more information, see Streaming API Error Codes in the Streaming API Developer Guide. Standard-volume event messages that are generated after exceeding the allocation are stored in the event bus. You can retrieve stored standard-volume event messages as long as they're within the 24-hour retention window.

To monitor your standard-volume event delivery usage, use the `limits` REST API resource, and inspect the `DailyStandardVolumePlatformEvents` value. And to monitor the publishing usage, inspect the `HourlyPublishedStandardVolumePlatformEvents` value. For more information, see List Organization Limits in the *REST API Developer Guide*.

[1]To request a higher number of standard-volume events delivered to CometD clients, contact Salesforce to purchase an add-on license. The add-on license increases your daily limit of delivered events by 100,000 more events. For example, for Unlimited Edition, the add-on license increases the daily limit of delivered events from 50,000 to 150,000 events. You can purchase multiple add-ons to meet your

event requirements for CometD clients. To avoid deployment problems and degradation in service, we recommend that the number of events delivered to CometD clients not exceed 5 million per day. If you require more external events, contact your Salesforce representative to understand how the product can scale to meet your needs.

Platform Event Error Status Codes

When publishing an event message results in an error, a status code is returned in the SaveResult.

Synchronous Errors

The following error status codes are returned immediately in the publish call result.

`LIMIT_EXCEEDED`
: The number of published platform event messages exceeded the hourly publishing limit or the test limit for event messages published from an Apex test context.

`PLATFORM_EVENT_PUBLISHING_UNAVAILABLE`
: Publishing platform event messages failed because a service was temporarily unavailable. Try again later.

`PLATFORM_EVENT_ENCRYPTION_ERROR`
: The platform event messages could not be published due to a problem with encryption. A misconfiguration in your Salesforce org or a general encryption service error can cause this problem.

In Apex, the status code is returned in the `Database.SaveResult` in the `Database.Error` object. In SOAP API, the status code is returned in the `SaveResult` object. In REST API, the status code is returned in the `errors` field in the JSON message.

Standard Platform Event Objects

Check out the standard platform events that Salesforce publishes.

IN THIS SECTION:

Change Data Capture Events
: Salesforce Change Data Capture publishes change events, which represent changes to Salesforce records. Changes include record creation, updates to an existing record, deletion of a record, and undeletion of a record. Change Data Capture events are available since API version 44.0.

Standard Platform Event Object List
: Salesforce publishes standard platform events in response to an action that occurred in the org or to report errors. For example, LoginEventStream monitors user login activity and BatchApexErrorEvent reports errors encountered in batch Apex jobs. You can subscribe to a standard platform event using the subscription mechanism that the event supports.

Change Data Capture Events

Salesforce Change Data Capture publishes change events, which represent changes to Salesforce records. Changes include record creation, updates to an existing record, deletion of a record, and undeletion of a record. Change Data Capture events are available since API version 44.0.

Change Event Name

Change events are available for all custom objects and a subset of standard objects. The name of a change event is based on the name of the corresponding object for which it captures the changes. For a list of supported standard objects, see *StandardObjectName*ChangeEvent in the *Object Reference for Salesforce and Lightning Platform*.

Standard Object Change Event Name

```
<Standard_Object_Name>ChangeEvent
```

Example: `AccountChangeEvent`

Custom Object Change Event Name

```
<Custom_Object_Name>__ChangeEvent
```

Example: `Employee__ChangeEvent`

Subscription Channels

Subscription channels for change events depend on the name of the change event you want to receive messages for. Also, a generic channel is provided to receive all messages.

Channel for All Change Events

To receive event messages for all objects selected for Change Data Capture, use this channel:

```
/data/ChangeEvents
```

Standard Object Channel

To receive event messages for changes in a standard object, use this channel:

```
/data/<Standard_Object_Name>ChangeEvent
```

Example: AccountChangeEvent

Custom Object Channel

To receive event messages for changes in a custom object, use this channel:

```
/data/<Custom_Object_Name>__ChangeEvent
```

Example: Employee__ChangeEvent

Change Event Fields

The record fields in the change event correspond to the fields on the associated Salesforce object or entity that triggered the change. Only new or updated fields are included in the event message with a populated value.

For example, the fields that can be sent in a change event for the Account object are the Account fields. To look up the fields of a standard object, see Object Reference for Salesforce and Lightning Platform.

Each change event also contains header fields. The header fields are included inside the `ChangeEventHeader` field. They contain information about the event, such as whether the change was an update or delete and the name of the entity, like Account, among other things.

The following example shows a change event message for a new account received in a Pub/Sub API client.

```
{
  "ChangeEventHeader": {
    "entityName": "Account",
```

```
    "recordIds": [
      "0015f00002J9YYEAA3"
    ],
    "changeType": "CREATE",
    "changeOrigin": "com/salesforce/api/soap/60.0;client=SfdcInternalAPI/",
    "transactionKey": "0001ade9-3f74-0b99-dbc4-42e73424b774",
    "sequenceNumber": 1,
    "commitTimestamp": 1712693965000,
    "commitNumber": 1082985383811,
    "commitUser": "0055f000005mc66AAA",
    "nulledFields": [],
    "diffFields": [],
    "changedFields": []
  },
  "Name": "Acme",
  "Type": null,
  "ParentId": null,
  "BillingAddress": null,
  "ShippingAddress": null,
  "Phone": null,
  "Fax": null,
  "AccountNumber": null,
  "Website": null,
  "Sic": null,
  "Industry": null,
  "AnnualRevenue": null,
  "NumberOfEmployees": null,
  "Ownership": null,
  "TickerSymbol": null,
  "Description": "Sample account record.",
  "Rating": null,
  "Site": null,
  "OwnerId": "0055f000005mc66AAA",
  "CreatedDate": 1712693965000,
  "CreatedById": "0055f000005mc66AAA",
  "LastModifiedDate": 1712693965000,
  "LastModifiedById": "0055f000005mc66AAA",
  "Jigsaw": null,
  "JigsawCompanyId": null,
  "CleanStatus": "Pending",
  "AccountSource": null,
  "DunsNumber": null,
  "Tradestyle": null,
  "NaicsCode": null,
  "NaicsDesc": null,
  "YearStarted": null,
  "SicDesc": null,
  "DandbCompanyId": null,
  "OperatingHoursId": null,
  "CustomerPriority__c": null,
  "SLA__c": null,
  "Active__c": null,
  "NumberofLocations__c": null,
  "UpsellOpportunity__c": null,
```

```
  "SLASerialNumber__c": null,
  "SLAExpirationDate__c": null
}
```

Resources

For more information about Change Data Capture, see Change Data Capture Developer Guide and Pub/Sub API Developer Guide.

Standard Platform Event Object List

Salesforce publishes standard platform events in response to an action that occurred in the org or to report errors. For example, LoginEventStream monitors user login activity and BatchApexErrorEvent reports errors encountered in batch Apex jobs. You can subscribe to a standard platform event using the subscription mechanism that the event supports.

IN THIS SECTION:

AIPredictionEvent

Notifies subscribers when an Einstein feature, such as Prediction Builder or Case Classification, has written prediction results back to a target object and AI prediction field. This object is available in API version 46.0 and later.

AIUpdateRecordEvent

Notifies subscribers when Einstein Case Classification has generated a case field value prediction and potentially updated a case record. This object is available in API version 47.0 and later.

AppointmentSchedulingEvent

Notifies subscribers when an appointment schedule is added, updated, or deleted. This object is available in API version 50.0 and later.

AssetCancelInitiatedEvent

Notifies subscribers when the process started by the `/asset-management/assets/collection/actions/initiate-cancellation` process is complete. If the process is successful, use this event to learn about the cancellation order that was created. If the process isn't successful, use the RevenueTransactionErrorLog records to learn about the errors and how to fix them. This object is available in API version 55.0 and later.

AssetAmendInitiatedEvent

Notifies subscribers when the process started by the `/asset-management/assets/collection/actions/initiate-amend-quantity` REST request is complete. If the process is successful, use this event to learn about the amendment order that was created. If the process isn't successful, use the RevenueTransactionErrorLog records to learn about the errors and how to fix them. This object is available in API version 56.0 and later.

AssetRenewInitiatedEvent

Notifies subscribers when the process started by the `/asset-management/assets/collection/actions/initiate-renew` REST request is complete. If the process is successful, use this event to learn about the renewal order that was created. If the process isn't successful, use the RevenueTransactionErrorLog records to learn about the errors and how to fix them. This object is available in API version 55.0 and later.

AssetTokenEvent

Notifies subscribers of asset token issuance and registration of a connected device as an Asset. This object is available in API version 39.0 and later.

BatchApexErrorEvent

Notifies subscribers of errors and exceptions that occur during the execution of a batch Apex class. This object is available in API version 44.0 and later.

BillingScheduleCreatedEvent

Notifies subscribers when the `/actions/standardCreateBillingScheduleFromOrderItem` request is complete. This object is available in API version 55.0 and later.

CommerceDiagnosticEvent

Tracks checkout, pricing, search, and other activity within your Commerce implementation to monitor events and diagnose issues. This object is available in API version 49.0 and later.

ConsentEvent

Notifies subscribers of changes to consent fields or contact information on all core objects. This object is available in API version 50.0 and later.

ConsentUnsubscribeAllEvent

Notifies subscribers when a user unsubscribes from all communications on a preference form created in Preference Manager. This object is available in API version 60.0 and later.

CreateAssetOrderEvent

Notifies subscribers that the process started by the `/actions/standard/createOrUpdateAssetFromOrder` request is complete. If the process is successful, use this event to learn about the new assets. If the request isn't successful, use this event to learn about the errors and how to fix them. This object is available in API version 55.0 and later.

CreditInvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/invoices/{invoiceId}/actions/credit` request is complete. This object is available in API version 55.0 and later.

CreditMemoProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/credit-memos` request is complete. This object is available in API version 55.0 and later.

DataObjectDataChgEvent

Notifies subscribers of an action within Data Cloud. This object is available in API version 53.0 and later.

DataObjectMetadataChgEvent

Notifies subscribers of a metadata change within Data Cloud for these objects: Data Lake, Data Model, and Calculated Insight. This object is available in API version 53.0 and later.

DatasetExportEvent

Notifies subscribers on the export of an Analytics dataset. This object is available in API version 41.0 and later.

DiscoveryPredictionEvent

Notifies subscribers when Einstein Discovery has written prediction history results. This object is available in API version 57.0 and later.

ExtlRecShrEvent

Reserved for future use. This object is available in API version 61.0 and later.

FirstBillPaymentSetupEvent

Notifies subscribers when a first bill payment is set up. This object is available in API version 60.0 and later.

FlowExecutionErrorEvent

Notifies subscribers of errors related to screen flow executions. This object is available in API version 47.0 and later.

FlowOrchestrationEvent

Notifies subscribers that a paused instance of an orchestration is ready to be resumed. This object is available in API version 53.0 and later.

FOStatusChangedEvent

Notifies subscribers of changes to the status of a fulfillment order record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 48.0 and later.

FulfillOrdItemQtyChgEvent

Notifies subscribers of changes to the quantity of a fulfillment order line item record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 53.0 and later.

InvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/billing/invoices` request is complete. The process groups billing schedules by grouping keys and creates one invoice per grouping key. `InvoiceProcessedEvent` is a top-level object that contains a list of `InvoiceProcessedDetailEvents`, where each detail event represents an attempt to create one invoice. This object is available in API version 55.0 and later.

NegInvcLineProcessedEvent

Notifies subscribers when a negative invoice line is converted to a credit memo. This object is available in API version 56.0 and later.

OmniTrackingEvent

Notifies subscribers about a user interaction with a FlexCard or OmniScript that's tracked for OmniAnalytics. This object is available in API version 60.0 and later.

OrderStatusChangedEvent

Notifies subscribers of changes to the status of an order record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 51.0 and later.

OrderSummaryCreatedEvent

Notifies subscribers of the creation of an order summary record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 48.0 and later.

OrderSumStatusChangedEvent

Notifies subscribers of changes to the status of an order summary record. Use this event to trigger subscribers such as flows in your order workflow. This object is available in API version 48.0 and later.

PaymentCreationEvent

Notifies subscribers when the process started by the `/actions/standard/paymentSale` request is complete. This object is available in API version 55.0 and later.

PendingOrdSumProcEvent

Notifies subscribers that a PendingOrderSummary record was processed. If the process succeeded, an OrderSummary was created and the PendingOrderSummary can be deleted. Use this event to trigger subscribers such as flows in your order workflow. This object is available in API version 56.0 and later.

PlatformStatusAlertEvent

Notifies subscribers of alerts that occur during the processing of a user request or service job execution. This object is available in API version 45.0 and later.

ProcessExceptionEvent

Notifies subscribers of errors that occur during payment processing (capture, apply, and refund) on an order summary. Use this event to trigger subscribers such as flows in your order workflow. This object is available in API version 50.0 and later.

QuoteSaveEvent

Notifies subscribers that the process started by the `/actions/standard/quotesaveevent` request is complete. If the process is successful, use this event to learn about the updated quote. If the request isn't successful, use this event to learn about the errors and how to fix them. This object is available in API version 58.0 and later.

QuoteToOrderCompletedEvent

Notifies subscribers when the `/actions/standard/createOrderFromQuote` REST request is complete. If the request is successful, use this event to learn about the Order record. If the request isn't successful, use this event to learn about the errors associated with the request. This object is available in API version 56.0 and later.

RealtimeAlertEvent

Notifies subscribers of Amazon CloudWatch alarm events from your Service Cloud Voice Amazon Connect instance. This object is available in API version 54.0 and later.

RemoteKeyCalloutEvent

Notifies subscribers of callouts that fetch encrypted key material from a customer endpoint. This object is available in API versions 45.0 and later.

ServiceAppointmentEvent

Notifies subscribers of the service appointment details that are generated from the event platform. This object is available in API version 59.0 and later.

VoidInvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/invoices/{invoiceId}/actions/void` request is complete. The request attempts to void an invoice by crediting an invoice and changing its status to Voided, which prevents further changes. This object is available in API version 55.0 and later. This object is available in API version 55.0 and later.

WebStoreUserCreatedEvent

Notifies subscribers of the creation of a new user for a WebStore. This object is available in API version 59.0 and later.

Real-Time Event Monitoring Objects

Check out the standard platform event and object pairs for Real-Time Event Monitoring. For most platform events used in Real-Time Event Monitoring, corresponding objects store the event data. For more information, see Real-Time Event Monitoring in Salesforce Help.

AIPredictionEvent

Notifies subscribers when an Einstein feature, such as Prediction Builder or Case Classification, has written prediction results back to a target object and AI prediction field. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔

Subscriber	Supported?
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/AIPredictionEvent`

Special Access Rules

Users with Customize Application permission have read access.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`Confidence`	**Type** double **Properties** Nillable **Description** Relative confidence strength of the generated prediction result. Higher values (near 1.0) indicate stronger confidence.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`FieldName`	**Type** string **Properties** Nillable **Description** API name of the AI prediction field that prediction results were written back to. An AI prediction field is a custom field created for storing and displaying the prediction scores on records. The name is specified in `ObjectName.FieldName` format, for example,

Field	Details
	`Lead.predicted_score__c`. For Case Classification prediction results, this field can be null.
`HasError`	**Type** boolean **Properties** Defaulted on create **Description** `true` if there was an error while gathering information to create an event message, `false` otherwise.
`InsightId`	**Type** string **Properties** Nillable **Description** The unique ID of the created AIRecordInsight record that generated the event message.
`PredictionEntityId`	**Type** string **Properties** Nillable **Description** The unique ID of the created AIInsightValue record associated with the AIRecordInsight that generated the event message.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`TargetId`	**Type** string **Properties** Nillable **Description** The unique ID of the record Einstein is writing prediction results to.

Usage

When Einstein writes prediction results back to AI prediction fields, record save custom logic, such as Apex triggers, workflow rules, and assignment rules, aren't run for efficiency reasons. To add custom logic based on Einstein prediction results, subscribe to AIPredictionEvent for notifications of prediction result updates. Every time prediction results are written back to a Salesforce record, an AIPredictionEvent event is created and AIPredictionEvent subscribers are notified.

SEE ALSO:

Object Reference for Salesforce and Lightning Platform : AIRecordInsight

AIUpdateRecordEvent

Notifies subscribers when Einstein Case Classification has generated a case field value prediction and potentially updated a case record. This object is available in API version 47.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/AIUpdateRecordEvent`

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`ErrorCode`	**Type** picklist **Properties** Nillable, Restricted picklist

Field	Details
	Description Indicates whether an error occurred in the automatic case update, and describes the nature of the error. Values are: • `none`—No error occurred. • `entity_locked`—The case is locked for editing by an approval process. • `no_access`—The selected Einstein user or automatic process user doesn't have permission to make the update. For example, the user needs permission to update cases or the case field in question, or needs sharing-based access to the case. • `validation_rule`—The update violates a case validation rule. • `other`—A different error occurred. Available in API version 50.0 and later.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** Further describes an error that occurred in the automatic case update.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`IsUpdated`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the related case (`RecordId`) was updated by Einstein Case Classification. If a case value prediction falls below the required confidence level selected in the predictive model, the case is not updated (`false`). If the prediction meets the confidence level requirement, the case field is updated and the case is saved (`true`). It is only updated if at least one field has a confidence threshold defined for the field's Automate Value. Available in API version 49.0 and later.
`RecordId`	**Type** string

Field	Details
	Properties None **Description** The record in which the prediction results are written.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`UpdatedFields`	**Type** complexvalue **Properties** Nillable **Description** Indicates which record fields, if any, were updated in the event. Available in API version 49.0 and later.

Usage

When Einstein Case Classification generates a case field value prediction, an AIUpdateRecordEvent event message is generated on case create whether or not Einstein updates the case, and if at least one of the prediction fields has a confidence threshold set in the Automate Value setting. A prediction will not result in a case update if its confidence level falls below the confidence threshold defined for the field's Automate Value setting. Subscribe to AIUpdateRecordEvent to be notified of such changes and to rerun case routing logic.

If all fields have a prediction that meets the confidence threshold criteria and an unexpected error prevents recommendations from being auto-applied, an AIUpdateRecordEvent is published with a corresponding ErrorCode and Error Message.

If a case doesn't match the data segment filters for any of the apps, we don't score and auto-apply recommendations or publish any events.

There are additional considerations to auto-apply recommendations with dependent picklists. Learn More

AppointmentSchedulingEvent

Notifies subscribers when an appointment schedule is added, updated, or deleted. This object is available in API version 50.0 and later.

Supported Calls

`describeSObjects()`

Special Access Rules

AppointmentSchedulingEvent is available as part of Salesforce Scheduler.

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/AppointmentSchedulingEvent`

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`AssignedResourceFields`	**Type** AsgnRsrcApptSchdEvent[] **Properties** Nillable **Description** The assigned resources associated with the appointment.
`ChangeType`	**Type** string **Properties** Nillable **Description** The operation that caused the change. For example: CREATE, UPDATE, DELETE.
`CorrelationId`	**Type** string **Properties** Nillable

Field	Details
	Description The universally unique identifier (UUID) that correlates the appointment with the platform event.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`ServiceAppointmentFields`	**Type** SvcApptSchdEvent[] **Properties** Nillable **Description** The service appointments associated with the appointment.

Example

This example event message is for a new appointment with two assigned resources.

```
{
  "schema": "Zog7FKcPWV9DeEIEVHsoug",
  "payload": {
    "CreatedById": "005xx000001X7dlAAC",
    "ChangeType": "CREATE",
    "ServiceAppointmentFields": {
      "ParentRecordId": "001RM000003rwkfYAA",
      "ContactId": "003RM000006EpajYAC",
      "Status": "None",
      "AdditionalInformation": "Sample additional information",
      "ServiceTerritoryId": "0Hhxx0000004mu4",
      "Comments": "Sample comment",
```

```
      "Email": "abc@example.com",
      "Address": "1 Market Street San Francisco CA 94105 United States",
      "WorkTypeId": "08qxx0000004C92",
      "WorkTypeBlockTimeBeforeAppointment": 30,
      "WorkTypeBlockTimeAfterAppointment": 1,
      "WorkTypeBlockTimeBeforeUnit": "minutes",
      "WorkTypeBlockTimeAfterUnit": "hours",
      "ServiceAppointmentId": "08pxx0000005Ip6",
      "ScheduledEndTime": "2020-02-28T00:45:00.000Z",
      "Subject": "Apply for Privileged Customer Card",
      "AppointmentType": "null",
      "StatusCategory": "None",
      "DurationInMinutes": 60,
      "Phone": "4155551212",
      "ScheduledStartTime": "2020-02-27T23:45:00.000Z"
    },
    "AssignedResourceFields": [
      {
        "IsPrimaryResource": true,
        "ServiceResourceUserName": "Rachel Adams",
        "ServiceResourceUserId": "005xx000001X7dl",
        "AssignedResourceId": "03rxx0000004gLc",
        "ServiceResourceId": "0Hnxx0000004C92",
        "ServiceResourceUserEmail": "ra@example.com",
        "IsRequiredResource": true
      },
      {
        "IsPrimaryResource": false,
        "ServiceResourceUserName": "Andrew Collins",
        "ServiceResourceUserId": "005xx000001XPNl",
        "AssignedResourceId": "03rxx0000004gNE",
        "ServiceResourceId": "0Hnxx0000006z8q",
        "ServiceResourceUserEmail": "ac@example.com",
        "IsRequiredResource": false
      }
    ],
    "CreatedDate": "2020-02-25T01:57:39.936Z",
    "CorrelationId": "d7c0bbGiUObLF6BD3NaG"
  },
  "event": {
    "replayId": 3
  }
}
```


IN THIS SECTION:

AsgnRsrcApptSchdEvent

Represents the assigned resources that are part of various platform events. This object is included in a streamed notification received on the channels for the parent platform events. You can't subscribe to the AsgnRsrcApptSchdEvent channel directly. This object is available in API version 50.0 and later.

SvcApptSchdEvent

Represents the service appointment event. This object is included in a streamed notification received on the channels for the parent platform events. You can't subscribe to the SvcApptSchdEvent channel directly. This object is available in API version 50.0 and later.

AsgnRsrcApptSchdEvent

Represents the assigned resources that are part of various platform events. This object is included in a streamed notification received on the channels for the parent platform events. You can't subscribe to the AsgnRsrcApptSchdEvent channel directly. This object is available in API version 50.0 and later.

Supported Calls

`describeSObjects()`

Parent Platform Events

- AppointmentSchedulingEvent
- ServiceAppointmentEvent

Fields

Field	Details
`AssignedResourceId`	**Type** string **Properties** Nillable **Description** ID of the assigned resource.
`ChangedFields`	**Type** complexvalue **Properties** Nillable **Description** A list of fields that changed.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`IsPrimaryResource`	**Type** boolean **Properties** Defaulted on create

Field	Details
	Description Indicates whether the resource is primary.
`IsRequiredResource`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the resource is required.
`ServiceResourceId`	**Type** string **Properties** Nillable **Description** ID of the service resource assigned to the event.
`ServiceResourceUserEmail`	**Type** string **Properties** Nillable **Description** Email of the service resource user assigned to the event.
`ServiceResourceUserId`	**Type** string **Properties** Nillable **Description** ID of the user record associated with the service resource assigned to the event.
`ServiceResourceUserName`	**Type** string **Properties** Nillable **Description** Username as per the user record associated with the service resource assigned to the event.

Example

This example shows the assigned resources associated with the event.

```
{
  "IsPrimaryResource": true,
  "ServiceResourceUserName": "Rachel Adams",
  "ServiceResourceUserId": "005xx000001X7dl",
  "AssignedResourceId": "03rxx0000004gLc",
  "ServiceResourceId": "0Hnxx0000004C92",
  "ServiceResourceUserEmail": "ra@example.com",
  "IsRequiredResource": true
}
```

SvcApptSchdEvent

Represents the service appointment event. This object is included in a streamed notification received on the channels for the parent platform events. You can't subscribe to the SvcApptSchdEvent channel directly. This object is available in API version 50.0 and later.

Supported Calls

`describeSObjects()`

Parent Platform Events

- AppointmentSchedulingEvent
- ServiceAppointmentEvent

Fields

Field	Details
`AdditionalInformation`	**Type** string **Properties** Nillable **Description** Additional information about the service appointment.
`Address`	**Type** string **Properties** Nillable **Description** The address of the service appointment.
`AppointmentType`	**Type** string

Field	Details
	Properties Nillable **Description** The service appointment type.
`ChangedFields`	**Type** complexvalue **Properties** Nillable **Description** List of fields that changed.
`Comments`	**Type** string **Properties** Nillable **Description** Comments about the service appointment.
`ContactId`	**Type** string **Properties** Nillable **Description** ID of the contact associated with the service appointment.
`DurationInMinutes`	**Type** double **Properties** Nillable **Description** The duration of the service appointment in minutes.
`Email`	**Type** string **Properties** Nillable **Description** The email associated with the service appointment.
`EventUuid`	**Type** string

Field	Details
	Properties Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ParentRecordId`	**Type** string **Properties** Nillable **Description** ID of the parent record associated with the service appointment.
`Phone`	**Type** string **Properties** Nillable **Description** The phone number associated with the service appointment.
`ScheduledEndTime`	**Type** dateTime **Properties** Nillable **Description** The scheduled end time of the service appointment.
`ScheduledStartTime`	**Type** dateTime **Properties** Nillable **Description** The scheduled start time of the service appointment.
`ServiceAppointmentId`	**Type** string **Properties** Nillable **Description** ID of the service appointment.

Field	Details
`ServiceTerritoryId`	**Type** string **Properties** Nillable **Description** ID of the service territories associated with the service appointment.
`Status`	**Type** string **Properties** Nillable **Description** The status of the service appointment.
`StatusCategory`	**Type** string **Properties** Nillable **Description** The status category of the service appointment.
`Subject`	**Type** string **Properties** Nillable **Description** The subject of the service appointment.
`WorkTypeBlockTimeAfterAppointment`	**Type** int **Properties** Nillable **Description** The period of time occurring after the appointment that is typically blocked for this work type.
`WorkTypeBlockTimeAfterUnit`	**Type** string **Properties** Nillable

Field	Details
	Description The unit of the period specified for WorkTypeBlockTimeAfterAppointment. Values include `hour` and `minute`.
WorkTypeBlockTimeBeforeAppointment	**Type** int **Properties** Nillable **Description** The period of time occurring before the appointment that is typically blocked for this work type.
WorkTypeBlockTimeBeforeUnit	**Type** string **Properties** Nillable **Description** The unit of the period specified for WorkTypeBlockTimeBeforeAppointment. Values include `hour` and `minute`.
WorkTypeId	**Type** string **Properties** Nillable **Description** ID of the work type associated with the service appointment.

Example

This example shows the service appointment fields associated with the event.

```
{
  "ParentRecordId": "001RM000003rwkfYAA",
  "ContactId": "003RM000006EpajYAC",
  "Status": "None",
  "AdditionalInformation": "Sample additional information",
  "ServiceTerritoryId": "0Hhxx0000004mu4",
  "Comments": "Sample comment",
  "Email": "abc@example.com",
  "Address": "1 Market Street San Francisco CA 94105 United States",
  "WorkTypeId": "08qxx0000004C92",
  "WorkTypeBlockTimeBeforeAppointment": 30,
  "WorkTypeBlockTimeAfterAppointment": 1,
  "WorkTypeBlockTimeBeforeUnit": "minutes",
  "WorkTypeBlockTimeAfterUnit": "hours",
```

```
  "ServiceAppointmentId": "08pxx0000005Ip6",
  "ScheduledEndTime": "2020-02-28T00:45:00.000Z",
  "Subject": "Apply for Chase Sapphire Preferred Card",
  "AppointmentType": "null",
  "StatusCategory": "None",
  "DurationInMinutes": 60,
  "Phone": "4157286216",
  "ScheduledStartTime": "2020-02-27T23:45:00.000Z"
}
```

AssetCancelInitiatedEvent

Notifies subscribers when the process started by the `/asset-management/assets/collection/actions/initiate-cancellation` process is complete. If the process is successful, use this event to learn about the cancellation order that was created. If the process isn't successful, use the RevenueTransactionErrorLog records to learn about the errors and how to fix them. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/AssetCancelInitiatedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`AssetCancelErrorDetailEvents`	**Type** AssetCancelErrorDtlEvent[] on page 152 **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available in API versions 55.0 and 56.0 only.
`CancellationRecordId`	**Type** string **Properties** Nillable **Description** The ID of the cancellation record; for example, the cancellation order. If the process failed, this field is null.
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`HasErrors`	**Type** boolean **Properties** Defaulted on create **Description** `true` if errors occurred during the processing; otherwise `false`.
`ReplayId`	**Type** string

Field	Details
	Properties Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/asset-management/assets/collection/actions/initiate-cancellation` response. Use this ID to identify the event for a specific request.

IN THIS SECTION:

AssetCancelErrorDtlEvent

Contains information about errors that occurred during the processing of an `/asset-management/assets/collection/actions/initiate-cancellation` request. This object is included in an `AssetCancelInitiatedEvent` message. You can't subscribe to `AssetCancelErrorDtlEvent` directly. This object is available in API versions 55.0 and 56.0 only.

AssetCancelErrorDtlEvent

Contains information about errors that occurred during the processing of an `/asset-management/assets/collection/actions/initiate-cancellation` request. This object is included in an `AssetCancelInitiatedEvent` message. You can't subscribe to `AssetCancelErrorDtlEvent` directly. This object is available in API versions 55.0 and 56.0 only.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available if Subscription Management is installed in your org.

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** Nillable **Description** Reference code for the type of error that occurred.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** Information about the error that occurred during processing.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

AssetAmendInitiatedEvent

Notifies subscribers when the process started by the `/asset-management/assets/collection/actions/initiate-amend-quantity` REST request is complete. If the process is successful, use this event to learn about the amendment order that was created. If the process isn't successful, use the RevenueTransactionErrorLog records to learn about the errors and how to fix them. This object is available in API version 56.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓

Subscriber	Supported?
Streaming API (CometD)	✔

Subscription Channel

`/event/AssetAmendInitiatedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`AmendmentRecordId`	**Type** string **Properties** Nillable **Description** The ID of the amendment record; for example, the amendment order. If the process failed, this field is null.
`AssetAmendErrorDetailEvents`	**Type** AssetAmendErrorDtlEvent[] **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available in API versions 55.0 and 56.0 only.
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string

Field	Details
	Properties Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`HasErrors`	**Type** boolean **Properties** Defaulted on create **Description** `true` if errors occurred during the processing of this request; otherwise `false`.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `requestIdentifier` parameter in the `/asset-management/assets/collection/actions/initiate-renew` response. Use this ID to identify the event for a specific request.

IN THIS SECTION:

AssetAmendErrorDtlEvent

Contains information about errors that occurred during the processing of an `/asset-management/assets/collection/actions/initiate-amend-quantity` request. This object is included in an `AssetAmendInitiatedEvent` message. You can't subscribe to `AssetAmendErrorDtlEvent` directly. This object is available in API version 56.0 only.

AssetAmendErrorDtlEvent

Contains information about errors that occurred during the processing of an `/asset-management/assets/collection/actions/initiate-amend-quantity` request. This object is included in an `AssetAmendInitiatedEvent` message. You can't subscribe to `AssetAmendErrorDtlEvent` directly. This object is available in API version 56.0 only.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available when Subscription Management is enabled.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** Nillable **Description** Reference code for the type of error that occurred.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** Information about the error that occurred during processing.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

AssetRenewInitiatedEvent

Notifies subscribers when the process started by the `/asset-management/assets/collection/actions/initiate-renew` REST request is complete. If the process is successful, use this event to learn about the renewal order that was created. If the process isn't successful, use the RevenueTransactionErrorLog records to learn about the errors and how to fix them. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/AssetRenewInitiatedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`AssetRenewErrorDetailEvents`	**Type** AssetRenewErrorDtlEvent[] on page 159 **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available in API versions 55.0 and 56.0 only.

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`HasErrors`	**Type** boolean **Properties** Defaulted on create **Description** Contains `true` if errors occurred during the process; otherwise `false`. The default value is `false`.
`RenewalRecordId`	**Type** string **Properties** Nillable **Description** The ID of the renewal record; for example, the renewal order. If the process failed, this field is null.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** The unique ID returned in the `requestIdentifier` parameter in the `/asset-management/assets/collection/actions/initiate-renew` response. Use this ID to identify the event for a specific request.

IN THIS SECTION:

AssetRenewErrorDtlEvent

Contains information about errors that occurred during the processing of an `/asset-management/assets/collection/actions/initiate-renew` request. This object is included in an `AssetRenewInitiatedEvent` message. You can't subscribe to `AssetRenewErrorDtlEvent` directly. This object is available in API versions 55.0 and 56.0 only.

AssetRenewErrorDtlEvent

Contains information about errors that occurred during the processing of an `/asset-management/assets/collection/actions/initiate-renew` request. This object is included in an `AssetRenewInitiatedEvent` message. You can't subscribe to `AssetRenewErrorDtlEvent` directly. This object is available in API versions 55.0 and 56.0 only.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** Nillable **Description** Reference code for the type of error that occurred.
`ErrorMessage`	**Type** string

Field	Details
	Properties Nillable **Description** Information about the error that occurred during processing.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

AssetTokenEvent

Notifies subscribers of asset token issuance and registration of a connected device as an Asset. This object is available in API version 39.0 and later.

An asset token event records successful completion of an OAuth 2.0 asset token flow for a connected device. An event is published whenever an access token and actor token (optional) are successfully exchanged for an asset token. This object is designed to support custom business processes, such as automatic logging of a case when an event occurs. Create Apex triggers that subscribe to an event and execute after asset token issuance. This object is read only and can't be retrieved using a SOQL query. Asset token events are not displayed in the Setup user interface for Platform Events.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/AssetTokenEvent`

Event Delivery Allocation Enforced

Yes

Fields

Field Name	Details
`ActorTokenPayload`	**Type** textarea **Properties** Nillable **Description** If the asset token request included an actor token, the payload portion containing claims about the connected device, asset token, and if applicable, the registered Asset.
`AssetId`	**Type** reference **Properties** Nillable **Description** ID of the Asset record if the Asset was newly created or an existing Asset was linked to in the asset token request.
`AssetName`	**Type** string **Properties** Nillable **Description** If specified in the actor token, the display name of the existing Asset. This value is otherwise `null`.
`AssetSerialNumber`	**Type** string **Properties** Nillable **Description** If specified in the actor token, the serial number of the existing Asset. This value is otherwise `null`.
`ConnectedAppId`	**Type** reference **Properties** Nillable

Field Name	Details
	Description ID of the connected app associated with the access token for the device.
`DeviceId`	**Type** string **Properties** Nillable **Description** ID of the connected device. Value is the `did` (device ID) claim specified in the actor token.
`DeviceKey`	**Type** textarea **Properties** Nillable **Description** If specified in the actor token, the device-specific RSA public key as a JSON Web Key (JWK). Value is the `jwk` claim within the confirmation claim from the actor token.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`Expiration`	**Type** dateTime **Properties** Nillable **Description** The expiration time on or after which the asset token JWT must not be accepted for processing. A numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds.
`Name`	**Type** string **Properties** Nillable

Field Name	Details
	Description Display name of the asset token.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`UserId`	**Type** reference **Properties** Nillable **Description** ID of the user associated with the access token.

Usage

The following example shows how to trigger an action after an asset token event.

```
trigger AssetTokenEventTrigger on AssetTokenEvent (after insert) {
      System.assertEquals(1,Trigger.new.size(),'One record expected');
      AssetTokenEvent event = Trigger.new[0];
      AssetTokenRecord__c record = new AssetTokenRecord__c();
      record.ConnectedAppId__c = event.ConnectedAppId;
      record.UserId__c = event.UserId;
      record.AssetId__c = event.AssetId;
      record.AssetTokenName__c = event.AssetTokenName;
      record.DeviceId__c = event.DeviceId;
      record.DeviceKey__c = event.DeviceKey;
      record.Expiration__c = event.Expiration;
      record.AssetSerialNumber__c = event.AssetSerialNumber;
      record.AssetName__c = event.AssetName;
      record.ActorTokenPayload__c = event.ActorTokenPayload;
      insert(record);
}
```

BatchApexErrorEvent

Notifies subscribers of errors and exceptions that occur during the execution of a batch Apex class. This object is available in API version 44.0 and later.

Batch Apex classes can fire platform events when encountering an error or exception. Clients listening to the event channel can tell how often it failed, which records were in scope at the time of failure, and other exception details.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/BatchApexErrorEvent`

Special Access Rules

Only the Salesforce Platform can fire this event; Apex code and the API cannot. Users with Customize Application Permission have read access.

Event Delivery Allocation Enforced

Yes

Fields

Field Name	Details
`AsyncApexJobId`	**Type** string **Properties** Nillable **Description** The AsyncApexJob record for the batch Apex job that fired this event.
`DoesExceedJobScopeMaxLength`	**Type** boolean **Properties** Defaulted on create

Field Name	Details
	Description True if the JobScope field is truncated due to the message exceeding the character limit.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ExceptionType`	**Type** string **Properties** Nillable **Description** The Apex exception type name. Internal platform errors are represented as the `System.UnexpectedException` type.
`JobScope`	**Type** textarea **Properties** Nillable **Description** The Record IDs that are in scope if the event was fired from the `execute()` method of a batch job. If the batch job uses custom iterators instead of sObjects, `JobScope` is the `toString()` representation of the iterable objects. Maximum length is 40000 characters.
`Message`	**Type** string **Properties** Nillable **Description** Exception message text. Maximum length is 5000 characters.
`Phase`	**Type** string **Properties** Nillable

Field Name	Details
	Description The phase of the batch job when it encountered an error. **Possible Values** • START • EXECUTE • FINISH
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestId`	**Type** string **Properties** Nillable **Description** The unique ID of the batch job that fired the event. Event monitoring customers can use this information to correlate the error with logging information.
`StackTrace`	**Type** string **Properties** Nillable **Description** The Apex stacktrace of the exception, if available. Maximum length is 5000 characters.

Usage

`BatchApexErrorEvent` messages are generated by batch Apex jobs that implement the `Database.RaisesPlatformEvents` interface and have unhandled Apex exceptions during processing. For more information, see the Apex Developer Guide.

BillingScheduleCreatedEvent

Notifies subscribers when the `/actions/standardCreateBillingScheduleFromOrderItem` request is complete. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/billingschedulecreatedevent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`BillSchdCreatedEventDetail`	**Type** BillSchdCreatedEventDetail[] on page 168 **Properties** Nillable **Description** One `BillingScheduleCreatedEventDetail` entry is created for each order item in the `BillingScheduleCreatedEvent` request. One `BillSchdCreatedEventDetail` is created for each error that occurred.
`CorrelationIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** ID returned in the `CreateBillingScheduleFromOrderItem` response. Use this ID to identify the `BillingScheduleCreatedEvent` for a specific request.

IN THIS SECTION:

BillSchdCreatedEventDetail

Contains information about each order item in the `/actions/standardCreateBillingScheduleFromOrderItem` request and any errors that occurred while processing the request. This object is included in an `BillingScheduleCreatedEvent` message. You can't subscribe to `BillSchdCreatedEventDetail` directly. This object is available in API version 55.0 and later.

BillSchdCreatedEventDetail

Contains information about each order item in the `/actions/standardCreateBillingScheduleFromOrderItem` request and any errors that occurred while processing the request. This object is included in an `BillingScheduleCreatedEvent` message. You can't subscribe to `BillSchdCreatedEventDetail` directly. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available when Subscription Management is enabled in your org.

Fields

Field	Details
`BillingScheduleId`	**Type** Id **Properties** Nillable **Description** If the request was successful, this field contains the ID of the billing schedule for the order item.
`ErrorCode`	**Type** string **Properties** Nillable **Description** If the request wasn't successful, this field contains the error code.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** If the request wasn't successful, this field contains the error message.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`IsSuccess`	**Type** boolean **Properties** Nillable

Field	Details
	Description Indicates whether the request to create a billing schedule for the order item was successful.
`OrderItemId`	**Type** reference **Properties** Nillable **Description** The ID of the order item used in the `/actions/standardCreateBillingScheduleFromOrderItem` REST request. This field is a relationship field. **Relationship Name** OrderItem **Relationship Type** Lookup **Refers To** OrderItem

CommerceDiagnosticEvent

Tracks checkout, pricing, search, and other activity within your Commerce implementation to monitor events and diagnose issues. This object is available in API version 49.0 and later.

Supported Calls

`create()`,`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/CommerceDiagnosticEvent`

Special Access Rules

CommerceDiagnosticEvent is available only if the B2B Commerce license is enabled.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`B2bEdition`	**Type** string **Properties** Create, Nillable **Description** The edition of B2B Commerce. The edition can include Lightning (LB2B), CCRZ, or future flavors. This field is available in API version 51.0 and later.
`B2bVersion`	**Type** string **Properties** Create, Nillable **Description** This field is optional. For a managed package, `B2BVersion` includes Major, Minor, Patch revision numbers. For Lightning, `B2BVersion` includes the optional service version. This field is available in API version 51.0 and later.
`BrowswerDeviceType`	**Type** int **Properties** Create, Nillable **Description** A code used to identify the browser and device type. This field is available in API version 51.0 and later. The code is in the format “BBVVVXYZ,” with the following signification: • BB — Two digits that indicate the browser type. – INTERNET_EXPLORER: “10” – CHROME: "13" – FIREFOX: "11" – SAFARI: "14" – OPERA: "15" – ANDROID_WEBKIT: "16"

Field	Details
	– NETSCAPE: "17" – OTHER_WEBKIT: "18" – OTHER_GECKO: "19" – OTHER_KHTML: "20" – OTHER_MOBILE: "21" – SALESFORCE_DESKTOP: "22" – BLACKBERRY: "23" – GOOD_ACCESS: "24" – EDGE: "25" – SALESFORCE_MOBILE: "26" • VVV—Three digits that indicate version, with leading zeroes. • XYZ—Browser-type specific flags or options. Each digit in XYZ represents a different flag depending on the BrowserType: – X=1: If the parser recognizes a "touch" browser. Here, touch means the older touch native client, not that the device supports touch. – Y=1: If the parser recognizes a browser in compatibility mode. Only for IE. – Z=1: If the browser is recognized as `MOBILE`. – Z=2: If the browser is recognized as `PHONE`. – Z=3: If the browser is recognized as `TABLET`. – Z=4: If the browser is a recognized as `MEDIA PLAYER`. – Z=6: Only for Opera Mini.
`ContextId`	**Type** string **Properties** Create, Nillable **Description** The Key Business Domain Value in which the operation is done. For example, for Cart, the `ContextId` is *`cartId`*.
`ContextId2`	**Type** string **Properties** Create, Nillable **Description** Another field used to identify a context ID for a given operation.
`ContextMap`	**Type** string

Field	Details
	Properties Create, Nillable **Description** A JSON string that captures extra operational context or other diagnostic information.
`CorrelationId`	**Type** string **Properties** Create, Nillable **Description** Used to correlate client and server calls, and other async calls to Commerce subsystems. Calls can take place across several services and operations.
`Count`	**Type** int **Properties** Create, Nillable **Description** The number of records impacted by an operation.
`EffectiveAccountId`	**Type** string **Properties** Create, Nillable **Description** The Commerce Effective Account ID in the context of an operation.
`ErrorCode`	**Type** string **Properties** Create, Nillable **Description** The API error code that appears when an operation fails.
`ErrorMessage`	**Type** string **Properties** Create, Nillable **Description** The user-friendly error message that appears when an operation fails.

Field	Details
`EventDate`	**Type** dateTime **Properties** Create, Nillable **Description** The date when the event occurred.
`EventIdentifier`	**Type** string **Properties** Create, Nillable **Description** The unique ID of the event, which is shared with the corresponding object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field value to correlate the event with its corresponding object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`IsRetry`	**Type** boolean **Properties** Create, Defaulted on create **Description** Describes whether an event occurred during a retried operation (`true`), or not (`false`). Default value is `false`.
`Operation`	**Type** string **Properties** Create, Nillable **Description** The operation where the event originated. For example, `CreateCart`, `EditCart`, and `CreateOrder`.
`OperationStage`	**Type** string

Field	Details
	Properties Create, Nillable **Description** The stage of the operation where the event originated. This value varies depending on the operation.
`OperationStatus`	**Type** string **Properties** Create, Nillable **Description** The status of the operation. Values include: • `Success` • `SystemError` • `AdminError` • `UserError` • `DependencyError`
`OperationTime`	**Type** string **Properties** Create, Nillable **Description** Duration of the operation in minutes and/or seconds.
`OsVersion`	**Type** int **Properties** Create, Nillable **Description** Code used to identify the operating system and version. `OsVersion` is equal to 9999 for an unknown platform. This field is available in API version 51.0 and later.
`RelatedEventIdentifier`	**Type** string **Properties** Create, Nillable **Description** EventIdentifier (UUID) of the related event.

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`ServiceName`	**Type** string **Properties** Create, Nillable **Description** The service where the event originated. When Commerce generates the event, possible values include: • `BuyerGroup` • `BuyerAccount` • `BuyerManagement` • `Cart` • `CartAsync` • `Checkout` • `Entitlements` • `Order` • `Pricing` • `ProductEtl` • `Products` • `ReOrder` • `Search` • `Storefront` • `Integration` • `Wishlist` • `ExternalManagedAccouts` • `EffectiveAccountService` • `EffectiveAccountUIService`
`UserId`	**Type** string

Field	Details
	Properties Create, Nillable **Description** The ID of the user associated with this event.
`Username`	**Type** string **Properties** Create, Nillable **Description** Reserved for future use.
`WebStoreId`	**Type** string **Properties** Create, Nillable **Description** The ID of the Webstore associated with this event.
`WebStoreType`	**Type** string **Properties** Create, Nillable **Description** The type of webstore. For example: `B2B`, `B2C`, and `OMS`. This field is available in API version 51.0 and later.

SEE ALSO:

Subscribing to Platform Events

ConsentEvent

Notifies subscribers of changes to consent fields or contact information on all core objects. This object is available in API version 50.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/ConsentEvent`

Special Access Rules

Users with ReadAllData or PrivacyDataAccess permissions have read access.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`AssociatedIds`	**Type** string **Properties** Nillable **Description** A list of IDs associated with the changed record. Possible IDs are: • `globalPartyId` • `individual` • `lead` • `contact` • `personAccount` • `user` • `contactPoint` • `contactPointConsent` • `contactPointTypeConsent`

Field	Details
`ChangeInitiator`	**Type** string **Properties** Nillable **Description** The ID of the user who changed the record.
`ChangeTimestamp`	**Type** dateTime **Properties** Nillable **Description** Indicates the date and time the change event occurred.
`ChangeType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Indicates the type of change made to the record. Possible values are: • `Create` • `Delete` • `Undelete` • `Unknown` • `Update`
`ConsentCaptureSource`	**Type** string **Properties** Nillable **Description** Indicates how consent was captured. For example, if the ConsentCaptureType is a website, the ConsentCaptureSource is the website URL.
`ConsentCaptureType`	**Type** string **Properties** Nillable

Field	Details
	Description Indicates the type of source consent was captured through. For example, a website or online form.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`NewValues`	**Type** string **Properties** Nillable **Description** Indicates new values that were added to the object, if relevant.
`ObjectName`	**Type** string **Properties** Nillable **Description** The name of the object for which the change event was captured.
`RecordId`	**Type** string **Properties** Nillable **Description** The ID of the record that was changed.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

ConsentUnsubscribeAllEvent

Notifies subscribers when a user unsubscribes from all communications on a preference form created in Preference Manager. This object is available in API version 60.0 and later.

Supported Calls

`create()`, `describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/ConsentUnsubscribeAllEvent`

Event Delivery Allocation Enforced

Yes

Special Access Rules

This object is available for users with the Privacy Center license and the Manage Preference Manager user permission.

Fields

Field	Details
`EventDetails`	**Type** textarea **Properties** Create **Description** JSON text that contains the details for the unsubscribe all event, such as the timestamp for when the event happened.
`EventUuid`	**Type** string

Field	Details
	Properties Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

CreateAssetOrderEvent

Notifies subscribers that the process started by the `/actions/standard/createOrUpdateAssetFromOrder` request is complete. If the process is successful, use this event to learn about the new assets. If the request isn't successful, use this event to learn about the errors and how to fix them. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/CreateAssetOrderEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available if Subscription Management is installed in your org. Users must have Read access on this event to receive or view event notifications.

Fields

Field	Details
`AssetDetails`	**Type** CreateAssetOrderDtlEvent on page 185 **Properties** Nillable **Description** A list of AssetDetail records created as a result of a successful `createOrUpdateAssetFromOrder` request. Each AssetDetail contains an order item ID, asset ID, and IsSuccess flag. If the request failed, the AssetDetail also contains an error code and error message.
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** The unique ID returned in the `/createOrUpdateAssetFromOrder` response. Use this ID to identify the event for a specific request.

Example: A user successfully runs a `createOrUpdateAssetFromOrder` request on an order with two order items. The published createAssetOrderEvent contains the following information.

- RequestId: 0001
- AssetDetail
  - OrderItemId: 802XX0000000001
  - AssetId: 02iXX000000001
  - IsSuccess: True
- AssetDetail
  - OrderItemId: 802XX0000000001
  - AssetId: 02iXX000000002
  - IsSuccess: True

Example: A user runs a `createOrUpdateAssetFromOrder` request on an order with two order items, but doesn't have Create access on assets. The request fails, and the published createAssetOrderEvent contains the following information.

- RequestId: 0002
- AssetDetail
  - OrderItemId: 802XX0000000001
  - IsSuccess: False
  - ErrorCode: `INSUFFICIENT_ACCESS`
  - ErrorMessage: User doesn't have Create Access to asset.
- AssetDetail
  - OrderItemId: 802XX0000000001
  - IsSuccess: False
  - ErrorCode: `INSUFFICIENT_ACCESS`
  - ErrorMessage: User doesn't have Create Access to asset.

IN THIS SECTION:

CreateAssetOrderDtlEvent

Contains information about an attempt to create or update an asset as a result of `/actions/standard/createOrUpdateAssetFromOrder`. If the request was successful, the event shows information about the asset. If the request failed, the event shows error information. This object is included in an `CreateAssetOrderEvent` message. You can't subscribe to `CreateAssetOrderDtlEvent` directly. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/CreateAssetOrderDtlEvent`

Special Access Rules

This object is available if Subscription Management is installed in your org. Users must have Read access on this event to receive or view event notifications.

Fields

Field	Details
`AssetId`	**Type** reference **Properties** Nillable

Field	Details
	Description The ID of the asset that was created or updated. This field is a relationship field. **Relationship Name** Asset **Relationship Type** Lookup **Refers To** Asset
`ErrorCode`	**Type** string **Properties** Nillable **Description** Reference code for the type of error that occurred.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** Information about the error that occurred after the request was made.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the request to create the asset for the order item was successful (true) or not (false). The default value is `false`. Available in API version 61.0 and later.

Field	Details
`OrderItemId`	**Type** reference **Properties** **Description** The ID of the order item used in the request. Available in API version 61.0 and later. This field is a relationship field. **Relationship Name** OrderItem **Relationship Type** Lookup **Refers To** OrderItem
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

CreditInvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/invoices/{invoiceId}/actions/credit` request is complete. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/CreditInvoiceProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`CrMemoProcessErrDtlEvents`	**Type** CreditMemoProcessedErrDtlEvent[] **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available only in API versions 55.0–58.0. See the `ErrorDetails` field for error messages and error codes.
`CreditMemoId`	**Type** reference **Properties** Nillable **Description** The credit memo created as the result of a successful request. This field is a relationship field. **Relationship Name** CreditMemo **Relationship Type** Lookup **Refers To** CreditMemo

Field	Details
`ErrorDetails`	**Type** string **Properties** Nillable **Description** If the request fails, this field shows error messages, error codes, and the ID of the record on which the errors occurred. This field is available in API 58.0 and later.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`InvoiceId`	**Type** reference **Properties** Nillable **Description** The invoice credited as the result of a successful request. This field is a relationship field. **Relationship Name** Invoice **Relationship Type** Lookup **Refers To** Invoice
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the request was successful. The default value is 'false'.
`ReplayId`	**Type** string **Properties** Nillable

Field	Details
	Description Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the response. Use this ID to identify the event for a specific request.

IN THIS SECTION:

CrMemoProcessErrDtlEvent

Contains information about errors that occurred while creating or applying a credit memo as part of a request. This object is included in a `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` message. You can't subscribe to `CrMemoProcessErrDtlEvent` directly. This object is available in API versions 55.0–58.0. In API version 58.0, this field returns a null result. See the `ErrorDetails` field on the `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` object for error information.

CrMemoProcessErrDtlEvent

Contains information about errors that occurred while creating or applying a credit memo as part of a request. This object is included in a `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` message. You can't subscribe to `CrMemoProcessErrDtlEvent` directly. This object is available in API versions 55.0–58.0. In API version 58.0, this field returns a null result. See the `ErrorDetails` field on the `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` object for error information.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** Nillable **Description** Reference code for the type of error that occurred.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** Information about the error that occurred during processing.
`ErrorSourceId`	**Type** reference **Properties** Nillable **Description** The ID of the record on which the error occurred during the credit memo creation process and the application process. This field is a polymorphic relationship field. **Relationship Name** ErrorSource **Relationship Type** Lookup **Refers To** CreditMemo, CreditMemoLine, Invoice, InvoiceLine
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

CreditMemoProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/credit-memos` request is complete. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/CreditMemoProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`CrMemoProcessErrDtlEvents`	**Type** CreditMemoProcessedErrDtlEvent[] on page 335

Field	Details
	Properties Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available only in API versions 55.0–58.0. See the `ErrorDetails` field for error messages and error codes.
`CreditMemoId`	**Type** reference **Properties** Nillable **Description** The credit memo created as the result of a successful request. This field is a relationship field. **Relationship Name** CreditMemo **Relationship Type** Lookup **Refers To** CreditMemo
`ErrorDetails`	**Type** string **Properties** Nillable **Description** If the request fails, this field shows error messages, error codes, and the ID of the record on which the errors occurred. This field is available in API 58.0 and later.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create

Field	Details
	Description Indicates whether the Create Standalone Credit Memo action was successful. The default value is 'false'.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/commerce/invoicing/credit-memos` response. Use this ID to identify the event for a specific request.

Example: A user successfully runs a `/commerce/invoicing/credit-memos`, creates one credit memo, and receives this platform event when the request completes.

```
{
  "IsSuccess": true,
  "CrMemoProcessErrDtlEvents": null,
  "CreatedById": "005RO0000000g4LYYAY",
  "CorrelationIdentifier": "50gRO0000000jxc",
  "CreatedDate": "2023-03-17T15:09:18Z",
  "ErrorDetails": "[]",
  "InvoiceId": "3ttRO0000006839YAA",
  "CreditMemoId": "50gRO0000000jxcYAA",
  "RequestIdentifier": "d488e070-0fd8-4cde-a9fd-d7ca38d040f5"
}
```

Example: A user runs a `/commerce/invoicing/invoices/{invoiceId}/actions/credit` request, which fails because the credit memo's amount is greater than the invoice's balance.

```
{
  "IsSuccess": false,
  "CrMemoProcessErrDtlEvents": null,
  "CreatedById": "005RO0000000g4LYYAY",
  "CorrelationIdentifier": "50gRO0000000jzi",
```

```
  "CreatedDate": "2023-03-17T22:55:11Z",
    "ErrorDetails": "[{
 "ErrorSourceId": "50gRO0000000jzi",
 "ErrorCode": "RECORD_UPDATE_FAILED",
 "ErrorMessage": "An error occurred while updating the credit memo status to POSTED:
Child events testing - fail updating credit memo status to posted Failed object Ids :
 50gRO0000000jzi"
}]",
  "CreditMemoId": "50gRO0000000jziYAA",
  "RequestIdentifier": "9123a706-4a64-4beb-8942-4eb5abd1e59f"
 },
```


DataObjectDataChgEvent

Notifies subscribers of an action within Data Cloud. This object is available in API version 53.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/DataObjectDataChgEvent`

Special Access Rules

DataObjectDataChgEvent is available only if Data Cloud is enabled.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`ActionDeveloperName`	**Type** string **Properties** Nillable **Description** The developer name associated with this action.
`EventCreationDateTime`	**Type** dateTime **Properties** Nillable **Description** The date and time when the event occurred.
`EventPrompt`	**Type** picklist **Properties** Nillable, Restricted Picklist **Description** The data manipulation language action that triggered this event. Possible values are: • `DELETE` • `INSERT` • `UPDATE`
`EventPublishDateTime`	**Type** dateTime **Properties** Nillable **Description** The date and time when the event was published.
`EventSchemaVersion`	**Type** string **Properties** Nillable **Description** The version of the event schema.

Field	Details
`EventType`	**Type** string **Properties** Nillable **Description** The type of event that occurred.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`Offset`	**Type** string **Properties** Nillable **Description** The number of rows to skip before starting to return.
`PayloadCurrentValue`	**Type** textarea **Properties** Nillable **Description** Current data values with enriched fields.
`PayloadPrevValue`	**Type** textarea **Properties** Nillable **Description** Previous data values with enriched fields. This field is optional depending on the source object.
`PayloadSchema`	**Type** textarea **Properties** Nillable

Field	Details
	Description The schema for the event payload.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SourceObjectDeveloperName`	**Type** string **Properties** Nillable **Description** The developer name of the object that triggered the data change event.

DataObjectMetadataChgEvent

Notifies subscribers of a metadata change within Data Cloud for these objects: Data Lake, Data Model, and Calculated Insight. This object is available in API version 53.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/DataObjectMetadataChgEvent`

Special Access Rules

DataObjectMetadataChgEvent is available only if Data Cloud is enabled.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`CurrentValue`	**Type** textarea **Properties** Nillable **Description** The serialized schema of the current metadata.
`EventCreationDate`	**Type** dateTime **Properties** Nillable **Description** The date and time when the event occurred.
`EventPublishDate`	**Type** dateTime **Properties** Nillable **Description** The date and time when the event published.
`EventType`	**Type** string **Properties** Nillable **Description** The type of event that occurred.
`EventUuid`	**Type** string

Field	Details
	Properties Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`PreviousValue`	**Type** textarea **Properties** Nillable **Description** The serialized schema of the metadata before the change.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SchemaVersion`	**Type** string **Properties** Nillable **Description** The version of the event schema.
`SourceTableDeveloperName`	**Type** string **Properties** Nillable **Description** The source entity name on which the metadata change has occurred.
`Trigger`	**Type** string **Properties** Nillable

Field	Details
	Description The trigger data definition language type that caused the event. Examples: DELETE, INSERT, or UPDATE

DatasetExportEvent

Notifies subscribers on the export of an Analytics dataset. This object is available in API version 41.0 and later.

Supported Calls

`create()`, `describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/DatasetExportEvent`

Special Access Rules

DatasetExportEvent is available only if the CRM Analytics license is enabled.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`DataflowInstanceId`	**Type** string **Properties** Create, Nillable

Field	Details
	Description The ID of the dataflow instance for the dataset.
`DataflowExportId`	**Type** string **Properties** Create, Nillable **Description** The ID of the dataset export.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`Message`	**Type** string **Properties** Create, Nillable **Description** The message for the dataset export.
`Owner`	**Type** string **Properties** Create, Nillable **Description** The owner of the dataset export.
`PublisherInfo`	**Type** string **Properties** Create, Nillable **Description** The publisher information for the dataset export.
`PublisherType`	**Type** picklist **Properties** Create, Nillable, Restricted picklist

Field	Details
	Description The publisher type for the dataset export. Values include: • `EinsteinDiscovery`
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`Status`	**Type** picklist **Properties** Create, Nillable, Restricted picklist **Description** The status the dataset export. Values include: • `Cancelled` • `Completed` • `Failed` • `InProgess` • `New`

DiscoveryPredictionEvent

Notifies subscribers when Einstein Discovery has written prediction history results. This object is available in API version 57.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓

Subscriber	Supported?
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/DiscoveryPredictionEvent`

Special Access Rules

Users with CRM Analytics licenses have read access.

Fields

Field	Details
`ActualValue`	**Type** string **Properties** Nillable **Description** The actual value of the outcome field on the Einstein Discovery predicted object.
`CreatedBy`	**Type** user **Properties** Nillable **Description** The user that started the Einstein Discovery prediction run.
`CreatedById`	**Type** ID **Properties** Nillable **Description** The unique ID of the user that started the Einstein Discovery prediction run.
`CreatedDate`	**Type** dateTime **Properties** Nillable **Description** The creation date of the event.

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`GoalId`	**Type** string **Properties** Nillable **Description** The unique ID of the Einstein Discovery prediction goal.
`HasError`	**Type** boolean **Properties** Defaulted on create **Description** `true` if there was an error while making the prediction, `false` otherwise.
`ModelId`	**Type** string **Properties** Nillable **Description** The unique ID of the Einstein Discovery model used for the prediction.
`PredictedValue`	**Type** string **Properties** Nillable **Description** The predicted value from the Einstein Discovery prediction run.
`PredictionTime`	**Type** dateTime **Properties** Nillable **Description** The date and time of the Einstein Discovery prediction run.

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RunId`	**Type** string **Properties** Nillable **Description** The unique ID of the Einstein Discovery prediction run.
`TargetEntity`	**Type** string **Properties** Nillable **Description** The target entity that Einstein Discovery is writing prediction results to.
`TargetField`	**Type** string **Properties** Nillable **Description** The target field that Einstein Discovery is writing the prediction value to.
`TargetId`	**Type** string **Properties** Nillable **Description** The unique ID of the target entity Einstein Discovery is writing prediction results to.
`ValueType`	**Type** string **Properties** Nillable

Field	Details
	Description The type of the Einstein Discovery prediction value.

ExtlRecShrEvent

Reserved for future use. This object is available in API version 61.0 and later.

FirstBillPaymentSetupEvent

Notifies subscribers when a first bill payment is set up. This object is available in API version 60.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/`

Event Delivery Allocation Enforced

Yes

Special Access Rules

FirstBillPaymentSetupEvent is available when B2B Commerce or D2C Commerce is enabled in your org.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** Correlation ID passed in the invocable action input.
`ErrorCode`	**Type** string **Properties** Nillable **Description** Error code for a failed first bill payment setup result.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** Error message for a failed first bill payment setup result.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Required. Shows whether the first bill payment is successfully set up. The default value is `false`.
`OrderSummaryId`	**Type** reference **Properties** Nillable **Description** ID of the OrderSummary in the first bill payment setup process. This field is a relationship field.

Field	Details
	Relationship Name OrderSummary **Relationship Type** Lookup **Refers To** OrderSummary
`ProcessingStage`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Processing Stage Enum. Possible values are: • `BillingSchedulesCreation` • `InvoiceGeneration` • `PaymentApplication` • `PaymentMethodAssociation` • `PreProcessing`
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** Request ID returned as an output of an invocable action.

FlowExecutionErrorEvent

Notifies subscribers of errors related to screen flow executions. This object is available in API version 47.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`ContextObject`	**Description** Reserved for future use.
`ContextRecordId`	**Description** Reserved for future use.
`ElementApiName`	**Type** string **Properties** Nillable **Description** The API name of the flow element that was executed when the flow execution error occurred.
`ElementType`	**Type** string **Properties** Nillable **Description** The type of flow element.
`ErrorId`	**Type** string

Field	Details
	Properties Nillable **Description** The ID of the error.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** The message about the error that occurred.
`EventDate`	**Type** dateTime **Properties** None **Description** Required. The date and time when the error occurred. This field always contains a value.
`EventIdentifier`	**Type** string **Properties** None **Description** Required. The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object. This field always contains a value.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`EventType`	**Type** string **Properties** None

Field	Details
	Description Required. The type of flow event. Valid value is Error—An event that occurs when a flow execution generates an error. This field always contains a value.
`ExtendedErrorCode`	**Type** string **Properties** Nillable **Description** The code that references more details about the error.
`FlowApiName`	**Type** string **Properties** None **Description** Required. The API name of the flow that the error occurred for. This field always contains a value.
`FlowExecutionEndDate`	**Description** Reserved for future use.
`FlowExecutionStartDate`	**Type** dateTime **Properties** Nillable **Description** The date and time when the error-generating flow execution starts.
`FlowNamespace`	**Type** string **Properties** Nillable **Description** The namespace of the error-generating flow.
`FlowVersionId`	**Type** string **Properties** None **Description** Required. The ID of the error-generating flow version. This field always contains a value.

Field	Details
`InterviewBatchId`	**Description** Reserved for future use.
`InterviewGuid`	**Type** string **Properties** None **Description** Required. The globally unique identifier of the error-generating flow interview. This field always contains a value.
`InterviewRequestId`	**Description** Reserved for future use.
`InterviewStartDate`	**Type** dateTime **Properties** None **Description** Required. The date and time when the error-generating flow interview starts. This field always contains a value.
`InterviewStartedById`	**Type** reference **Properties** None **Description** Required. The ID of the flow interview when it was started. This field always contains a value.
`ProcessType`	**Type** string **Properties** Nillable **Description** The type of the flow. Valid value is: • `Flow`—A flow that requires user interaction because it contains one or more screens or local actions, choices, or dynamic choices. In the UI and Salesforce Help, it's a screen flow. Screen flows can be launched from the UI, such as with a flow action, Lightning page, or web tab.
`RelatedRecordId`	**Description** Reserved for future use.

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`StageQualifiedApiName`	**Description** Reserved for future use.

FlowOrchestrationEvent

Notifies subscribers that a paused instance of an orchestration is ready to be resumed. This object is available in API version 53.0 and later.

A `FlowOrchestrationEvent` is automatically published when an assigned user completes a screen flow associated with an interactive step, when an asynchronous background step is completed, or when a MuleSoft step is completed.

Supported Calls

`create()`, `describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	✓
Processes	
Streaming API (CometD)	

Event Delivery Allocation Enforced

No

Fields

Field	Details
`EventPayload`	**Type** textarea **Properties** Create, Nillable **Description** Output parameters from the interactive, asynchronous background step, or MuleSoft step that generated the event. Setting the payload for manually published flow orchestration events isn't supported.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`OrchestrationInstanceId`	**Type** string **Properties** Create **Description** The orchestration instance being tracked.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`StepInstanceId`	**Type** string **Properties** Create, Nillable **Description** The ID of the step instance that generated the event.

Field	Details
`StepStatus`	**Type** picklist **Properties** Create, Nillable, Restricted picklist **Description** The resultant status of the step instance that generated the event. If specified, `StepInstanceId` is required. Possible values are: • `Canceled`—For internal use only. • `Completed`—The step instance completed. • `Discontinued`—For internal use only. • `Error`—The step instance or a screen flow associated with the step encountered an error. • `InProgress`—For internal use only. • `NotStarted`—For internal use only. • `Suspended`—For internal use only. This field is available in API version 55.0 and later.

FOStatusChangedEvent

Notifies subscribers of changes to the status of a fulfillment order record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 48.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/FOStatusChangedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

FOStatusChangedEvent is available as part of Salesforce Order Management.

Fields

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`FulfillmentOrderId`	**Type** reference **Properties** Nillable **Description** ID of the FulfillmentOrder whose status changed. This value is functionally required, but is nillable because fulfillment order records can be deleted to comply with data protection and privacy requirements.
`NewStatus`	**Type** picklist **Properties** None **Description** Required. The new value of the Status field on the FulfillmentOrder. Possible values are defined by the Status field picklist on the FulfillmentOrder object. Default available values are: • `Allocated` • `Assigned` • `Canceled` • `Draft` • `Fulfilled` • `Pack Complete` This value is available in API v57.0 and later. • `Pick Complete`

Field	Details
	• `Pickpack` • `Printed` • `Rejected`
`NewStatusCategory`	**Type** picklist **Properties** Restricted picklist **Description** Required. The new value of the StatusCategory field on the FulfillmentOrder. Possible values are: • `Activated` • `Canceled` • `Closed` • `Draft` • `Fulfilling` • `Rejected`
`OldStatus`	**Type** picklist **Properties** Nillable **Description** The previous value of the Status field on the FulfillmentOrder. Possible values are defined by the Status field picklist on the FulfillmentOrder object. Default available values are: • `Allocated` • `Assigned` • `Canceled` • `Draft` • `Fulfilled` • `Pack Complete` This value is available in API v57.0 and later. • `Pick Complete` • `Pickpack` • `Printed` • `Rejected`
`OldStatusCategory`	**Type** picklist

Field	Details
	Properties Nillable, Restricted picklist **Description** The previous value of the StatusCategory field on the FulfillmentOrder. Possible values are: • `Activated` • `Cancelled` • `Closed` • `Draft` • `Fulfilling` • `Rejected`
`OrderSummaryId`	**Type** reference **Properties** Nillable **Description** ID of the OrderSummary associated with the FulfillmentOrder.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

FulfillOrdItemQtyChgEvent

Notifies subscribers of changes to the quantity of a fulfillment order line item record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 53.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/FulfillOrdItemQtyChgEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

FulfillOrdItemQtyChgEvent is available as part of Salesforce Order Management.

Fields

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`FulfillmentOrderLineItemId`	**Type** reference **Properties** Nillable **Description** ID of the FulfillmentOrderLineItem whose quantity changed. This value is functionally required, but is nillable because fulfillment order line item records can be deleted to comply with data protection and privacy requirements.
`NewQuantity`	**Type** double

Field	Details
	Properties None **Description** Required. The new value of the Quantity field on the FulfillmentOrderLineItem.
`OldQuantity`	**Type** double **Properties** None **Description** The previous value of the Quantity field on the FulfillmentOrderLineItem.
`OrderItemSummaryId`	**Type** reference **Properties** Nillable **Description** ID of the OrderItemSummary associated with the FulfillmentOrderLineItem.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

InvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/billing/invoices` request is complete. The process groups billing schedules by grouping keys and creates one invoice per grouping key. `InvoiceProcessedEvent` is a top-level object that contains a list of `InvoiceProcessedDetailEvents`, where each detail event represents an attempt to create one invoice. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/InvoiceProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`InvoiceErrorDetailEvent`	**Type** InvoiceErrorDetailEvent[] on page 224 **Properties** Nillable

Field	Details
	Description Contains information about errors that occurred during processing.
`InvoiceProcessedDetailEvent`	**Type** InvoiceProcessedDetailEvent[] on page 225 **Properties** Nillable **Description** A list of InvoiceProcessedDetailEvent records. Each record contains information about an attempt to create an invoice from one or more billing schedules that share a grouping key.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the Create Order from Invoice action was successful. The default value is `false`.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/commerce/billing/invoices` response. Use this ID to identify the event for a specific request.

IN THIS SECTION:

InvoiceErrorDetailEvent

Contains information about errors that occurred during the processing of a `/commerce/billing/invoices` request. This object is included in an `InvoiceProcessedEvent` message. You can't subscribe to `InvoiceErrorDetailEvent` directly. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** None **Description** Reference code for the type of error that occurred.
`ErrorMessage`	**Type** string **Properties** None **Description** Information about the error that occurred during processing.
`ErrorSourceId`	**Type** reference

Field	Details
	Properties Nillable **Description** The ID of the record where the error occurred. Can be an invoice or a billing schedule. This field is a polymorphic relationship field. **Relationship Name** ErrorSource **Relationship Type** Lookup **Refers To** Invoice
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

InvoiceProcessedDetailEvent

Notifies subscribers of the results of an attempt to create an invoice from billing schedules as part of `/commerce/billing/invoices`. InvoiceProcessedDetailEvent contains the results of an attempt to create an invoice from one or more billing schedules that share a grouping key. Each InvoiceProcessedDetailEvent for an action is grouped within the parent object InvoiceProcessedEvent. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`EventUuid`	**Type** string

Field	Details
	Properties Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`InvoiceErrorDetailEvents`	**Type** InvoiceErrorDetailEvent[] **Properties** Nillable **Description** A list of errors that occurred while attempting to create the invoice.
`InvoiceId`	**Type** reference **Properties** Nillable **Description** The ID of the new invoice. This field is a relationship field that refers to an invoice. **Relationship Name** Invoice **Relationship Type** Lookup **Refers To** Invoice
`InvoiceStatus`	**Type** string **Properties** Nillable **Description** The value of the Status field on the invoice.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the invoice creation attempt was successful. The default value is `false`.

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

NegInvcLineProcessedEvent

Notifies subscribers when a negative invoice line is converted to a credit memo. This object is available in API version 56.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/NegInvcLineProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`CrMemoProcessErrDtlEvents`	**Type** CrMemoProcessErrDtlEvent **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available only in API versions 55.0–58.0. See the `ErrorDetails` field for error messages and error codes.
`CreditMemoId`	**Type** reference **Properties** Nillable **Description** The ID of the credit memo created as a result of the successful conversion of a negative invoice line. This field is a relationship field. **Relationship Name** CreditMemo **Relationship Type** Lookup **Refers To** CreditMemo
`ErrorDetails`	**Type** string **Properties** Nillable **Description** If the request fails, this field shows error messages, error codes, and the ID of the record on which the errors occurred. This field is available in API 58.0 and later.

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`InvoiceId`	**Type** reference **Properties** Nillable **Description** ID of the invoice that this event is in reference to. This field is a relationship field. **Relationship Name** Invoice **Relationship Type** Lookup **Refers To** Invoice
`IsAutomatedNegativeInvoiceLineConversion`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether this event is generated either by an automated process to convert negative invoice lines to credit memos or by a manual process. If `true`, the event was generated by an automatic process. If `false`, the event was generated by a manual process.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates that the negative invoice lines were converted successfully to credit memos. The default value is `false`.
`ReplayId`	**Type** string

Field	Details
	Properties Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** This field is always empty.

Example: A user successfully submits a negative invoice line, which creates one credit memo, and generates this platform event when the request completes.

```
{
  CrMemoProcessErrDtlEvents: null
  CreatedById: "005xx000001X8efAAC"
  CreatedDate: "2022-08-11T16:44:34.652Z"
  CreditMemoId: "50gxx000000gwFyAAI"
  ErrorDetails:."[]"
..InvoiceId: "3ttxx0000001Vg5AAE"
  IsSuccess: true
  RequestIdentifier: null
```

OmniTrackingEvent

Notifies subscribers about a user interaction with a FlexCard or OmniScript that's tracked for OmniAnalytics. This object is available in API version 60.0 and later.

Note: This platform event is part of OmniStudio Standard, not OmniStudio for Vlocity.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔

Subscriber	Supported?
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/OmniTrackingEvent`

Special Access Rules

Using OmniAnalytics requires having an OmniStudio license and enabling OmniAnalytics in Setup.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`ActionContainerName`	**Type** string **Properties** Nillable **Description** The full name of the FlexCard or OmniScript for which user interactions are tracked.
`ComponentType`	**Type** picklist **Properties** Restricted picklist **Description** Required. The type of component for which user interactions are tracked. Possible values are: • `Flexcard`—A context-sensitive display of Salesforce data and clickable actions. • `Omniscript`—A multi-page wizard that guides a user through a business process.
`EventName`	**Type** string

Field	Details
	Properties None **Description** Required. The name of the user interaction or user inferface response, such as `Card Load`, `Card Unload`, or `UI Action`.
`EventPayload`	**Type** textarea **Properties** None **Description** Required. The request payload sent for the user interaction, which typically includes data from the FlexCard or OmniScript.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`Timestamp`	**Type** dateTime **Properties** None **Description** Required. The timestamp indicating when the event occurred.
`TrackingCategory`	**Type** string **Properties** None

Field	Details
	Description Required. A category for this event and other events with a similar business purpose.
`TrackingGroup`	**Type** string **Properties** None **Description** Required. The name of the related OmniTrackingGroup object.

OrderStatusChangedEvent

Notifies subscribers of changes to the status of an order record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 51.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/OrderStatusChangedEvent`

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`NewStatus`	**Type** picklist **Properties** Restricted picklist **Description** The order's status after the status change. Possible values are: • `Activated` • `Draft`
`NewStatusCode`	**Type** picklist **Properties** Restricted picklist **Description** The order `StatusCode` after the status change. Possible values are: • `Activated` • `Canceled` • `Draft` • `Expired`
`OldStatus`	**Type** picklist **Properties** Restricted picklist **Description** The order's status before the status change. Possible values are: • `Activated`

Field	Details
	• `Draft`
`OldStatusCode`	**Type** picklist **Properties** Restricted picklist **Description** The order `StatusCode` before the status change. Possible values are: • `Activated` • `Canceled` • `Draft` • `Expired`
`OrderId`	**Type** reference **Properties** Nillable **Description** ID of the order whose status was changed. Used only if the order is an Original Order.
`RelatedOrderId`	**Type** reference **Properties** Nillable **Description** ID of the order whose status was changed. Used only if the order isn't an Original Order.
`RelatedOrderType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of related order. Shown only if the order with the changed status isn't an OriginalOrder. Possible values are: • `Change Order` • `Reduction Order`
`ReplayId`	**Type** string

Field	Details
	Properties Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

Usage

To use OrderStatusChangedEvent, Enable Order Events must be enabled in the Order Settings page.

When an order is created and activated in one transaction, `OldStatus` is `Draft` and `NewStatus` is `Activated`.

When an order's status is updated multiple times in one transaction, `OldStatus` is the status at the beginning of the transaction before any changes. `NewStatus` is the final status after all updates.

OrderSummaryCreatedEvent

Notifies subscribers of the creation of an order summary record. Use this event to trigger flows and processes in your order workflow. This object is available in API version 48.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/OrderSummaryCreatedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

OrderSummaryCreatedEvent is available as part of Salesforce Order Management.

Fields

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`OrderId`	**Type** reference **Properties** Nillable **Description** ID of the original order associated with the created OrderSummary.
`OrderSummaryId`	**Type** reference **Properties** Nillable **Description** ID of the created OrderSummary
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

OrderSumStatusChangedEvent

Notifies subscribers of changes to the status of an order summary record. Use this event to trigger subscribers such as flows in your order workflow. This object is available in API version 48.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/OrderSumStatusChangedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

OrderSumStatusChangedEvent is available as part of Salesforce Order Management.

Fields

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`NewStatus`	**Type** picklist **Properties** Defaulted on create, Nillable **Description** Required. The new value of the Status field on the OrderSummary.

Field	Details
	Possible values are based on the OrderSummary statuses defined in your org. The default value is `Created`.
`OldStatus`	**Type** picklist **Properties** Defaulted on create, Nillable **Description** Required. The previous value of the Status field on the OrderSummary. Possible values are based on the OrderSummary statuses defined in your org. The default value is `Created`.
`OrderId`	**Type** reference **Properties** Nillable **Description** ID of the original order associated with the OrderSummary. This field is a relationship field. **Relationship Name** Order **Relationship Type** Lookup **Refers To** Order
`OrderSummaryId`	**Type** reference **Properties** Nillable **Description** The ID of the OrderSummary that changed. This value is functionally required, but is nillable because order summary records can be deleted to comply with data protection and privacy requirements. This field is a relationship field. **Relationship Name** OrderSummary **Relationship Type** Lookup **Refers To** OrderSummary

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

PaymentCreationEvent

Notifies subscribers when the process started by the `/actions/standard/paymentSale` request is complete. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/PaymentCreationEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

To access Commerce Payments entities, your org must have a Salesforce Order Management license with the Payment Platform org permission activated. Commerce Payments entities are available only in Lightning Experience.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`ErrorCode`	**Type** string **Properties** Nillable **Description** Error code sent from the payment gateway after a request encountered an error.
`ErrorMessage`	**Type** textarea **Properties** Nillable **Description** Message sent from the payment gateway after a request encountered an error.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the request was successful. The default value is 'false'.
`PaymentGatewayLogId`	**Type** reference **Properties** Nillable

Field	Details
	Description The payment gateway log containing information about the communication with the payment gateway. This is a relationship field. **Relationship Name** PaymentGatewayLog **Relationship Type** Lookup **Refers To** PaymentGatewayLog
`PaymentId`	**Type** reference **Properties** Nillable **Description** The payment created as the result of a successful request. This is a relationship field. **Relationship Name** Payment **Relationship Type** Lookup **Refers To** Payment
`PaymentStatus`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The status of the payment created after a successful request. This field reflects the status upon payment creation, and isn't updated after further changes to the payment's status. Possible values are: • `Canceled` • `Draft` • `Failed` • `Pending` • `Processed`

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/actions/standard/paymentSale` response. Use this ID to identify the event for a specific request.
`Type`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Indicates whether the payment was made for a payment capture request or payment sale request. Possible values are: • `Capture` • `Sale`

PendingOrdSumProcEvent

Notifies subscribers that a PendingOrderSummary record was processed. If the process succeeded, an OrderSummary was created and the PendingOrderSummary can be deleted. Use this event to trigger subscribers such as flows in your order workflow. This object is available in API version 56.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/PendingOrdSumProcEvent`

Special Access Rules

PendingOrdSumProcEvent is available as part of Salesforce Order Management with the High Scale Orders feature.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** Nillable **Description** If the OrderSummary creation returned an error, this field contains the error code.
`ErrorMessage`	**Type** string **Properties** Nillable **Description** If the OrderSummary creation returned an error, this field contains the error message.
`EventUuid`	**Type** string **Properties** Nillable

Field	Details
	Description A universally unique identifier (UUID) that identifies a platform event message.
`ExternalReferenceIdentifier`	**Type** picklist **Properties** Defaulted on create **Description** Unique identifier copied from the PendingOrderSummary to the OrderSummary. This value is set to *`B2C realm ID`* + "_" + *`B2C instance ID`* + "@" + *`B2C Commerce catalog/domain ID`* + "@" + *`B2C Commerce order number`*.
`IsSuccess`	**Type** reference **Properties** Nillable **Description** Indicates whether the OrderSummary was created.
`OrderSummaryId`	**Type** reference **Properties** Nillable **Description** The ID of the OrderSummary that was created from the PendingOrderSummary.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

PlatformStatusAlertEvent

Notifies subscribers of alerts that occur during the processing of a user request or service job execution. This object is available in API version 45.0 and later.

For example, suppose that a formula is evaluated as part of processing user requests. A platform event message can be generated during the processing of a user request when an error is encountered from evaluating an invalid formula.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/PlatformStatusAlertEvent`

Special Access Rules

Accessing this object requires the Customize Application, Modify All Data, or Manage Next Best Action Strategies user permission.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`ApiErrorCode`	**Type** string **Properties** Nillable **Description** The API error code.
`ComponentName`	**Type** string **Properties** Nillable **Description** Name of the component in which the alert occurred.

Field	Details
`EventDate`	**Type** datetime **Properties** Nillable **Description** Date and time when the event occurred. Example: 2018-12-18 21:59:48
`EventIdentifier`	**Type** string **Properties** Nillable **Description** Unique identifier of the event. This field is reserved for future use and is always null in API version 45.0.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ExtendErrorCode`	**Type** string **Properties** Nillable **Description** Extended error code which provides more details about the issue.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** EventIdentifier (uuid) of the related event. This field is reserved for future use and is always null in API version 45.0.
`ReplayId`	**Type** string **Properties** Nillable

Field	Details
	Description Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestId`	**Type** string **Properties** Nillable **Description** The unique ID of the service job that fired the event. It can be used to correlate the alert with logging information.
`ServiceJobId`	**Type** string **Properties** Nillable **Description** Service-specific job ID, if one exists. For Next Best Action, the service job ID is `executionToken`. This field can be used to correlate the alert with logging information.
`ServiceName`	**Type** string **Properties** Nillable **Description** Name of the service that triggered the alert.
`StatusType`	**Type** string **Properties** Nillable **Description** Status of the event.
`SubComponentName`	**Type** string **Properties** Nillable **Description** Name of the subcomponent where the alert occurs.

Field	Details
`Subject`	**Type** string **Properties** Nillable **Description** Short description of the alert.
`UserId`	**Type** reference **Properties** Nillable **Description** ID of the user who caused the event.
`Username`	**Type** string **Properties** Nillable **Description** Username of the user who caused the event.

Usage

The following example shows how to process platform status alert events. Only internal services can publish these events. This Apex trigger example fires when a platform event message is published and creates a Chatter post on the admin profile with event details.

```
trigger PlatformStatusAlertEventTrigger on PlatformStatusAlertEvent (after insert) {
    List<Feeditem> posts = new List<Feeditem>();
    Id profileId = [select Id from User where User.Profile.Name = 'System Administrator'
limit 1].Id;

    for(PlatformStatusAlertEvent e : trigger.new) {
       Feeditem post = New Feeditem();
       post.ParentId= profileId;
       post.Body = 'Alert occured in the service: ' + e.ServiceName + '\n' +
            'APIErrorCode: ' + e.APIErrorCode + '\n' +
            'ComponentName: ' + e.ComponentName + '\n' +
            'EventDate: ' + e.EventDate + '\n'+
            'EventIdentifier: ' + e.EventIdentifier + '\n' +
            'ExtendedErrorCode: '+ e.ExtendedErrorCode + '\n' +
            'RelatedEventIdentifier: ' + e.RelatedEventIdentifier + '\n' +
            'ReplayId: ' + e.ReplayId + '\n' +
            'RequestId: ' + e.RequestId + '\n' +
            'ServiceJobId: ' + e.ServiceJobId + '\n' +
            'ServiceName: ' + e.ServiceName + '\n'+
            'StatusType: ' + e.StatusType + '\n' +
```

```
            'SubComponentName: ' + e.SubComponentName + '\n' +
            'Subject: '+ e.Subject + '\n' +
            'UserId: ' + e.UserId + '\n' +
            'Username: ' + e.Username + '\n';
        posts.add(post);
    }
    if (posts.size() > 0) {
        insert(posts);
    }
}
```

Example: The code example ultimately displays as a Chatter post that contains the following:

Alert occurred in the service: Next Best Action Strategy

APIErrorCode: INVALID_OPERATION

ComponentName: Strategy_for_error_event_demo

EventDate: 2018-12-18 21:59:48

EventIdentifier: null

ExtendedErrorCode: FORMULA_EXPRESSION_INVALID

RelatedEventIdentifier: null

ReplayId: 63

RequestId: TID:89715900005e40b69a

ServiceJobId: 1014fd4e-4a19-4910-be36-377a7f2f1b75

ServiceName: Next Best Action Strategy

StatusType: Error

SubComponentName: filter_node1

Subject: Something went wrong with filter element 'filter_node1': 'Unknown function ISBLANC. Check spelling.'

UserId: 005RM000001ZnzAYAS

Username: xxx@yyy.com

ProcessExceptionEvent

Notifies subscribers of errors that occur during payment processing (capture, apply, and refund) on an order summary. Use this event to trigger subscribers such as flows in your order workflow. This object is available in API version 50.0 and later.

Supported Calls

`create()`, `describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔

Subscriber	Supported?
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/ProcessExceptionEvent`

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`AttachedToId`	**Type** reference **Properties** Create **Description** ID of the object associated with the ProcessException. This field is a polymorphic relationship field. **Relationship Name** AttachedTo **Relationship Type** Lookup **Refers To** CreditMemo, FulfillmentOrder, Invoice, Order, OrderItem, OrderItemSummary, OrderPaymentSummary, OrderSummary, Payment, PaymentAuthorization, Refund, ReturnOrder, WebCart, WebStore
`BackgroundOperationId`	**Type** reference **Properties** Create, Nillable **Description** The operation where the exception occurred. This field is a relationship field. **Relationship Name** BackgroundOperation

Field	Details
	Relationship Type Lookup **Refers To** BackgroundOperation
`Description`	**Type** textarea **Properties** Create, Nillable **Description** Detailed description of the ProcessException.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ExceptionType`	**Type** picklist **Properties** Create **Description** Process type that caused the exception. Possible values are: • `Commerce Inventory Delete Reservation Failed` This value is reserved for future use. • `Commerce Inventory Update Reservation Failed` This value is reserved for future use. • `OM Apply Failed` • `OM Capture Failed` • `OM Refund Failed` • `OM RMA Failed` This value is available in API v52.0 and later. • `Place Order Failed` This value is available in API v57.0 and later.
`ExternalReference`	**Type** string **Properties** Create, Nillable

Field	Details
	Description Description of external entities associated with the ProcessException.
`Message`	**Type** string **Properties** Create **Description** Short description of the ProcessException
`OrderSummaryId`	**Type** reference **Properties** Create, Nillable **Description** ID of the OrderSummary associated with the ProcessException. The ProcessException component is displayed on this OrderSummary. This field is a relationship field. **Relationship Name** OrderSummary **Relationship Type** Lookup **Refers To** OrderSummary
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`Severity`	**Type** picklist **Properties** Create, Defaulted on create, Nillable

Field	Details
	Description Severity of the ProcessException. Each severity value corresponds to one severity category. You can customize the severity picklist to represent your business processes. If you customize the severity picklist, include at least one severity value for each severity category. Severity is set to Null when creating events for payment failures. Possible values are: • `High` • `Low` • `Null` The default value is `High`.

QuoteSaveEvent

Notifies subscribers that the process started by the `/actions/standard/quotesaveevent` request is complete. If the process is successful, use this event to learn about the updated quote. If the request isn't successful, use this event to learn about the errors and how to fix them. This object is available in API version 58.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Streaming API (CometD)	✔

Streaming API Subscription Channel

`/event/QuoteSaveEvent`

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`QuoteId`	**Type** reference **Properties** Nillable **Description** The ID of the quote associated with this event. This field is a relationship field. **Relationship Name** Quote **Relationship Type** Lookup **Refers To** Quote
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** The unique ID returned in the `/quotesaveevent` response. Use this ID to identify the event for the specific request.
`Status`	**Type** picklist **Properties** Defaulted on create, Nillable, Restricted picklist **Description** The default value is `NotStarted`. Possible values are: • `CompletedWithPricing` • `CompletedWithTax` • `CompletedWithoutPricing` • `NotStarted` • `PriceCalculationFailed` • `PriceCalculationInProgress` • `PriceCalculationQueued` • `SaveFailedOrIncomplete` • `Saving` • `TaxCalculationFailed` • `TaxCalculationInProgress` • `TaxCalculationSuccess` • `TaxCalculationWaiting`

QuoteToOrderCompletedEvent

Notifies subscribers when the `/actions/standard/createOrderFromQuote` REST request is complete. If the request is successful, use this event to learn about the Order record. If the request isn't successful, use this event to learn about the errors associated with the request. This object is available in API version 56.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/QuoteToOrderCompletedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available with Subscription Management.

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`HasErrors`	**Type** boolean **Properties** Defaulted on create

Field	Details
	Description Contains `true` if errors occurred during the process; otherwise `false`. The default value is `false`.
`OrderId`	**Type** string **Properties** Nillable **Description** The ID of the order created from the quote. If the process failed, this field is null.
`QuoteToOrderErrorDetailEvents`	**Type** QuoteToOrderErrDtlEvent[] **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `actions/standard/createOrderFromQuote` response. Use this ID to identify the event for a specific request.

IN THIS SECTION:

QuoteToOrderErrDtlEvent

Contains information about any errors that occurred while processing the `/actions/standard/createOrderFromQuote` REST request. One `QuoteToOrderErrDtlEvent` record is created for each error that occurred. This object is included in an `QuoteToOrderCompletedEvent` message. You can't subscribe to `QuoteToOrderErrDtlEvent` directly. This object is available in API version 56.0 and later.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** Nillable **Description** The error code; for example, INVALID_INPUT.
`ErrorMessage`	**Type** textarea **Properties** Filter, Group, Nillable, Sort **Description** Information about the error that occurred during processing.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

Field	Details
`PrimaryRecordId`	**Type** reference **Properties** **Description** The record on which the error occurred; for example, the order that was created by the request. **Relationship Name** PrimaryRecord **Relationship Type** Lookup **Refers To** Asset, Invoice, Order
`RelatedRecordId`	**Type** reference **Properties** Nillable **Description** Optional. A secondary record on which the error occurred; for example, the order item. **Relationship Name** RelatedRecord **Relationship Type** Lookup **Refers To** InvoiceLine, OrderItem
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

RealtimeAlertEvent

Notifies subscribers of Amazon CloudWatch alarm events from your Service Cloud Voice Amazon Connect instance. This object is available in API version 54.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/RealtimeAlertEvent`

Special Access Rules

Accessing this object requires Service Cloud Voice with Amazon Connect enabled.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`Description`	**Type** string **Properties** None **Description** Required. Description of the alert fired.
`EventDateTime`	**Type** datetime **Properties** None **Description** Date and time when the alert occurred. For example: `2020-12-18 21:59:48`.

Field	Details
`Name`	**Type** string **Properties** None **Description** Required. Name of the alert fired.
`Payload`	**Type** string **Properties** None **Description** More information and data associated with the alert.
`Severity`	**Type** string **Properties** None **Description** Required. Severity of the triggered alarm. Possible values: • Critical • Warning • Information When using Amazon CloudWatch with Service Cloud Voice, the following alarm states map to severity values: • ALARM maps to Critical • INSUFFICIENT_DATA maps to Warning • OK maps to Information
`Source`	**Type** string **Properties** None **Description** Required. The source of the alert. For example, this value can be the service name: `AWS Cloudwatch`.

Usage

The following example shows how to process RealtimeAlert events. This Apex trigger example fires when a platform event message is published and creates a Chatter post on the admin profile with event details.

```
trigger RealtimeAlertEventTrigger on RealtimeAlertEvent (after insert) {
   Id profileId = [select Id from User where User.Profile.Name = 'System Administrator'
limit 1].Id;
    for(RealtimeAlertEvent e : trigger.new) {
       Feeditem Post = New Feeditem();
       Post.ParentId= profileId;
       Post.Body = 'Alert occurred in the service: ' + e.Source + '\n' +
            'Name: ' + e.Description + '\n' +
            'Severity: ' + e.EventDateTime + '\n' +
            'Payload: ' + e.Payload + '\n' +
            'EventDate: ' + e.EventDateTime + '\n' +
            'Description: ' + e.Name + '\n';
       insert Post;
    }
}
```

Example: The code example displays as a Chatter post that contains the following:

```
Alert occurred in the service: aws cloudwatch alarm
Name: alert description
Severity: 2021-11-05 11:11:11
Payload: payload
EventDate: 2021-11-05 11:11:11
Description: alert name
```

RemoteKeyCalloutEvent

Notifies subscribers of callouts that fetch encrypted key material from a customer endpoint. This object is available in API versions 45.0 and later.

The `RemoteKeyCalloutEvent` captures events related to the success or failure of a callout that fetches encrypted key material from an end point. Based on the Platform Events framework, a `RemoteKeyCalloutEvent` is published every time a callout is made to an external key service. This event lets you monitor your cache-only key callouts in real time, and receive alerts about any errors that might occur. You can subscribe to events with after insert Apex triggers and store events in custom objects, security information event management (SIEM), or other back-end systems.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	
Processes	

Subscriber	Supported?
Pub/Sub API	
Streaming API (CometD)	✔

Subscription Channel

`/event/RemoteKeyCalloutEvent`

Special Access Rules

Access to `RemoteKeyCalloutEvent` data requires purchasing Salesforce Shield or Shield Platform Encryption. The `RemoteKeyCalloutEvent` only applies to callouts that fetch cache-only key material.

Event Delivery Allocation Enforced

Yes

Fields

Field	Details
`Details`	**Type** textarea **Properties** Nillable **Description** A JSON representation with more information about the `StatusCode`. Not all status codes (for example, SUCCESS) show a populated Details field. Populated Details fields include key-value pairs that you can use to make Apex triggers and other programmatic assertions.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ReplayID`	**Type** string **Properties** Nillable

Field	Details
	Description Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** When Replay Detection for Cache-Only Keys is enabled, a unique marker automatically generated and sent with every callout. This marker includes the key identifier, a nonce generated for that callout instance, and the nonce required from the endpoint. Available in API version 45.0 and later.
`StatusCode`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** A code that characterizes the error. The full list of status codes is available in the WSDL file for your org.
`TenantSecretID`	**Type** reference **Properties** Nillable **Description** The record ID of the tenant secret associated with the published event.

Usage

To view a `RemoteKeyCalloutEvent` and perform custom actions after your callout, create an after insert Apex trigger in Dev Console. These triggers let you assign custom actions for your event. You can set in-app alerts and send email alerts to people who maintain your key service, including users who don't have a Salesforce login.

For longer-term monitoring, you can store `RemoteKeyCalloutEvent` data in custom objects and custom fields, SIEM, or other back-end systems. Then use business rules to send alerts. For example, you can set an alert that sends admins an email when something is wrong with a key service.

Here's an example of an after insert trigger that stores `RemoteKeyCalloutEvent` results in a custom object called Key Service Callout Log. The custom object also draws data from the `TenantSecret` object.

Table 2: Sample Custom Object: Key Service Callout Log

Field Label	Field Name	Data Type
Key Service Callout Log ID	Name	Auto Number
Details	Details__c	Text(255)
Replay Detection	Replay_Detection__c	Text (255)
Status Code	Status_Code__c	Text(255)
Tenant Secret Id	Tenant_Secret_Id__c	Text(50)
Tenant Secret Status	Tenant_Secret_Status__c	Text(255)
Type	Type__c	Text(10)
Version	Version__c	Number(10,0)

If you use this trigger sample, adjust the field API names to suit your needs.

```
trigger RemoteKeyCalloutEvent on RemoteKeyCalloutEvent (after insert){
    List<Key_Service_Callout_Log__c> l = new List<Key_Service_Callout_Log__c>();
    Set<ID> TenantSecretIds = new Set<ID>();
    Map<ID, TenantSecret> TenantSecrets;
    for(RemoteKeyCalloutEvent event : Trigger.new){
       if(event.TenantSecretId != null && !TenantSecretIds.contains(event.TenantSecretId))

            TenantSecretIds.add(event.TenantSecretId);
    }
    if(TenantSecretIds != null && !TenantSecretIds.isEmpty())
      TenantSecrets = new Map<ID, TenantSecret>([SELECT Type, Version, Status FROM
TenantSecret where Id In: TenantSecretIds]);

    for(RemoteKeyCalloutEvent event : Trigger.new){
        Key_Service_Callout_Log__c log = new Key_Service_Callout_Log__c();
      log.Status_Code__c = event.StatusCode;
        log.Tenant_Secret_ID__c = event.TenantSecretId;
          log.Replay_Detection__c = event.RequestIdentifier;
      log.Details__c = event.Details;
        if(TenantSecrets != null && TenantSecrets.containsKey(event.TenantSecretId)){
            log.Type__c = TenantSecrets.get(event.TenantSecretId).Type;
            log.Version__c = TenantSecrets.get(event.TenantSecretId).Version;
            log.Tenant_Secret_Status__c = TenantSecrets.get(event.TenantSecretId).Status;

        }
        l.add(log);
    }

    insert l;
}
```

To troubleshoot callout errors, review the `StatusCode` and `Details` fields. These fields give you information about remote key callout errors or exceptions in raw JSON format. Successful, empty callout, and timeout responses return empty `Details` fields.

Table 3: Cache-Only Key Service Errors and Status Codes

RemoteKeyCalloutEvent Status Code	Error	Tips for Fixing the Problem
DESTROY_HTTP_CODE	The remote key service returned an HTTP error: {000}. A successful HTTP response returns a 200 code.	To find out what went wrong, review the HTTP response code.
ERROR_HTTP_CODE	The remote key service returned an unsupported HTTP response code: {000}. A successful HTTP response returns a 200 code.	To find out what went wrong, review the HTTP response code.
MALFORMED_CONTENT_ENCRYPTION_KEY	The remote key service returned a content encryption key in the JWE that couldn't be decrypted with the certificate's private key. Either the JWE is corrupted, or the content encryption key is encrypted with a different key.	Check that you set up your named credential properly and are using the correct BYOK-compatible certificate.
MALFORMED_DATA_ENCRYPTION_KEY	The content encryption key couldn't decrypt the data encryption key that was returned in the remote key service's JWE. The data encryption key is either malformed, or encrypted with a different content encryption key.	Check that you set up your named credential properly and are using the correct BYOK-compatible certificate. Named credentials must call out to an HTTPS endpoint.
MALFORMED_JSON_RESPONSE	We can't parse the JSON returned by your remote key service. Contact your remote key service for help.	Contact your remote key service.
MALFORMED_JWE_RESPONSE	The remote key service returned a malformed JWE token that can't be decoded. Contact your remote key service for help.	Contact your remote key service.
EMPTY_RESPONSE	The remote key service callout returned an empty response. Contact your remote key service for help.	Contact your remote key service.
RESPONSE_TIMEOUT	The remote key service callout took too long and timed out. Try again.	If your key service is unavailable after multiple callout attempts, contact your remote key service.
UNKNOWN_ERROR	The remote key service callout failed and returned an error: {000}.	Contact your remote key service.
INCORRECT_KEYID_IN_JSON	The remote key service returned JSON with an incorrect key ID. Expected: {valid keyID}. Actual: {invalid keyID}.	Check that you set up your named credential properly and are using the correct BYOK-compatible certificate.
INCORRECT_KEYID_IN_JWE_HEADER	The remote key service returned a JWE header with an incorrect key ID. Expected: {valid keyID}. Actual: {invalid keyID}.	Check that you set up your named credential properly and are using the correct BYOK-compatible certificate.

RemoteKeyCalloutEvent Status Code	Error	Tips for Fixing the Problem
INCORRECT_ALGORITHM_IN_JWE_HEADER	The remote key service returned a JWE header that specified an unsupported algorithm (alg): {algorithm}.	The algorithm for encrypting the content encryption key in your JWE header must be in RSA-OAEP format.
INCORRECT_ENCRYPTION_ALGORITHM_IN_JWE_HEADER	The remote key service returned a JWE header that specified an unsupported encryption algorithm (enc): {your enc}.	The algorithm for encrypting the data encryption key in your JWE header must be in A256GCM format.
INCORRECT_DATA_ENCRYPTION_KEY_SIZE	Data encryption keys encoded in a JWE must be 32 bytes. Yours is {value} bytes.	Make sure that your data encryption key is 32 bytes.
ILLEGAL_PARAMETERS_IN_JWE_HEADER	Your JWE header must use {0}, but no others. Found: {1}.	Remove the unsupported parameters from your JWE header.
MISSING_PARAMETERS_IN_JWE_HEADER	Your JWE header is missing one or more parameters. Required: {0}. Found:{1}.	Make sure that your JWE header includes all required values. For example, if Replay Detection is enabled, the JWE header must include the nonce value extracted from the cache-only key callout.
AUTHENTICATION_FAILURE_RESPONSE	Authentication with the remote key service failed with the following error: {error}.	Check the authentication settings for your chosen named credential.
POTENTIAL_REPLAY_ATTACK_DETECTED	The remote key service returned a JWE header with an incorrect nonce value. Expected: {0}. Actual: {1}	Make sure that your JWE header includes the RequestID included in the callout.
UNKNOWN_ERROR	The remote key service callout failed and returned an error: java.security.cert.CertificateExpiredException: NotAfter: {date and time of expiration}	The certificate for your cache-only key expired. Update your cache-only key material to use an active BYOK-compatible certificate.

SEE ALSO:

Apex Developer Guide: Triggers

Apex Developer Guide: Add an Apex Trigger

SOAP API Developer Guide: Custom Objects

Salesforce Help: Cache-Only Key Service

ServiceAppointmentEvent

Notifies subscribers of the service appointment details that are generated from the event platform. This object is available in API version 59.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/ServiceAppointmentEvent`

Special Access Rules

This object is available when Salesforce Scheduler is enabled.

Fields

Field	Details
`AsgnRsrcApptSchdDtlEvent`	**Type** AsgnRsrcApptSchdEvent[] **Properties** Nillable **Description** One or multiple assigned resource records related to the scheduler appointment event.
`ChangeType`	**Type** string **Properties** Nillable **Description** The operation that caused the change. For example: CREATE, UPDATE, DELETE.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`ServiceApptSchduleEvent`	**Type** SvcApptSchdEvent[] **Properties** Nillable **Description** The service appointment related to the scheduler appointment event.

VoidInvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/invoices/{invoiceId}/actions/void` request is complete. The request attempts to void an invoice by crediting an invoice and changing its status to Voided, which prevents further changes. This object is available in API version 55.0 and later. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/VoidInvoiceProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`CrMemoProcessErrDtlEvents`	**Type** CrMemoProcessErrDtlEvent[] **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available only in API versions 55.0–58.0. See the `ErrorDetails` field for error messages and error codes.
`CreditMemoId`	**Type** reference **Properties** Nillable **Description** The credit memo created to void the invoice as the result of a successful request. This field is a relationship field. **Relationship Name** CreditMemo **Relationship Type** Lookup **Refers To** CreditMemo
`ErrorDetails`	**Type** string

Field	Details
	Properties Nillable **Description** If the request fails, this field shows error messages, error codes, and the ID of the record on which the errors occurred. This field is available in API 58.0 and later.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`InvoiceId`	**Type** reference **Properties** Nillable **Description** The invoice that was voided as the result of a successful request. This field is a relationship field. **Relationship Name** Invoice **Relationship Type** Lookup **Refers To** Invoice
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the request was successful. The default value is 'false'.
`ReplayId`	**Type** string **Properties** Nillable

Field	Details
	Description Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/commerce/billing/invoices/{invoiceId}/actions/void` response. Use this ID to identify the event for a specific request.

WebStoreUserCreatedEvent

Notifies subscribers of the creation of a new user for a WebStore. This object is available in API version 59.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Streaming API (CometD)	✓

Streaming API Subscription Channel

`/event/WebStoreUserCreatedEvent`

Special Access Rules

WebStoreUserCreatedEvent is available only if the B2B or B2C Commerce license is enabled.

Fields

Field	Details
`ActionSource`	**Type** picklist **Properties** Restricted picklist **Description** The source of the published event. Possible values are: • `InviteToReorderPortal` • `Others`
`AddedById`	**Type** reference **Properties** Nillable **Description** The ID of the user who invited the new user to the WebStore. This field is a relationship field. **Relationship Name** AddedBy **Relationship Type** Lookup **Refers To** User
`AddedUserId`	**Type** reference **Properties** Nillable **Description** The ID of the user created for the WebStore. This field is a relationship field. **Relationship Name** AddedUser **Relationship Type** Lookup **Refers To** User

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`WebStoreId`	**Type** reference **Properties** Nillable **Description** The ID of the WebStore. This field is a relationship field. **Relationship Name** WebStore **Relationship Type** Lookup **Refers To** WebStore

Real-Time Event Monitoring Objects

Check out the standard platform event and object pairs for Real-Time Event Monitoring. For most platform events used in Real-Time Event Monitoring, corresponding objects store the event data. For more information, see Real-Time Event Monitoring in Salesforce Help.

Note: Real-Time Event Monitoring objects sometimes contain sensitive data. Assign object permissions to Real-Time Events accordingly in profiles or permission sets.

Platform Event	Object for Event Storage	Can Be Used in a Transaction Security Policy?
ApiAnomalyEvent	ApiAnomalyEventStore	✓

Platform Event	Object for Event Storage	Can Be Used in a Transaction Security Policy?
ApiEventStream	ApiEvent	✓
BulkApiResultEvent	BulkApiResultEventStore	✓
ConcurLongRunApexErrEvent	Not Available	
CredentialStuffingEvent	CredentialStuffingEventStore	✓
FileEvent	FileEventStore	✓
GuestUserAnomalyEvent	GuestUserAnomalyEventStore	✓
Not Available	IdentityVerificationEvent	
Not Available	IdentityProviderEventStore	
LightningUriEventStream	LightningUriEvent	
ListViewEventStream	ListViewEvent	✓
LoginAsEventStream	LoginAsEvent	
LoginEventStream	LoginEvent	✓
LogoutEventStream	LogoutEvent	
MobileEmailEvent	MobileEmailEventStore	
MobileEnforcedPolicyEvent	MobileEnfPolicyEventStore	
MobileScreenshotEvent	MobileScreenshotEventStore	
MobileTelephonyEvent	MobileTelephonyEventStore	
PermissionSetEvent	PermissionSetEventStore	✓
ReportAnomalyEvent	ReportAnomalyEventStore	✓
ReportEventStream	ReportEvent	✓
SessionHijackingEvent	SessionHijackingEventStore	✓
UriEventStream	UriEvent	

Note: Real-Time Event monitoring objects that were introduced as part of the beta release in API version 46.0 follow a naming convention that is no longer used in later API versions. In particular:

- The name format of a platform event object was ***ObjectName***`EventStream`.
- The name format of the corresponding big object used for storage was ***ObjectName***`Event`.

New event objects introduced after API version 46.0 use the following standard platform event naming convention.

- The name format of a platform event object is ***ObjectName***`Event`.
- The name format of the corresponding object used for storage is ***ObjectName***`EventStore`.

ApiAnomalyEvent

Track anomalies in how users make API calls. This object is available in API version 50.0 and later.

Supported Calls

`describeSObjects()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	✓
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Event Delivery Allocation Enforced

No

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Nillable

Field	Details
	Description The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`Operation`	**Type** string **Properties** Nillable **Description** The API call that generated the event. For example, `Query`.
`PolicyId`	**Type** reference **Properties** Nillable

Field	Details
	Description The ID of the transaction policy associated with this event. For example, `0NIB000000000KOOAY`. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values include: • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The type of entities associated with the event.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** The unique ID of a single transaction. A transaction can contain one or more events. Each event in a given transaction has the same REQUEST_ID. For example, `3nWgxWbDKWWDIk0FKfF5D`.
`RowsProcessed`	**Type** double **Properties** Nillable **Description** Total row count for the current operation. For example, `2500`.
`Score`	**Type** double **Properties** Nillable **Description** A number from 0 through 100 that represents the anomaly score for the API execution or export tracked by this event. The anomaly score shows how the user's current API activity is different from their typical activity. A low score indicates that the user's current API activity is similar to their usual activity. A high score indicates that it's different.

`SecurityEventData`	**Type**
	textarea
	Properties
	Nillable
	Description
	The set of features about the API activity that triggered this anomaly event.
	Let's say, for example, that a user typically downloads 10 accounts but then they deviate from that pattern and download 1,000 accounts. This event is triggered and the contributing features are captured in this field. Potential features include row count, column count, average row size, the day of week, and the browser's user agent used for the report activity. The data captured in this field also shows how much a particular feature contributed to this anomaly event being triggered, represented as a percentage. The data is in JSON format.
	Example
	This example shows that the average row count contributed more than 95% to the anomaly being triggered. Other anomalous features, such as the autonomous system, day of the week the report was run, the browser used, and the number of columns, contributed much less.
	`[`
	`{`

Field	Details

	<pre>"featureName": "rowCount",
	"featureValue": "1937568",
	"featureContribution": "95.00 %"
	},
	{
	"featureName": "autonomousSystem",
	"featureValue": "Bigleaf Networks, Inc.",
	"featureContribution": "1.62 %"
	},
	{
	"featureName": "dayOfWeek",
	"featureValue": "Sunday",
	"featureContribution": "1.42 %"
	},
	{
	"featureName": "userAgent",
	"featureValue": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
	AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132
	Safari/537.36}",
	"featureContribution": "1.21 %"
	},
	{
	"featureName": "periodOfDay",
	"featureValue": "Evening",
	"featureContribution": ".09 %"
	},
	{
	"featureName": "averageRowSize",
	"featureValue": "744",
	"featureContribution": "0.08 %"
	},
	{
	"featureName": "screenResolution",
	"featureValue": "900x1440",
	"featureContribution": "0.07 %"
	}
	]</pre>

`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.
`SourceIp`	**Type** string

Field	Details
	Properties Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the API anomaly that caused this event to be created. **Example** • `API was exported from an infrequent network (BigLeaf Networks Inc.)` • `API was generated with an unusually high number of rows (111141)`
`Uri`	**Type** string **Properties** Nillable **Description** The URI of the page that's receiving the request.
`UserAgent`	**Type** string **Properties** Nillable **Description** UserAgent used in HTTP request, post-processed by the server.
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string

Field	Details
	Properties Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

ApiAnomalyEventStore

Tracks anomalies in how users make API calls. ApiAnomalyEventStore is an object that stores the event data of ApiAnomalyEvent. This object is available in API version 50.0 and later.

Supported Calls

`describeLayout()` `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`ApiAnomalyEventNumber`	**Type** string **Properties** Autonumber, Defaulted on create, Filter, idLookup, Sort **Description** The unique number automatically assigned to the event when it's created. You can't change the format or value for this field.
`EvaluationTime`	**Type** double **Properties** Filter, Nillable, Sort **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime

Field	Details
	Properties Filter, Sort **Description** Required. The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Group, Sort **Description** Required. The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`LastReferencedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The timestamp for when the current user last viewed a record related to this record.
`LastViewedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The timestamp for when the current user last viewed this record. If this value is null, it's possible that this record was referenced (`LastReferencedDate`) and not viewed.
`LoginKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`Operation`	**Type** string **Properties** Nillable

Field	Details
	Description The API call that generated the event. For example, `Query`.
`PolicyId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** The ID of the transaction policy associated with this event. For example, `0NIB000000000KOOAY`. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Filter, Group, Nillable, Restricted picklist, Sort **Description** The result of the transaction policy. Possible values include: • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The type of entities associated with the event.
`RequestIdentifier`	**Type** string **Properties** Nillable

Field	Details
	Description The unique ID of a single transaction. A transaction can contain one or more events. Each event in a given transaction has the same REQUEST_ID. For example, `3nWgxWbDKWWDIk0FKfF5D`.
`RowsProcessed`	**Type** double **Properties** Nillable **Description** Total row count for the current operation. For example, `2500`.
`Score`	**Type** double **Properties** Filter, Nillable, Sort **Description** A number from 0 through 100 that represents the anomaly score for the API execution or export tracked by this event. The anomaly score shows how the user's current API activity is different from their typical activity. A low score indicates that the user's current API activity is similar to their usual activity, a high score indicates that it's different.

`SecurityEventData`	**Type**
	textarea
	Properties
	Nillable
	Description
	The set of features about the API activity that triggered this anomaly event.
	Let's say, for example, that a user typically downloads 10 accounts but then they deviate from that pattern and download 1,000 accounts. This event is triggered and the contributing features are captured in this field. Potential features include row count, column count, average row size, the day of week, and the browser's user agent used for the report activity. The data captured in this field also shows how much a particular feature contributed to this anomaly event being triggered, represented as a percentage. The data is in JSON format.
	Example
	This example shows that the average row count contributed more than 95% to the anomaly being triggered. Other anomalous attributes, such as the autonomous system, day of the week the report was run, the browser used, and the number of columns, contributed much less.
	`[`
	`{`
	`"featureName": "rowCount",`
	`"featureValue": "1937568",`

Field	Details

	"featureContribution": "95.00 %"
	},
	{
	"featureName": "autonomousSystem",
	"featureValue": "Bigleaf Networks, Inc.",
	"featureContribution": "1.62 %"
	},
	{
	"featureName": "dayOfWeek",
	"featureValue": "Sunday",
	"featureContribution": "1.42 %"
	},
	{
	"featureName": "userAgent",
	"featureValue": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36}",
	"featureContribution": "1.21 %"
	},
	{
	"featureName": "periodOfDay",
	"featureValue": "Evening",
	"featureContribution": ".09 %"
	},
	{
	"featureName": "averageRowSize",
	"featureValue": "744",
	"featureContribution": "0.08 %"
	},
	{
	"featureName": "screenResolution",
	"featureValue": "900x1440",
	"featureContribution": "0.07 %"
	}
	]

`SessionKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.
`SourceIp`	**Type** string **Properties** Filter, Group, Nillable, Sort

Field	Details
	Description The source IP address of the client that logged in. For example, `126.7.4.2`.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the report anomaly that caused this event to be created. **Example** • `Report was exported from an infrequent network (BigLeaf Networks Inc.)` • `Report was generated with an unusually high number of rows (111141)`
`Uri`	**Type** string **Properties** Nillable **Description** The URI of the page that's receiving the request.
`UserAgent`	**Type** string **Properties** Nillable **Description** UserAgent used in HTTP request, post-processed by the server.
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable

Field	Details
	Description The origin username in the format of `user@company.com` at the time the event was created.

Associated Object

This object has the following associated object. It's available in the same API version as this object.

ApiAnomalyEventStoreFeed
Feed tracking is available for the object.

ApiEvent

Tracks these user-initiated read-only API calls: `query()`, `queryMore()`, and `count()`. Captures API requests through SOAP API and Bulk API for the Enterprise and Partner WSDLs. Tooling API calls and API calls originating from a Salesforce mobile app aren't captured. You can use ApiEvent in a transaction security policy. ApiEvent is a big object that stores the event data of ApiEventStream. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`AdditionalInfo`	**Type** string **Properties** Nillable **Description** JSON serialization of additional information that's captured from the `HTTP` headers during an API request. For example, `{"field1": "value1", "field2": "value2"}`. For usage tips and examples, see the Working With the AdditionalInfo Field section.
`ApiType`	**Type** string **Properties** Nillable

Field	Details
	Description The API that was used. Values include: • `Bulk` • `REST` • `SOAP Enterprise` • `SOAP Partner` • `N/A`
`ApiVersion`	**Type** double **Properties** Nillable **Description** The version number of the API.
`Application`	**Type** string **Properties** Nillable **Description** The application used to access the org. For example, CRM Analytics or Salesforce Developers Connector.
`Client`	**Type** string **Properties** Nillable **Description** The service that executed the API event. If you're using an unrecognized client, this field returns "Unknown" or a blank value.
`ConnectedAppId`	**Type** reference **Properties** Nillable **Description** The 15-character ID of the connected app associated with the API call. For example, 0H4RM00000000Kr0AI. The `ConnectedAppID` field populates when a call triggers an OAuth 2.0 authentication process, which identifies the connected app that's authorized to access Salesforce data on behalf of a user. When a user associated with the call already has an active authentication token, the `ConnectedAppID` is set to a null value.

Field	Details
`ElapsedTime`	**Type** int **Properties** Nillable **Description** The amount of time it took for the request to complete in milliseconds. The measurement of this value begins before the query executes and ends when the query completes. It doesn't include the amount of time it takes to return the result over the network.
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time when the specified API event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event. For example, 0a4779b0-0da1-4619-a373-0a36991dff90.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular series of API events. This field is also available on the LoginEvent, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. For example, 0YaB000002knVQLKA2.

Field	Details
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`Operation`	**Type** picklist **Properties** Nillable, Restricted Picklist **Description** The API call that generated the event. Possible values are `Query`, `QueryAll`, or `QueryMore`.
`Platform`	**Type** string **Properties** Nillable **Description** The operating system on the login machine. For example, iPhone, Mac OS, Linux, or Unknown.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. For this event, possible values are: • `Block` - The user was blocked from performing the operation that triggered the policy. • `Error` - The policy caused an undefined error when it executed.

Field	Details
	• `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The entities in the SOQL query. For example, Opportunity, Lead, Account, or Case. Can also include custom objects. For relationship queries, the value of this field contains all entities involved in the query. **Examples** • For `SELECT Contact.FirstName, Contact.Account.Name from Contact`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Account.Name, (SELECT Contact.FirstName, Contact.LastName FROM Account.Contacts) FROM Account`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Id, Name, Account.Name FROM Contact WHERE Account.Industry = 'media'`, the value of `QueriedEntities` is `Account, Contact`.
`Query`	**Type** string **Properties** Nillable **Description** The SOQL query. For example, `SELECT id FROM Lead`.
`Records`	**Type** json **Properties** Nillable

Field	Details
	Description A JSON string that represents the queried objects' metadata. This metadata includes the number of results of a query per entity type and the entity IDs. **Example** `{ "totalSize" : 1, "done" : true, "records" : [{ "attributes" : { "type" : "Account" }, "Id" : "001xx000003DMvCAAW", "Contacts" : { "totalSize" : 3, "done" : true, "records" : [{ "attributes" : { "type" : "Contact" }, "Id" : "003xx000004U7xKAAS" }, { "attributes" : { "type" : "Contact" }, "Id" : "003xx000004U7xLAAS" }, { "attributes" : { "type" : "Contact" }, "Id" : "003xx000004U7xMAAS" }] } }] }`
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.

Field	Details
`RowsProcessed`	**Type** double **Properties** Nillable **Description** The total number of rows of data returned from the API query when the user executed the query. For big objects, if the total number of returned rows is greater than the API batch size, `RowsProcessed` is `-1`.
`RowsReturned`	**Type** double **Properties** Nillable **Description** The number of rows of data returned in the current API batch. If `RowsProcessed` is less than the API batch size, `RowsReturned` is equal to `RowsProcessed`. If `RowsProcessed` is greater than the API batch size, `RowsReturned` equals either the API batch size or the number of rows in the last batch.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE` - A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW` - The user's security level for the current session meets the lowest requirements. This low level isn't available, or used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality.

Field	Details
	• `STANDARD` - The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The IP from which the API events originated. A Salesforce internal IP (such as from an API event originating from AppExchange) is shown as "Salesforce.com IP".
`UserAgent`	**Type** string **Properties** Nillable **Description** The platform or environment in which the API call originated. This field could include information about the operating system, application, or web protocol. For example, `Mozilla/5.0 (iPhone; CPU iPhone OS 13_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko)`
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

Working With the `AdditionalInfo` Field

`AdditionalInfo` enables you to extend the API event with custom data that can be queried later. For example, you can capture a correlation ID when a user executes a SOQL query from an external system that shares that unique ID. This process enables tracking API calls across systems. To store data with ApiEvent, begin all `AdditionalInfo` field names with `x-sfdc-addinfo-{field name}`. For example, a valid field assignment is `x-sfdc-addinfo-correlation_id = ABC123` where `x-sfdc-addinfo-correlation_id` is the field name and `ABC123` is the field value.

When defining field names, note the following:

- `x-sfdc-addinfo-` is case-*insensitive*; `x-sfdc-addinfo-{field name}` is the same as `X-SFDC-ADDINFO-{field name}` and `x-SfDc-AdDiNfO-{field name}`.
- Fields can contain only alphanumeric and "_" (underscore) characters.
- Field names must be from 2 through 29 characters in length, excluding `x-sfdc-addinfo-`.
- Field names that don't start with `x-sfdc-addinfo-` are ignored.
- Names that contain invalid characters after `x-sfdc-addinfo-` are ignored, and nothing is stored. For example, a valid field name is `x-sfdc-addinfo-correlation_id` but `x-sfdc-addinfo-correlation->id` isn't valid.
- Only the first 30 valid field names are stored in `AdditionalInfo`. If you store two valid field names—for example, `x-sfdc-addinfo-correlation_id` and `x-sfdc-addinfo-correlation_number`— you can store 28 extra field names. Field names aren't necessarily stored in the same order in which they were passed to authentication.
- You can't use existing API field names as `AdditionalInfo` names in the HTTP header. If the `AdditionalInfo` name conflicts with an object's API name, the field value isn't stored. For example, the HTTP header `X-SFDC-ADDINFO-UserId='abc123'` doesn't get stored in `AdditionalInfo`.
- Extra field values can contain only alphanumeric, "_," and "-" characters.
- Field values must be 255 characters in length or fewer. If a field value exceeds 255 characters, only the first 255 characters are stored, and the rest are truncated.
- Field values that contain invalid characters are saved with a field header of Empty String ("").
- Only the first 30 valid field names are stored in the `AdditionalInfo` field. They aren't guaranteed to be stored in the same order that they were passed into the authentication.
- When `AggregationFieldName` or `PlatformEventMetrics` is `SourceIp`, you can't filter on `AggregationFieldValue` if its value is *`Salesforce.com IP.`*

How to Pass Additional Information by Using HTTP with cURL

```
curl
https://yourInstance.salesforce.com/services/data/v34.0/query?q=SELECT+Name+From+Account
 -H "X-PrettyPrint:1" -H "x-sfdc-addinfo-correlationid:
d18c5a3f-4fba-47bd-bbf8-6bb9a1786624"
```

Example of Using Java

```
//adding additional info headers ..
Map<String, String> httpHeaders = new HashMap<String,String>();
httpHeaders.put("x-sfdc-addinfo-fieldname1" /* additional info field*/ ,
"d18c5a3f-4fba-47bd-bbf8-6bb9a1786624" /* value*/);
httpHeaders.put("x-sfdc-addinfo-fieldname2" /* additional info field*/ ,
"d18c5a3f-4fba-47bd-bbf8-6bb9a1786624" /* value*/);

ConnectorConfig config = new ConnectorConfig();
config.setUsername(userId);
config.setPassword(passwd);
config.setAuthEndpoint(authEndPoint);
config.setProxy(proxyHost, proxyPort);

//setting additional info headers
for (Map.Entry<String, String> entry : httpHeaders.entrySet()) {
```

```
      config.setRequestHeader(entry.getKey(), entry.getValue());
}
// Set the username and password if your proxy must be authenticated
      config.setProxyUsername(proxyUsername);
      config.setProxyPassword(proxyPassword);
      try {
        QueryResult queryResult = connection.query("SELECT Id, Name FROM Account");
        // etc.
      } catch (ConnectionException ce) {
       ce.printStackTrace();
      }
```

For the user interface, use proxy servers to intercept call and add required information.

Standard SOQL Usage

ApiEvent allows filtering over two fields: `EventDate` and `EventIdentifier`. The only supported SOQL functions on the ApiEvent object are `WHERE`, `ORDER BY`, and `LIMIT`. In the `WHERE` clause, you can only use comparison operators (<, >, <=, and >=). The != operator isn't supported. In the `ORDER BY` clause, you can only use EventDate `DESC`. Ascending order isn't supported with EventDate, and EventIdentifier sorting isn't supported.

Note: Date functions such as `convertTimeZone()` aren't supported—for example, `SELECT CALENDAR_YEAR(EventDate), Count(Id) FROM ApiEvent GROUP BY CALENDAR_YEAR(EventDate)` returns an error. You can use date literals in your queries and some date/time functions like `TODAY()`, `YESTERDAY()`, and `LAST_n_DAYS:1`. However, these functions use comparison operators behind the scenes. Therefore you can only use them in the final expression in the `WHERE` clause.

Example of an Unfiltered Query

This query is valid because it doesn't contain a `WHERE` clause. No special rules apply.

```
SELECT ApiType, Client, ElapsedTime, QueriedEntities, Username
FROM ApiEvent
```

Example of a Filtered Query

You can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.

```
SELECT ApiType, Client, ElapsedTime, QueriedEntities, Username
FROM ApiEvent
WHERE EventDate>=2014-11-27T14:54:16.000Z
```

SEE ALSO:

Big Objects Implementation Guide

ApiEventStream

Tracks these user-initiated read-only API calls: `query()`, `queryMore()`, and `count()`. Captures API requests through SOAP API and Bulk API for the Enterprise and Partner WSDLs. Tooling API calls and API calls originating from a Salesforce mobile app aren't captured. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/ApiEventStream`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AdditionalInfo`	**Type** string **Properties** Nillable **Description** JSON serialization of additional information that's captured from the `HTTP` headers during an API request. For example, `{"field1": "value1", "field2": "value2"}`.
`ApiType`	**Type** string **Properties** Nillable **Description** The API that was used. Values include:

Field	Details
	• `Bulk` • `REST` • `SOAP Enterprise` • `SOAP Partner` • `N/A`
`ApiVersion`	**Type** double **Properties** Nillable **Description** The version number of the API.
`Application`	**Type** string **Properties** Nillable **Description** The application used to access the org. For example, CRM Analytics or Salesforce Developers Connector.
`Client`	**Type** string **Properties** Nillable **Description** The service that executed the API event. If you're using an unrecognized client, this field returns "Unknown" or a blank value.
`ConnectedAppId`	**Type** string **Properties** Nillable **Description** The 15-character ID of the connected app associated with the API call. For example, 0H4RM00000000Kr0AI. The `ConnectedAppID` field populates when a call triggers an OAuth 2.0 authentication process, which identifies the connected app that's authorized to access Salesforce data on behalf of a user. When a user associated with the call already has an active authentication token, the `ConnectedAppID` is set to a null value.
`ElapsedTime`	**Type** int

Field	Details
	Properties Nillable **Description** The amount of time it took for the request to complete in milliseconds. The measurement of this value begins before the query executes and ends when the query completes. It doesn't include the amount of time it takes to return the result over the network.
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the specified API event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginHistoryId`	**Type** reference

Field	Details
	Properties Nillable **Description** Tracks a user session so you can correlate user activity with a particular series of API events. This field is also available on the LoginEvent, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. For example, 0YaB000002knVQLKA2.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`Operation`	**Type** picklist **Properties** Nillable, Restricted Picklist **Description** The API call that generated the event. Possible values are `Query`, `QueryAll`, or `QueryMore`.
`Platform`	**Type** string **Properties** Nillable **Description** The operating system on the login machine. For example, iPhone, Mac OS, Linux, or Unknown.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY.
`PolicyOutcome`	**Type** picklist

Field	Details
	Properties Nillable, Restricted picklist **Description** The result of the transaction policy. For this event, possible values are: • `Block` - The user was blocked from performing the operation that triggered the policy. • `Error` - The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The entities in the SOQL query. For example, Opportunity, Lead, Account, or Case. Can also include custom objects. For relationship queries, the value of this field contains all entities involved in the query. **Examples** • For `SELECT Contact.FirstName, Contact.Account.Name from Contact`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Account.Name, (SELECT Contact.FirstName, Contact.LastName FROM Account.Contacts) FROM Account`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Id, Name, Account.Name FROM Contact WHERE Account.Industry = 'media'`, the value of `QueriedEntities` is `Account, Contact`.
`Query`	**Type** textarea **Properties** Nillable **Description** The SOQL query. For example, `SELECT id FROM Lead`.

Field	Details
`Records`	**Type** json **Properties** Nillable **Description** A JSON string that represents the queried objects' metadata. This metadata includes the number of results of a query per entity type and the entity IDs. The `Records` field is set to a null value for BULK API queries. Bulk API queries from `ApiEventStream` can exceed bandwidth limitations due to the size of the `Records` field. To reduce the payload size, the `Records` field is set to a null value. **Example** `{ "totalSize" : 1, "done" : true, "records" : [{ "attributes" : { "type" : "Account" }, "Id" : "001xx000003DMvCAAW", "Contacts" : { "totalSize" : 3, "done" : true, "records" : [{ "attributes" : { "type" : "Contact" }, "Id" : "003xx000004U7xKAAS" }, { "attributes" : { "type" : "Contact" }, "Id" : "003xx000004U7xLAAS" }, { "attributes" : { "type" : "Contact" }, "Id" : "003xx000004U7xMAAS" }] } }] }`
`RelatedEventIdentifier`	**Type** string **Properties** Nillable

Field	Details
	Description Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RowsProcessed`	**Type** double **Properties** Nillable **Description** The total number of rows of data returned from the API query when the user executed the query. For big objects, if the total number of returned rows is greater than the API batch size, `RowsProcessed` is `-1`.
`RowsReturned`	**Type** double **Properties** Nillable **Description** The number of rows of data returned in the current API batch. If `RowsProcessed` is less than the API batch size, `RowsReturned` is equal to `RowsProcessed`. If `RowsProcessed` is greater than the API batch size, `RowsReturned` equals either the API batch size or the number of rows in the last batch.
`SessionKey`	**Type** string

Field	Details
	Properties Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • HIGH_ASSURANCE - A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • LOW - The user's security level for the current session meets the lowest requirements. This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality. • STANDARD - The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The IP from which the API events originated. A Salesforce internal IP (such as from an API event originating from AppExchange) is shown as "Salesforce.com IP".
`UserAgent`	**Type** string **Properties** Nillable **Description** The platform or environment in which the API call originated. This field could include information about the operating system, application, or web protocol. For example, `Mozilla/5.0 (iPhone; CPU iPhone OS 13_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko)`

Field	Details
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

BulkApiResultEvent

Tracks when a user downloads the results of a Bulk API or Bulk API 2.0 request.

Supported Calls

`describeSObjects()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.

Field	Details
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular login instance. This field is also available on the LoginHistory, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.

Field	Details
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, `0NIB000000000KOOAY`. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values include: • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`Query`	**Type** string **Properties** Nillable **Description** The SOQL query. For example, `SELECT Id FROM Account`
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, `bd76f3e7-9ee5-4400-9e7f-54de57ecd79c`.

Field	Details
	This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level isn't available or used in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users who are assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.

Field	Details
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

BulkApiResultEventStore

Tracks when a user downloads the results of a Bulk API request. BulkApiResultEventStore is a big object that stores the event data of BulkApiResultEvent. This object is available in API version 50.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`EvaluationTime`	**Type** double

Field	Details
	Properties Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular login instance. This field is also available on the LoginHistory, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`PolicyId`	**Type** reference

Field	Details
	Properties Nillable **Description** The ID of the transaction policy associated with this event. For example, `0NIB000000000KOOAY`. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values include: • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`Query`	**Type** string **Properties** Nillable **Description** The SOQL query. For example, `SELECT Id FROM Account`
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, `bd76f3e7-9ee5-4400-9e7f-54de57ecd79c`. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more

Field	Details
	events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level is not available or used in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`UserId`	**Type** reference

Field	Details
	Properties Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

ConcurLongRunApexErrEvent

Notifies subscribers of errors that occur when a Salesforce org exceeds the concurrent long-running Apex limit. If a high volume of these events occur concurrently in an org, we may rate limit the events based on resource availability. Event log files, which are the predecessor of Real-time Event Monitoring, provide a list of Apex-related events. For more information, see Apex-related EventLogFile events. This object is available in API version 49.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/ConcurLongRunApexErrEvent`

Event Delivery Allocation Enforced

No

Fields

Field	Details
`CurrentValue`	**Type** int **Properties** Nillable **Description** The current count of concurrent long-running Apex requests in the org.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the Apex request failed to start and generated the error. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object, if any. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LimitValue`	**Type** int **Properties** Nillable **Description** The limit value that was exceeded.
`LoginKey`	**Type** string

Field	Details
	Properties Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring.
`Quiddity`	**Type** string **Properties** Nillable **Description** The type of outer execution associated with this event. **Example** • `A`–QueryLocator Batch Apex (Batch Apex jobs run faster when the start method returns a QueryLocator object that doesn't include related records via a subquery. See Best Practices in Using Batch Apex.) • `B`– Bulk API and Bulk API 2.0 • `BA`–Batch Apex (for debugger) • `C`–Scheduled Apex • `E`–Inbound Email Service • `F`–Future • `H`–Apex REST • `I`–Invocable Action • `K`–Quick Action • `L`–Lightning • `M`–Remote Action • `Q`–Queuable • `R`–Synchronous uncategorized (default value for transactions not specified elsewhere) • `S`–Serial Batch Apex • `TA`–Tests Async • `TD`–Tests Deployment • `TS`–Tests Synchronous • `V`–Visualforce • `W`–SOAP Webservices • `X`–Execute Anonymous
`ReplayId`	**Type** string

Field	Details
	Properties Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestId`	**Type** string **Properties** Nillable **Description** The unique ID of the Apex request that fired the event.
`RequestUri`	**Type** string **Properties** Nillable **Description** The URI of the Apex request that failed to start and generated the error. **Example** /apex/ApexClassName
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. You can use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • HIGH_ASSURANCE - A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level.

Field	Details
	• LOW - The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • STANDARD - The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The IP address from which the Apex request originated.
`UserId`	**Type** reference **Properties** Nillable **Description** The unique ID of the user associated with the Apex request.
`Username`	**Type** string **Properties** Nillable **Description** The username of the user associated with the Apex request.

CredentialStuffingEvent

Tracks when a user successfully logs into Salesforce during an identified credential stuffing attack. Credential stuffing refers to large-scale automated login requests using stolen user credentials. This object is available in API version 49.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	✓
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/CredentialStuffingEvent`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AcceptLanguage`	**Type** string **Properties** Nillable **Description** List of HTTP Headers that specify the natural language, such as English, that the client understands. **Example** `zh, en-US;q=0.8, en;q=0.6`
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.

Field	Details
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the hijacking event was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`LoginType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of login used to access the session. See the LoginType field of LoginHistory in the Object Reference guide for the list of possible values.

Field	Details
`LoginUrl`	**Type** string **Properties** Nillable **Description** The URL of the login page. For example, ***`MyDomainName`*`.my.salesforce.com`**.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Error` - The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive

Field	Details
	events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`Score`	**Type** double **Properties** Nillable **Description** Indicates that a user successfully logged into Salesforce during an identified credential stuffing attack. The value of this field is always `1`.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the unauthorized user that successfully logged in after the credential stuffing attack. For example, `126.7.4.2`.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the threat that caused this event to be created. **Example** `Successful login from Credential Stuffing attack.`
`UserAgent`	**Type** textarea **Properties** Nillable

Field	Details
	Description The User-Agent header of the HTTP request of the unauthorized login. For example, `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36`.
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

CredentialStuffingEventStore

Tracks when a user successfully logs into Salesforce during an identified credential stuffing attack. Credential stuffing refers to large-scale automated login requests using stolen user credentials. CredentialStuffingEventStore is an object that stores the event data of CredentialStuffingEvent. This object is available in API version 49.0 and later.

Supported Calls

```
describeLayout(), describeSObjects(), getDeleted(), getUpdated(), query()
```

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`AcceptLanguage`	**Type** string **Properties** Filter, Group, Nillable, Sort

Field	Details
	Description List of HTTP Headers that specify the natural language, such as English, that the client understands. **Example** `zh, en-US;q=0.8, en;q=0.6`
`CredentialStuffingEventNumber`	**Type** string **Properties** Autonumber, Defaulted on create, Filter, idLookup, Sort **Description** The unique number automatically assigned to the event when it's created. You can't change the format or value for this field.
`EvaluationTime`	**Type** double **Properties** Filter, Nillable, Sort **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** Required. The time when the hijacking event was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Group, Sort **Description** Required. The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`LastReferencedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort

Field	Details
	Description The timestamp for when the current user last viewed a record related to this record.
`LastViewedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The timestamp for when the current user last viewed this record. If this value is null, it's possible that this record was referenced (`LastReferencedDate`) and not viewed.
`LoginKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`LoginType`	**Type** picklist **Properties** Filter, Group, Nillable, Restricted picklist, Sort **Description** The type of login used to access the session. See the LoginType field of LoginHistory in the Object Reference guide for the list of possible values.
`LoginUrl`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The URL of the login page. For example, ***`MyDomainName`***`.my.salesforce.com`.
`PolicyId`	**Type** reference **Properties** Filter, Group, Nillable, Sort

Field	Details
	Description The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Filter, Group, Nillable, Restricted picklist, Sort **Description** The result of the transaction policy. Possible values are: • `Error` - The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`Score`	**Type** double **Properties** Filter, Nillable, Sort **Description** Indicates that a user successfully logged into Salesforce during an identified credential stuffing attack. The value of this field is always `1`.
`SessionKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SourceIp`	**Type** string

Field	Details
	Properties Filter, Group, Nillable, Sort **Description** The source IP address of the unauthorized user that successfully logged in after the credential stuffing attack. For example, `126.7.4.2`.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the threat that caused this event to be created. **Example** `Successful login from Credential Stuffing attack.`
`UserAgent`	**Type** textarea **Properties** Nillable **Description** The User-Agent header of the HTTP request of the unauthorized login. For example, `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36`.
`UserId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The origin username in the format of `user@company.com` at the time the event was created.

Associated Object

This object has the following associated object. It's available in the same API version as this object.

CredentialStuffingEventStoreFeed
Feed tracking is available for the object.

CreditInvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/invoices/{invoiceId}/actions/credit` request is complete. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/CreditInvoiceProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable

Field	Details
	Description Reserved for future use.
`CrMemoProcessErrDtlEvents`	**Type** CreditMemoProcessedErrDtlEvent[] **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available only in API versions 55.0–58.0. See the `ErrorDetails` field for error messages and error codes.
`CreditMemoId`	**Type** reference **Properties** Nillable **Description** The credit memo created as the result of a successful request. This field is a relationship field. **Relationship Name** CreditMemo **Relationship Type** Lookup **Refers To** CreditMemo
`ErrorDetails`	**Type** string **Properties** Nillable **Description** If the request fails, this field shows error messages, error codes, and the ID of the record on which the errors occurred. This field is available in API 58.0 and later.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

Field	Details
`InvoiceId`	**Type** reference **Properties** Nillable **Description** The invoice credited as the result of a successful request. This field is a relationship field. **Relationship Name** Invoice **Relationship Type** Lookup **Refers To** Invoice
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the request was successful. The default value is 'false'.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the response. Use this ID to identify the event for a specific request.

IN THIS SECTION:

CrMemoProcessErrDtlEvent
Contains information about errors that occurred while creating or applying a credit memo as part of a request. This object is included in a `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` message. You can't subscribe to `CrMemoProcessErrDtlEvent` directly. This object is available in API versions 55.0–58.0. In API version 58.0, this field returns a null result. See the `ErrorDetails` field on the `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` object for error information.

CreditMemoProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/credit-memos` request is complete. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/CreditMemoProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** Reserved for future use.
`CrMemoProcessErrDtlEvents`	**Type** CreditMemoProcessedErrDtlEvent[] on page 335 **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available only in API versions 55.0–58.0. See the `ErrorDetails` field for error messages and error codes.
`CreditMemoId`	**Type** reference **Properties** Nillable **Description** The credit memo created as the result of a successful request. This field is a relationship field. **Relationship Name** CreditMemo **Relationship Type** Lookup **Refers To** CreditMemo
`ErrorDetails`	**Type** string **Properties** Nillable **Description** If the request fails, this field shows error messages, error codes, and the ID of the record on which the errors occurred. This field is available in API 58.0 and later.
`EventUuid`	**Type** string **Properties** Nillable

Field	Details
	Description A universally unique identifier (UUID) that identifies a platform event message.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the Create Standalone Credit Memo action was successful. The default value is 'false'.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/commerce/invoicing/credit-memos` response. Use this ID to identify the event for a specific request.

Example: A user successfully runs a `/commerce/invoicing/credit-memos`, creates one credit memo, and receives this platform event when the request completes.

```
{
  "IsSuccess": true,
  "CrMemoProcessErrDtlEvents": null,
  "CreatedById": "005RO000000g4LYYAY",
  "CorrelationIdentifier": "50gRO0000000jxc",
  "CreatedDate": "2023-03-17T15:09:18Z",
  "ErrorDetails": "[]",
  "InvoiceId": "3ttRO0000006839YAA",
  "CreditMemoId": "50gRO0000000jxcYAA",
  "RequestIdentifier": "d488e070-0fd8-4cde-a9fd-d7ca38d040f5"
}
```

Example: A user runs a `/commerce/invoicing/invoices/{invoiceId}/actions/credit` request, which fails because the credit memo's amount is greater than the invoice's balance.


```
{
  "IsSuccess": false,
  "CrMemoProcessErrDtlEvents": null,
  "CreatedById": "005RO0000000g4LYYAY",
  "CorrelationIdentifier": "50gRO0000000jzi",
  "CreatedDate": "2023-03-17T22:55:11Z",
    "ErrorDetails": "[{
 "ErrorSourceId": "50gRO0000000jzi",
 "ErrorCode": "RECORD_UPDATE_FAILED",
 "ErrorMessage": "An error occurred while updating the credit memo status to POSTED:
Child events testing - fail updating credit memo status to posted Failed object Ids :
 50gRO0000000jzi"
}]",
  "CreditMemoId": "50gRO0000000jziYAA",
  "RequestIdentifier": "9123a706-4a64-4beb-8942-4eb5abd1e59f"
 },
```


CrMemoProcessErrDtlEvent

Contains information about errors that occurred while creating or applying a credit memo as part of a request. This object is included in a `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` message. You can't subscribe to `CrMemoProcessErrDtlEvent` directly. This object is available in API versions 55.0–58.0. In API version 58.0, this field returns a null result. See the `ErrorDetails` field on the `CreditInvoiceProcessedEvent`, `CreditMemoProcessedEvent`, `NegInvcLineProcessedEvent`, or `VoidInvoiceProcessedEvent` object for error information.

Supported Calls

`describeSObjects()`

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`ErrorCode`	**Type** string **Properties** Nillable **Description** Reference code for the type of error that occurred.

Field	Details
`ErrorMessage`	**Type** string **Properties** Nillable **Description** Information about the error that occurred during processing.
`ErrorSourceId`	**Type** reference **Properties** Nillable **Description** The ID of the record on which the error occurred during the credit memo creation process and the application process. This field is a polymorphic relationship field. **Relationship Name** ErrorSource **Relationship Type** Lookup **Refers To** CreditMemo, CreditMemoLine, Invoice, InvoiceLine
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.

GuestUserAnomalyEvent

Tracks data access anomalies that are caused by guest user permission misconfiguration. This object is available in API version 60.0 and later.

Supported Calls

`describeSObjects()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	✓
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Event Delivery Allocation Enforced

No

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** A date value that represents the aggregate timeframe when the guest user's actions occurred.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object.
`EventUuid`	**Type** string **Properties** Nillable

Field	Details
	Description A universally unique identifier (UUID) that identifies a platform event message.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, `0NIB000000000KOOAY`. **Relationship Name** Policy **Relationship Type** Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values include: • `Error`—The policy caused an undefined error when it was executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient.

Field	Details
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestedEntities`	**Type** textarea **Properties** Nillable **Description** Objects queried by the guest user. For example: `[\" Topic \"].`
`Score`	**Type** double **Properties** Nillable **Description** Specifies how significantly the guest user behavior deviates from the other guest users. It's formatted as a number between 0 and 1.
`SecurityEventData`	**Type** textarea **Properties** Nillable **Description** The content data of the security event. This field is reserved for future use.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session.

Field	Details
`SoqlCommands`	**Type** textarea **Properties** Nillable **Description** SOQL commands run by the guest user.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, 126.7.4.2.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the threat that caused this event to be created. The summary lists the browser fingerprint features that most contributed to the threat detection along with their contribution to the total score.
`TotalControllerEvents`	**Type** int **Properties** Nillable **Description** The number of times controllers were triggered.
`UserAgent`	**Type** string **Properties** Nillable **Description** User Agent for this event.
`UserId`	**Type** reference **Properties** Nillable

Field	Details
	Description The origin user's unique ID. For example, `005000000000123`. This field is a polymorphic relationship field. **Relationship Name** User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.
`UserType`	**Type** string **Properties** Nillable **Description** Type of user of this event. For example, a guest user.

GuestUserAnomalyEventStore

Tracks data access anomalies that are caused by guest user permission misconfiguration. GuestUserAnomalyEventStore is an object that stores the event data of GuestUserAnomalyEvent. This object is available in API version 60.0 and later.

Supported Calls

`describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Filter, Nillable, Sort **Description** The amount of time it took to evaluate the policy in milliseconds.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** A date value that represents the aggregate timeframe when the guest user's actions occurred.
`EventIdentifier`	**Type** string **Properties** Filter, Group, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object.
`GuestUserAnomalyEventNumber`	**Type** string **Properties** Autonumber, Defaulted on create, Filter, idLookup, Sort **Description** The unique number automatically assigned to the event when it's created. You can't change the format or value for this field.
`LastReferencedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The date the event was last referenced.
`LastViewedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort

Field	Details
	Description The date the event was last viewed.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `1UqjLPQTWRdvRG4`.
`PolicyId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** The ID of the transaction policy associated with this event. For example, `0NIB000000000KOOAY`. **Relationship Name** Policy **Relationship Type** Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values include: • `Error`—The policy caused an undefined error when it was executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient.

Field	Details
`RequestedEntities`	**Type** textarea **Properties** Nillable **Description** Objects queried by the guest user. For example: `[\" Topic \"].`
`Score`	**Type** double **Properties** Filter, Nillable, Sort **Description** Specifies how significantly the guest user behavior deviates from the other guest users. It is formatted as a number between 0 and 1.
`SecurityEventData`	**Type** textarea **Properties** Nillable **Description** The content data of the security event. This field is reserved for future use.
`SessionKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The user's unique session ID. Use this value to identify all user events within a session.
`SoqlCommands`	**Type** textarea **Properties** Nillable **Description** SOQL commands run by the guest user.
`SourceIp`	**Type** string **Properties** Filter, Group, Nillable, Sort

Field	Details
	Description The source IP address of the client that logged in. For example, 126.7.4.2.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the threat that caused this event to be created. The summary lists the browser fingerprint features that most contributed to the threat detection along with their contribution to the total score.
`TotalControllerEvents`	**Type** int **Properties** Filter, Group, Nillable, Sort **Description** The number of times controllers were triggered.
`UserAgent`	**Type** string **Properties** Filter, Nillable, Sort **Description** User Agent for this event.
`UserId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** The origin user's unique ID. For example, `005000000000123`. This field is a polymorphic relationship field. **Relationship Name** User **Relationship Type** Lookup **Refers To** User

Field	Details
`Username`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The origin username in the format of `user@company.com` at the time the event was created.
`UserType`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** Type of user of this event. For example, a guest user.

PaymentCreationEvent

Notifies subscribers when the process started by the `/actions/standard/paymentSale` request is complete. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/PaymentCreationEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

To access Commerce Payments entities, your org must have a Salesforce Order Management license with the Payment Platform org permission activated. Commerce Payments entities are available only in Lightning Experience.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`ErrorCode`	**Type** string **Properties** Nillable **Description** Error code sent from the payment gateway after a request encountered an error.
`ErrorMessage`	**Type** textarea **Properties** Nillable **Description** Message sent from the payment gateway after a request encountered an error.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the request was successful. The default value is 'false'.

Field	Details
`PaymentGatewayLogId`	**Type** reference **Properties** Nillable **Description** The payment gateway log containing information about the communication with the payment gateway. This is a relationship field. **Relationship Name** PaymentGatewayLog **Relationship Type** Lookup **Refers To** PaymentGatewayLog
`PaymentId`	**Type** reference **Properties** Nillable **Description** The payment created as the result of a successful request. This is a relationship field. **Relationship Name** Payment **Relationship Type** Lookup **Refers To** Payment
`PaymentStatus`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The status of the payment created after a successful request. This field reflects the status upon payment creation, and isn't updated after further changes to the payment's status. Possible values are: • `Canceled` • `Draft` • `Failed`

Field	Details
	• `Pending` • `Processed`
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/actions/standard/paymentSale` response. Use this ID to identify the event for a specific request.
`Type`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Indicates whether the payment was made for a payment capture request or payment sale request. Possible values are: • `Capture` • `Sale`

VoidInvoiceProcessedEvent

Notifies subscribers when the process started by the `/commerce/invoicing/invoices/{invoiceId}/actions/void` request is complete. The request attempts to void an invoice by crediting an invoice and changing its status to Voided, which prevents further changes. This object is available in API version 55.0 and later. This object is available in API version 55.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/VoidInvoiceProcessedEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

This object is available when Subscription Management is enabled.

Fields

Field	Details
`CorrelationIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`CrMemoProcessErrDtlEvents`	**Type** CrMemoProcessErrDtlEvent[] **Properties** Nillable **Description** Contains a list of error messages and error codes if the request failed. This field is available only in API versions 55.0–58.0. See the `ErrorDetails` field for error messages and error codes.
`CreditMemoId`	**Type** reference

Field	Details
	Properties Nillable **Description** The credit memo created to void the invoice as the result of a successful request. This field is a relationship field. **Relationship Name** CreditMemo **Relationship Type** Lookup **Refers To** CreditMemo
`ErrorDetails`	**Type** string **Properties** Nillable **Description** If the request fails, this field shows error messages, error codes, and the ID of the record on which the errors occurred. This field is available in API 58.0 and later.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`InvoiceId`	**Type** reference **Properties** Nillable **Description** The invoice that was voided as the result of a successful request. This field is a relationship field. **Relationship Name** Invoice **Relationship Type** Lookup **Refers To** Invoice

Field	Details
`IsSuccess`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the request was successful. The default value is 'false'.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RequestIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID returned in the `/commerce/billing/invoices/{invoiceId}/actions/void` response. Use this ID to identify the event for a specific request.

FileEvent

Tracks when a user downloads a document. This information includes events performed on files. This object is available in API version 57.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	

Subscriber	Supported?
Processes	
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/FileEvent`

Event Delivery Allocation Enforced

No

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`CanDownloadPdf`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the downloaded PDF was converted from another file type. The default value is false.
`ContentSize`	**Type** int **Properties** Nillable **Description** The size of the document, in bytes.
`DocumentId`	**Type** string **Properties** Nillable **Description** The 18-character ID of the document that's being downloaded. The ID is a reference to the `ContentDocument` object.

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the transaction security policy in milliseconds.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time when the file event was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`FileAction`	**Type** string **Properties** Nillable **Description** The action taken on the file. Valid values are: • `API_DOWNLOAD` • `PREVIEW` • `UI_DOWNLOAD` • `UPLOAD`

Field	Details
	This field is available in API version 58.0 and later.
`FileName`	**Type** string **Properties** Nillable **Description** The name of the file, including the file extension.
`FileSource`	**Type** string **Properties** Nillable **Description** Origin of the document. Valid values are: • `S`—Document is located within Salesforce. Label is **Salesforce**. • `E`—Document is located outside of Salesforce. Label is **External**. • `L`—Document is located on a social network and accessed via Social Customer Service. Label is **Social Customer Service**.
`FileType`	**Type** string **Properties** Nillable **Description** The content type of the file. For example, `PDF`.
`IsLatestVersion`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the file is the most current version (true) or not (false). The default value is false.
`LoginKey`	**Type** string **Properties** Nillable

Field	Details
	Description The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This is a relationship field. **Relationship Name** Policy **Relationship Type** Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient.
`ProcessDuration`	**Type** double **Properties** Nillable

Field	Details
	Description The amount of time to download the file, in milliseconds.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, `bd76f3e7-9ee5-4400-9e7f-54de57ecd79c`. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are:

Field	Details
	• `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. This low level isn't available or used in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users who are assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005B0000001vURv`. This is a polymorphic relationship field. **Relationship Name** User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.
`VersionId`	**Type** string

Field	Details
	Properties Nillable **Description** The specific version of a document in Salesforce CRM Content or Salesforce Files. The ID is a reference to the ContentVersion object.
`VersionNumber`	**Type** string **Properties** Nillable **Description** The version number of the file.

FileEventStore

Tracks when a user downloads, previews, or uploads a file. FileEventStore is a big object that stores the event data of FileEvent. This object is available in API version 57.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`CanDownloadPdf`	**Type** boolean **Properties** Defaulted on create **Description** Indicates whether the downloaded PDF was converted from another file type. The default value is `false`.
`ContentSize`	**Type** int **Properties** Nillable

Field	Details
	Description The size of the document, in bytes
`DocumentId`	**Type** reference **Properties** Nillable **Description** The 18-character ID of the document that's being downloaded. The ID is a reference to the `ContentDocument` object. This is a relationship field. **Relationship Name** Document **Relationship Type** Lookup **Refers To** ContentDocument
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the transaction security policy in milliseconds.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time when the file event was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.

Field	Details
`FileAction`	**Type** string **Properties** Nillable **Description** The action taken on the file. Valid values are: • `API_DOWNLOAD` • `PREVIEW` • `UI_DOWNLOAD` • `UPLOAD` This field is available in API version 58.0 and later.
`FileName`	**Type** string **Properties** Nillable **Description** The name of the file, including the file extension.
`FileSource`	**Type** string **Properties** Nillable **Description** Origin of the document. Valid values are: • `S`—Document is located within Salesforce. Label is **Salesforce**. • `E`—Document is located outside of Salesforce. Label is **External**. • `L`—Document is located on a social network and accessed via Social Customer Service. Label is **Social Customer Service**.
`FileType`	**Type** string **Properties** Nillable **Description** The content type of the file.
`IsLatestVersion`	**Type** boolean

Field	Details
	Properties Defaulted on create **Description** Indicates whether the file is the most current version (true) or not (false). The default value is false.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This is a relationship field. **Relationship Name** Policy **Relationship Type** Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger.

Field	Details
	• `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient.
`ProcessDuration`	**Type** double **Properties** Nillable **Description** The amount of time to download the file, in milliseconds.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, `bd76f3e7-9ee5-4400-9e7f-54de57ecd79c`. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist

Field	Details
	Description Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. This low level isn't available or used in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users who are assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005B0000001vURv`. This is a polymorphic relationship field. **Relationship Name** User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string **Properties** Nillable

Field	Details
	Description The origin username in the format of `user@company.com` at the time the event was created.
`VersionId`	**Type** reference **Properties** Nillable **Description** The specific version of a document in Salesforce CRM Content or Salesforce Files. The ID is a reference to the ContentVersion object. This is a relationship field. **Relationship Name** Version **Relationship Type** Lookup **Refers To** ContentVersion
`VersionNumber`	**Type** string **Properties** Nillable **Description** The version number of the file.

IdentityProviderEventStore

Tracks problems and successes with inbound SAML or OpenID Connect authentication requests from another app provider. It also records outbound SAML responses when Salesforce is acting as an identity provider. IdentityProviderEventStore is a big object. This object is available in API version 51.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`AppId`	**Type** reference **Properties** Nillable **Description** The ID of the app provider seeking authentication.
`AuthSessionId`	**Type** reference **Properties** Nillable **Description** The ID of the authentication session.
`ErrorCode`	**Type** picklist **Properties** Restricted picklist **Description** The error code for the authentication issue. Possible values are: • `AppAccessDenied`—Error: App access denied • `AppBlocked`—Error: App blocked • `ClientUnapproved`—Error: Invalid grant • `CodeExpired`—Error: Expired authorization code • `InternalError`—Error: Internal Error • `InvalidAuthnRequest`—Error: Unable to parse AuthnRequest from service provider • `InvalidClientCredentials`—Error: Invalid client credentials • `InvalidCode`—Error: Invalid authorization code • `InvalidDeviceId`—Error: Invalid device ID • `InvalidIdpEndpoint`—Error: Invalid Identity Provider Endpoint URL • `InvalidIssuer`—Error: Invalid Issuer • `InvalidScope`—Error: One or more invalid scopes • `InvalidSessionLevel`—Error: Invalid session level • `InvalidSettings`—Error: IdP certificate is invalid or doesn't exist • `InvalidSignature`—Error: Invalid Signature • `InvalidSp`—Error: Misconfigured or invalid service provider

Field	Details
	• `InvalidSpokeSp`—Error: Invalid spoke SP settings • `InvalidUserCredentials`—Error: Invalid user credentials • `NoAccess`—Error: User doesn't have access to this service provider • `NoCustomAttrValue`—Error: User doesn't have a value for the subject custom attribute • `NoCustomField`—Error: Custom field not found • `NoSpokeId`—Error: No Spoke ID found • `NoSubdomain`—Error: My Domain isn't configured • `NoUserFedId`—Error: User doesn't have a Federation Identifier selected • `OauthError`—OAuth Error • `Success` • `UnableToResolve`—Error: Unable to resolve request into a Service Provider • `UnknownError`—Unknown Error
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date and time of the event.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique identifier for each record in IdentityProviderEventStore.
`HasLogoutUrl`	**Type** boolean **Properties** Defaulted on create **Description** Whether a logout URL has been assigned to the app. Users are redirected to this URL when they log out. The default value is `false`.
`IdentityUsed`	**Type** string **Properties** Nillable

Field	Details
	Description The identity (username) of the user being authenticated.
`InitiatedBy`	**Type** picklist **Properties** Restricted picklist **Description** The code describing how the authentication request was initiated. Possible values are: • `IdP`—IdP-Initiated SAML • `OauthAuthorize`—OAuth Authorization • `OauthTokenExchange`—OAuth Token Exchange • `SP`—SP-Initiated SAML • `Unused`
`SamlEntityUrl`	**Type** string **Properties** **Description** The authentication URL of the SAML provider.
`SsoType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of SSO. Possible values are: • `Oidc` • `Saml`
`UserId`	**Type** reference **Properties** Nillable

Field	Details
	Description The ID of the user seeking authentication.

SEE ALSO:

Big Objects Implementation Guide

IdentityVerificationEvent

Tracks user identity verification events in your org. IdentityVerificationEvent is a big object that stores the event data when users are prompted to verify their identity. Available in API version 47.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`Activity`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The action the user attempted that requires identity verification. Possible values include: • `AccessReports`—The user attempted to access reports or dashboards. • `Apex`—The user attempted to access a Salesforce resource with a verification Apex method. • `ChangeEmail`—The user attempted to change an email address. • `ConnectSms`—The user attempted to connect a phone number. • `ConnectToopher`—The user attempted to connect Salesforce Authenticator. • `ConnectTotp`—The user attempted to connect a one-time password generator. • `ConnectU2F`—The user attempted to register a U2F security key. • `ConnectWebAuthRoaming`—The user attempted to register a WebAuthn security key. • `ConnectedApp`—The user attempted to access a connected app.

Field	Details
	• `EnableLL`—The user attempted to enroll in Lightning Login. • `ExportPrintReports`—The user attempted to export or print reports or dashboards. • `ExternalClientApp`— The user attempted to access an external client app. • `ExtraVerification`—`ExtraVerification`—Reserved for future use. • `ListView`—The user attempted to access a list view. • `Login`—The user attempted to log in. • `Registration`—Reserved for future use. • `TempCode`—The user attempted to generate a temporary verification code.
`City`	**Type** string **Properties** Nillable **Description** The city where the user's IP address is physically located. This value isn't localized. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`Country`	**Type** string **Properties** Nillable **Description** The country where the user's IP address is physically located. This value isn't localized. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`CountryIso`	**Type** string **Properties** Nillable **Description** The ISO 3166 code for the country where the user's IP address is physically located. For more information, see Country Codes - ISO 3166.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date and time of the identity verification attempt, for example, `7/19/2025, 3:19:13 PM PDT`. The time zone is based on GMT.

Field	Details
`EventGroup`	**Type** string **Properties** Nillable **Description** ID of the verification attempt. Verification can involve several attempts and use different verification methods. For example, in a user's session, a user enters an invalid verification code (first attempt). The user then enters the correct code and successfully verifies identity (second attempt). Both attempts are part of a single verification and, therefore, have the same ID.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`Latitude`	**Type** double **Properties** Nillable **Description** The latitude where the user's IP address is physically located. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so that you can correlate user activity with a particular login instance.

Field	Details
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring.
`Longitude`	**Type** double **Properties** Nillable **Description** The longitude where the user's IP address is physically located. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`Policy`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The identity verification security policy or setting. • `CustomApex`—Identity verification made by a verification Apex method. • `DeviceActivation`—Identity verification required for users logging in from an unrecognized device or new IP address. This verification is part of Salesforce's risk-based authentication. • `EnableLightningLogin`— Identity verification required for users enrolling in Lightning Login. This verification is triggered when the user attempts to enroll. Users are eligible to enroll if they have the Lightning Login User user permission and the org has enabled Allow Lightning Login in Session Settings. • `ExtraVerification`—Reserved for future use. • `HighAssurance`—High assurance session required for resource access. This verification is triggered when the user tries to access a resource, such as a connected app, report, or dashboard, that requires a high-assurance session level. • `LightningLogin`—Identity verification required for internal users logging in via Lightning Login. This verification is triggered when the enrolled user attempts to log in. Users are eligible to log in if they have the Lightning Login User user permission and have successfully enrolled in Lightning Login. Also, from Session Settings in Setup, Allow Lightning Login must be enabled. • `PageAccess`—Identity verification required for users attempting to perform an action, such as changing an email address or adding a verification method for multi-factor authentication (MFA).

Field	Details
	• `Passwordless Login`—Identity verification required for customers attempting to log in to an Experience Cloud site that is set up for passwordless login. The admin controls which registered verification methods can be used, for example, email, SMS, Salesforce Authenticator, or TOTP. • `ProfilePolicy`—Session security level required at login. This verification is triggered by the Session security level required at login setting on the user's profile. • `TwoFactorAuthentication`—Multi-factor authentication (formerly called two-factor authentication) required at login. This verification is triggered by the Multi-Factor Authentication for User Interface Logins user permission assigned to a custom profile. Or the user permission is included in a permission set that is assigned to a user.
`PostalCode`	**Type** string **Properties** Nillable **Description** The postal code where the user's IP address is physically located. This value isn't localized. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`Remarks`	**Type** string **Properties** Nillable **Description** The text users see on the page or in Salesforce Authenticator when prompted to verify their identity. For example, if identity verification is required for users to log in, they see "You're trying to Log In to Salesforce." In this case, the Remarks value is "Log In to Salesforce." But if the Activity value is Apex, the Remarks value is a custom description specified in the Apex method. If users are verifying their identity using Salesforce Authenticator, the custom description also appears in the app. If the custom description isn't specified, the Remarks value is the name of the Apex method. The label is Activity Message.
`ResourceId`	**Type** reference **Properties** Nillable **Description** If the `Activity` value is ConnectedApp, the `ResourceId` value is the ID of the connected app. The label is Connected App ID.
`SessionKey`	**Type** string

Field	Details
	Properties Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—Used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—Indicates that the user's security level for the current session meets the lowest requirements. Note: This low level is not available or used in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—Indicates that the user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The IP address of the machine from which the user attempted the action that requires identity verification. For example, the IP address of the machine from where the user tried to log in or access reports. If it's a non-login action that required verification, the IP address can be different from the address from where the user logged in. This address can be an IPv4 or IPv6 address.
`Status`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The status of the identity verification attempt.

Field	Details
	• `AutomatedSuccess`—Salesforce approved the request for access because the request came from a trusted location. After a user enables location services in Salesforce, the user can designate trusted locations. When the user trusts a location for a particular activity, such as logging in from a recognized device, that activity is approved from the trusted location for as long as the location is trusted. • `Denied`—The user denied the approval request in the authenticator app. • `FailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `FailedInvalidCode`—The user entered an invalid verification code. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user attempted to enter a password too many times. • `FailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `InProgress`—Salesforce challenged the user to verify identity and is waiting for either the user to respond or for Salesforce to send an automated response. • `Initiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `ReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `Succeeded`—The user's identity was verified.
`Subdivision`	**Type** string **Properties** Nillable **Description** The name of the subdivision where the user's IP address is physically located. In the United States, this value is usually the state name (for example, Pennsylvania). This value isn't localized. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`UserId`	**Type** reference **Properties** Nillable **Description** ID of the user verifying identity.
`Username`	**Type** string **Properties** Nillable

Field	Details
	Description The username of the user challenged for identity verification in `user@company.com` format.
`VerificationMethod`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The method by which the user attempted to verify identity in the verification event. • `BuiltInAuthenticator`—Reserved for future use. • `Email`—Salesforce sent an email with a verification code to the address associated with the user's account. • `EnableLL`—Salesforce Authenticator sent a notification to the user's mobile device to enroll in Lightning Login. • `LL`—Salesforce Authenticator sent a notification to the user's mobile device to approve login via Lightning Login. • `Password`—Salesforce prompted for a password. • `SalesforceAuthenticator`—Salesforce Authenticator sent a notification to the user's mobile device to verify account activity. • `Sms`—Salesforce sent a text message with a verification code to the user's mobile device. SMS messaging requires a Salesforce add-on license for Identity Verification Credits. • `TempCode`—A Salesforce admin or a user with the Manage Multi-Factor Authentication in User Interface permission generated a temporary verification code for the user. • `Totp`—An authenticator app generated a time-based, one-time password (TOTP) on the user's mobile device. • `U2F`—A U2F security key-generated required credentials for the user. • `WebAuthnRoamingAuthenticator`—A WebAuthn security key generated the required credentials for the user.

Standard SOQL Usage

Example

```
SELECT Username, EventGroup, Activity, Policy, Status, VerificationMethod, City, Country,
Latitude, Longitude FROM IdentityVerificationEvent
```

Async SOQL Usage

With Async SOQL, you can filter on any field in IdentityVerificationEvent and use any comparison operator in your query.

Example: Find all successful identity verification events in the org

```
SELECT Username, EventGroup, Activity, Policy, Status, VerificationMethod, Latitude,
Longitude FROM IdentityVerificationEvent WHERE Status='Succeeded'
```

LightningUriEvent

Detects when a user creates, accesses, updates, or deletes a record in Lightning Experience only. LightningUriEvent is a big object that stores the event data of LightningUriEventStream. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Note: LightningUriEvent doesn't track Setup events.

The browser sends Lightning URI events, including inline record changes, to the server in batches. Batches are generally sent when the user navigates around the page and when the page closes or refreshes. The event's timestamp reflects the time that the server receives the batch, not the time that the user changes records on the client side. This batch upload behavior is subject to change, so we don't recommend relying on certain actions to upload batches to the server.

Fields

Field	Details
`AppName`	**Type** string **Properties** Nillable **Description** The name of the application that the user accessed.
`ConnectionType`	**Type** string **Properties** Nillable **Description** The type of connection. **Possible Values** • `CDMA1x` • `CDMA` • `EDGE` • `EVDO0` • `EVDOA` • `EVDOB` • `GPRS`

Field	Details
	• HRPD • HSDPA • HSUPA • LTE • WIFI
DeviceId	**Type** string **Properties** Nillable **Description** The unique identifier used to identify a device when tracking events. DEVICE_ID is a generated value that's created when the mobile app is initially run after installation.
DeviceModel	**Type** string **Properties** Nillable **Description** The name of the device model.
DevicePlatform	**Type** string **Properties** Nillable **Description** The type of application experience in name:experience:form format. **Possible Values** Name • APP_BUILDER • CUSTOM • S1 • SFX Experience • BROWSER • HYBRID Form • DESKTOP • PHONE • TABLET

Field	Details
`DeviceSessionId`	**Type** string **Properties** Nillable **Description** The unique identifier of the user's session based on page load time. When the user reloads a page, a new session is started.
`Duration`	**Type** double **Properties** Nillable **Description** The duration in milliseconds since the page start time.
`EffectivePageTime`	**Type** double **Properties** Nillable **Description** Indicates how many milliseconds it took for the page to load before a user could interact with the page's functionality. Multiple factors can affect effective page time, such as network speed, hardware performance, or page complexity.
`EffectivePageTimeDeviationErrorType`	**Type** string **Properties** Nillable **Description** Indicates the origin of an error. This field is populated when EffectivePageTimeDeviationReason contains the `PageHasError` value. This field is available in API version 58.0 and later. **Possible Values** • `Custom`—An error originating from the customer's system or network. • `System`—An error originating in Salesforce.
`EffectivePageTimeDeviationReason`	**Type** string **Properties** Nillable

Field	Details
	Description The reason for deviation in page loading time. This field is available in API version 58.0 and later. **Possible Values** • `PageInDom`—The page was loaded from a cache. • `PageHasError`—An undefined page loading error occurred. • `PageNotLoaded`—If a customer navigates away from a page while loading processes are in progress, the page doesn't finish loading. • `PreviousPageNotLoaded`—When navigating to a new page, and the previous page hasn't completed loading, the next page is considered to have a deviation. Incomplete loading processes on a previous page can affect how the next page loads. • `InteractionsBeforePageLoaded`—A user interacts with a page element before the page is fully loaded. • `PageInBackgroundBeforeLoaded`—A background loading process runs on a page. Background processes can run when users don't interact with a page, such as when they navigate to another browser tab.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the specified URI event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`HasEffectivePageTimeDeviation`	**Type** boolean **Description** When a deviation is detected, `EffectivePageTimeDeviation` records `true`. The default value is `false`.
`LoginKey`	**Type** string

Field	Details
	Properties Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `8gHOMQu+xvjCmRUt`.
`Operation`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The operation being performed on the entity. For example, `Read, Create, Update`, or `Delete`. Create and update operations are captured in pairs; that is, expect two event records for each operation. The first record represents the start of the operation, and the second record represents whether the operation was successful or not. If there isn't a second event recorded for a create or update operation, the user canceled the operation or the operation failed with client-side validation. For example, when a required field is empty.
`OsName`	**Type** string **Properties** Nillable **Description** The operating system name.
`OsVersion`	**Type** string **Properties** Nillable **Description** The operating system version.
`PageStartTime`	**Type** dateTime **Properties** Nillable **Description** The time when the page was initially loaded, measured in milliseconds.

Field	Details
	Example `1471564788642`
`PageUrl`	**Type** url **Properties** Nillable **Description** Relative URL of the top-level Lightning Experience or Salesforce mobile app page that the user opened. The page can contain one or more Lightning components. Multiple record IDs can be associated with `PageUrl`. **Example** `/sObject/0064100000JXITSAA5/view`
`PreviousPageAppName`	**Type** string **Properties** Nillable **Description** The internal name of the previous application that the user accessed from the App Launcher.
`PreviousPageEntityId`	**Type** reference **Properties** Nillable **Description** The unique previous page entity identifier of the event.
`PreviousPageEntityType`	**Type** string **Properties** Nillable **Description** The previous page entity type of the event.
`PreviousPageUrl`	**Type** url **Properties** Nillable **Description** The relative URL of the previous Lightning Experience or Salesforce mobile app page that the user opened.

Field	Details
	Example `/sObject/006410000`
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The API name of the objects referenced by the URI.
`RecordId`	**Type** reference **Properties** Nillable **Description** The id of the record being viewed or edited. For example, `001RM000003cjx6YAA`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the EventIdentifier of the related event.
`SdkAppType`	**Type** string **Properties** Nillable **Description** The mobile SDK application type. **Possible Values** • `HYBRID` • `HYBRIDLOCAL` • `HYBRIDREMOTE` • `NATIVE` • `REACTNATIVE`
`SdkAppVersion`	**Type** string

Field	Details
	Properties Nillable **Description** The version of the mobile SDK the application uses.
`SdkVersion`	**Type** string **Properties** Nillable **Description** The mobile SDK application version number. **Example** `5.0`
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. This low level isn't available, or used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string

Field	Details
	Properties Nillable **Description** The source IP address of the client logging in. For example, `126.7.4.2`.
`UserAgent`	**Type** string **Properties** Nillable **Description** The type of client used to make the request (for example, the browser, application, or API) as a string. This field is available in API version 58.0 and later.
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005RM000001ctYJYAY`.
`Username`	**Type** string **Properties** Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`UserType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license. Each `UserType` is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values are: • `CsnOnly`—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • `CspLitePortal`—CSP Lite Portal license. Users whose access is limited because they're organization customers, and they access the application through a customer portal or Experience Cloud site. • `CustomerSuccess`—Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal.

Field	Details
	• `Guest` • `PowerCustomerSuccess`—Power Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • `PowerPartner`—Power Partner license. Users whose access is limited because they're partners and typically access the application through a partner portal or site. • `SelfService` • `Standard`—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses.

Standard SOQL Usage

LightningUriEvent allows filtering over two fields: `EventDate` and `EventIdentifier`. The only supported SOQL functions on the LightningUriEvent object are `WHERE`, `ORDER BY`, and `LIMIT`. In the `WHERE` clause, you can only use comparison operators (<, >, <=, and >=). The != operator isn't supported. In the `ORDER BY` clause, you can only use EventDate `DESC`. Ascending order isn't supported with EventDate, and EventIdentifier sorting isn't supported.

Note: Date functions such as `convertTimeZone()` aren't supported—for example, `SELECT CALENDAR_YEAR(EventDate), Count(Id) FROM UriEvent GROUP BY CALENDAR_YEAR(EventDate)` returns an error. You can use date literals in your queries and some date/time functions like `TODAY()`, `YESTERDAY()`, and `LAST_n_DAYS:1`. However, these functions use comparison operators behind the scenes. Therefore you can only use them in the final expression in the `WHERE` clause.

The following list provides some examples of valid queries:

- **Unfiltered**
  - **Valid**—Contains no `WHERE` clause, so no special rules apply.

```
SELECT EntityType, UserName, UserType
FROM LightningUriEvent
```

- **Filtered on** `EventDate`—you can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.
  - **Valid**—you can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.

```
SELECT EntityType, UserName, UserType
FROM LightningUriEvent
WHERE EventDate>=2014-11-27T14:54:16.000Z
```

Async SOQL Usage

With Async SOQL, you can filter on any field in LightningUriEvent and use any comparison operator in your query.

Find who is accessing Opportunities and related Contacts

```
SELECT EventDate, EventIdentifier, UserName, UserType, Name, EntityType, Operation,
LoginKey, SessionKey FROM LightningUriEvent WHERE RecordId='1000000000001'
```

SEE ALSO:

- UriEvent
- *Big Objects Implementation Guide*

LightningUriEventStream

Detects when a user creates, accesses, updates, or deletes a record in Lightning Experience only. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/LightningUriEventStream`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Note: LightningUriEventStream doesn't track Setup events.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AppName`	**Type** string **Properties** Nillable **Description** The name of the application that the user accessed.
`ConnectionType`	**Type** string **Properties** Nillable **Description** The type of connection. **Possible Values** • `CDMA1x` • `CDMA` • `EDGE` • `EVDO0` • `EVDOA` • `EVDOB` • `GPRS` • `HRPD` • `HSDPA` • `HSUPA` • `LTE` • `WIFI`
`DeviceId`	**Type** string **Properties** Nillable **Description** The unique identifier used to identify a device when tracking events. `DEVICE_ID` is a generated value that's created when the mobile app is initially run after installation.
`DeviceModel`	**Type** string

Field	Details
	Properties Nillable **Description** The name of the device model.
`DevicePlatform`	**Type** string **Properties** Nillable **Description** The type of application experience in `name:experience:form` format. **Possible Values** Name • `APP_BUILDER` • `CUSTOM` • `S1` • `SFX` Experience • `BROWSER` • `HYBRID` Form • `DESKTOP` • `PHONE` • `TABLET`
`DeviceSessionId`	**Type** string **Properties** Nillable **Description** The unique identifier of the user's session based on page load time. When the user reloads a page, a new session is started.
`Duration`	**Type** double **Properties** Nillable **Description** The duration in milliseconds since the page start time.

Field	Details
`EffectivePageTime`	**Type** double **Properties** Nillable **Description** Indicates how many milliseconds it took for the page to load before a user could interact with the page's functionality. Multiple factors can affect effective page time, such as network speed, hardware performance, or page complexity.
`EffectivePageTimeDeviationErrorType`	**Type** string **Properties** Nillable **Description** Indicates the origin of an error. This field is populated when EffectivePageTimeDeviationReason contains the `PageHasError` value. This field is available in API version 58.0 and later. **Possible Values** • `Custom`—An error originating from the customer's system or network. • `System`—An error originating in Salesforce.
`EffectivePageTimeDeviationReason`	**Type** string **Properties** Nillable **Description** The reason for deviation in page loading time. This field is available in API version 58.0 and later. **Possible Values** • `PageInDom`—The page was loaded from a cache. • `PageHasError`—An undefined page loading error occurred. • `PageNotLoaded`—If a customer navigates away from a page while loading processes are in progress, the page doesn't finish loading. • `PreviousPageNotLoaded`—When navigating to a new page, and the previous page hasn't completed loading, the next page is considered to have a deviation. Incomplete loading processes on a previous page can affect how the next page loads. • `InteractionsBeforePageLoaded`—A user interacts with a page element before the page is fully loaded. • `PageInBackgroundBeforeLoaded`—A background loading process runs on a page. Background processes can run when users don't interact with a page, such as when they navigate to another browser tab.

Field	Details
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the specified URI event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`HasEffectivePageTimeDeviation`	**Type** boolean **Description** When a deviation is detected, `EffectivePageTimeDeviation` records `true`. The default value is `false`.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring.For example, 8gHOMQu+xvjCmRUt
`Operation`	**Type** picklist

Field	Details
	Properties Nillable, Restricted picklist **Description** The operation being performed on the entity. For example, `Read, Create, Update`, or `Delete`. Create and update operations are captured in pairs; that is, expect two event records for each operation. The first record represents the start of the operation, and the second record represents whether the operation was successful or not. If there isn't a second event recorded for a create or update operation, then the user canceled the operation, or the operation failed with client-side validation (for example, when a required field is empty).
`OsName`	**Type** string **Properties** Nillable **Description** The operating system name.
`OsVersion`	**Type** string **Properties** Nillable **Description** The operating system version.
`PageStartTime`	**Type** dateTime **Properties** Nillable **Description** The time when the page was initially loaded, measured in milliseconds. **Example** `1471564788642`
`PageUrl`	**Type** url **Properties** Nillable

Field	Details
	Description Relative URL of the top-level Lightning Experience or Salesforce mobile app page that the user opened. The page can contain one or more Lightning components. Multiple record IDs can be associated with `PageUrl`. **Example** `/sObject/0064100000JXITSAA5/view`
`PreviousPageAppName`	**Type** string **Properties** Nillable **Description** The internal name of the previous application that the user accessed from the App Launcher.
`PreviousPageEntityId`	**Type** string **Properties** Nillable **Description** The unique previous page entity identifier of the event.
`PreviousPageEntityType`	**Type** string **Properties** Nillable **Description** The previous page entity type of the event.
`PreviousPageUrl`	**Type** url **Properties** Nillable **Description** The relative URL of the previous Lightning Experience or Salesforce mobile app page that the user opened. **Example** `/sObject/006410000`
`QueriedEntities`	**Type** string **Properties** Nillable

Field	Details
	Description The API name of the objects referenced by the URI.
`RecordId`	**Type** string **Properties** Nillable **Description** The id of the record being viewed or edited. For example, `001RM000003cjx6YAA`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the EventIdentifier of the related event.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SdkAppType`	**Type** string **Properties** Nillable **Description** The mobile SDK application type. **Possible Values** • `HYBRID` • `HYBRIDLOCAL` • `HYBRIDREMOTE` • `NATIVE` • `REACTNATIVE`

Field	Details
`SdkAppVersion`	**Type** string **Properties** Nillable **Description** The version of the mobile SDK the application uses.
`SdkVersion`	**Type** string **Properties** Nillable **Description** The mobile SDK application version number. **Example** 5.0
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.

Field	Details
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client logging in. For example, `126.7.4.2`.
`UserAgent`	**Type** string **Properties** Nillable **Description** The type of client used to make the request (for example, the browser, application, or API) as a string. This field is available in API version 58.0 and later.
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005RM000001ctYJYAY`.
`Username`	**Type** string **Properties** Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`UserType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license. Each `UserType` is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values are: • `CsnOnly`—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • `CspLitePortal`—CSP Lite Portal license. Users whose access is limited because they are organization customers and access the application through a customer portal or an Experience Cloud site.

Field	Details
	• `CustomerSuccess`—Customer Success license. Users whose access is limited because they are organization customers and access the application through a customer portal. • `Guest` • `PowerCustomerSuccess`—Power Customer Success license. Users whose access is limited because they are organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • `PowerPartner`—Power Partner license. Users whose access is limited because they are partners and typically access the application through a partner portal or site. • `SelfService` • `Standard`—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses.

SEE ALSO:

UriEventStream

ListViewEvent

Tracks when users access data with list views using Lightning Experience, Salesforce Classic, or the API. It doesn't track list views of Setup entities. You can use ListViewEvent in a transaction security policy. ListViewEvent is a big object that stores the event data of ListViewEventStream. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Note: For some default list views (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic), the `DeveloperName`, `ListViewId`, and `Name` fields are blank because the list view wasn't explicitly created by a user.

Field	Details
`AppName`	**Type** string **Properties** Nillable

Field	Details
	Description The name of the application that the user accessed. Possible values include `one:one` (browser) and `native:bridge` (mobile app).
`ColumnHeaders`	**Type** string **Properties** Nillable **Description** Comma-separated values of column headers of the list view. These values are the API names, not the labels shown in the UI. For example, `Name, BillingState, Phone,Type, Owner.Alias, CaseNumber, Contact.Name, Subject, Status, Priority, CreatedDate, Owner.NameOrAlias`.
`DeveloperName`	**Type** string **Properties** Nillable **Description** The unique name of the object in the API. This name contains only underscores and alphanumeric characters, and is unique in your org. If blank, the list view is a default list view (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic) and not explicitly created by a user. For example, AllAccounts or AllOpenLeads.
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the transaction security policy, in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time when the specified list view event was captured. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string

Field	Details
	Properties Filter, Sort **Description** The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`EventSource`	**Type** string **Properties** Nillable, Restricted picklist **Description** The source of the event. Possible values are: • `API`—The user generated the list view from an API call. • `Classic`—The user generated the list view from a page in the Salesforce Classic UI. • `Lightning`—The user generated the list view from a page in the Lightning Experience UI.
`ExecutionIdentifier`	**Type** string **Properties** Nillable **Description** When list view execution data is divided into multiple list view events, use this unique identifier to correlate the multiple data chunks. For example, each chunk might have the same `ExecutionIdentifier` of a50a4025-84f2-425d-8af9-2c780869f3b5, enabling you to link them together to get all the data for the list view execution. The `Sequence` field contains the incremental sequence numbers that indicate the order of the multiple events. For more information, see `Sequence`.

`FilterCriteria`	**Type**
	json
	Properties
	Nillable
	Description
	A JSON string that represents the list view's filter criteria at the time the event was captured.
	Example
	Here's a JSON string that represents filter criteria for an accounts list view. The list view shows only accounts of type "Prospect".
	`{"whereCondition":`
	`{"type":"soqlCondition","field":"Type",`

Field	Details

	<pre>"operator":"equals","values":["'Prospect'"]}
	}</pre>

ListViewId	**Type** reference **Properties** Nillable **Description** The ID of the list view associated with this event. If blank, the list view is a default list view (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic) and not explicitly created by a user. For example, 00BB0000001c73kMAA.
LoginHistoryId	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular series of list view events. This field is also available in the LoginEvent, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. For example, 0YaB000002knVQLKA2.
LoginKey	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
Name	**Type** string **Properties** Nillable **Description** The display name of the list view. If blank, the list view is a default list view (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic) and not explicitly created by a user. For example, All Accounts and All Open Leads.
NumberOfColumns	**Type** int

Field	Details
	Properties Nillable **Description** The number of columns in the list view.
`OrderBy`	**Type** string **Properties** Nillable **Description** The column that the list view is sorted by. For example, if a list view of accounts is sorted alphabetically by name, the `OrderBy` value is `[Name ASC NULLS FIRST, Id ASC NULLS FIRST]`. If the list is sorted alphabetically by type, the `OrderBy` value is `[Type ASC NULLS FIRST, Id ASC NULLS FIRST]`.
`OwnerId`	**Type** reference **Properties** Nillable **Description** The ID of the org or user who owns the list view. If the list view wasn't saved, this value is the same as `UserId`. For example, 005B0000001vURvIAM.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction security policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger.

Field	Details
	• `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, such as logging in from a recognized device, that activity is approved from the trusted location for as long as the location is trusted. • `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response. • `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but will retry. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified. This field isn't populated until all transaction security policies are processed for the real-time event.
`QueriedEntities`	**Type** string **Properties** Nillable

Field	Details
	Description The type of entities in the list view. For example, `Opportunity, Lead, Account,` or `Case`. Can also include custom objects.
`Records`	**Type** json **Properties** Nillable **Description** A JSON string that represents the list view's data. For example, `{"totalSize":1,"rows":[{"datacells":["005B0000001vURv","001B000000fewai"]}]}`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`RowsProcessed`	**Type** double **Properties** Nillable **Description** The total number of rows returned in the list view. When list data is divided into multiple list view events, this value is the same for all data chunks.
`Scope`	**Type** string **Properties** Nillable **Description** Represents the filter criteria for the list view. Possible values are: • `Delegated`—Records delegated to another user for action; for example, a delegated task.

Field	Details
	• `Everything`—All records, for example All Opportunities. • `Mine`—Records owned by the user running the list view, for example My Opportunities. • `MineAndMyGroups`—Records owned by the user running the list view, and records assigned to the user's queues. • `MyTerritory`—Records in the territory of the user seeing the list view. This option is available if territory management is enabled for your org. • `MyTeamTerritory`—Records in the territory of the team of the user seeing the list view. This option is available if territory management is enabled for your org. • `Queue`—Records assigned to a queue. • `Team`—Records assigned to a team.
`Sequence`	**Type** int **Properties** Nillable **Description** Incremental sequence number that indicates the order of multiple events that result from a given list view execution. When a list view execution returns many records, Salesforce splits this data into chunks based on the size of the records, and then creates multiple correlated ListViewEvents. The field values in each of these correlated ListViewEvents are the same, except for `Records`, which contains the different data chunks, and `Sequence`, which identifies each chunk in order. Every list view execution has a unique `ExecutionIdentifier` value to differentiate it from other list view executions. To view all the data chunks from a single list view execution, use the `Sequence` and `ExecutionIdentifier` fields in combination. For more information, `ExecutionIdentifier`.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are:

Field	Details
	• `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level will experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** The username in the format of `user@company.com` at the time the event was created.

Standard SOQL Usage

You can filter on two ordered fields: `EventDate` and `EventIdentifier`.

Example

```
SELECT Username, QueriedEntities, ListViewData, PolicyOutcome, Name FROM ListViewEvent
```

Async SOQL Usage

With Async SOQL, you can filter on any field in ListViewEvent and use any comparison operator in your query.

Example: Find all list views that users ran against Patent__c

```
SELECT EventDate, EventIdentifier, PolicyOutcome, EvaluationTime, ListViewId, Name FROM
ListViewEvent WHERE QueriedEntities='Patent__c'
```

SEE ALSO:

Big Objects Implementation Guide

ListViewEventStream

Tracks actions related to list views in Lightning Experience, Salesforce Classic, or the API. For example, the event captures when a user runs or exports a list view. It doesn't capture list view events of Setup entities. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/ListViewEventStream`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Note: For some default list views (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic), the `DeveloperName`, `ListViewId`, and `Name` fields are blank because the list view wasn't explicitly created by a user.

Field	Details
`AppName`	**Type** string **Properties** Nillable **Description** The name of the application that the user accessed. Possible values include `one:one` (browser) and `native:bridge` (mobile app).
`ColumnHeaders`	**Type** string **Properties** Nillable **Description** Comma-separated values of column headers of the list view. These values are the API names, not the labels shown in the UI. For example, `Name, BillingState, Phone,Type, Owner.Alias, CaseNumber, Contact.Name, Subject, Status, Priority, CreatedDate, Owner.NameOrAlias`.
`DeveloperName`	**Type** string **Properties** Nillable **Description** The unique name of the object in the API. This name contains only underscores and alphanumeric characters, and is unique in your org. If blank, the list view is a default list view (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic) and not explicitly created by a user. For example, AllAccounts or AllOpenLeads.
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the transaction security policy, in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Filter, Sort

Field	Details
	Description The time when the specified list view event was captured. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`EventSource`	**Type** string **Properties** Nillable, Restricted picklist **Description** The source of the event. Possible values are: • `API`—The user generated the list view from an API call. • `Classic`—The user generated the list view from a page in the Salesforce Classic UI. • `Lightning`—The user generated the list view from a page in the Lightning Experience UI.
`ExecutionIdentifier`	**Type** string **Properties** Nillable **Description** When list view execution data is divided into multiple list view events, use this unique identifier to correlate the multiple data chunks. For example, each chunk might have the same `ExecutionIdentifier` of a50a4025-84f2-425d-8af9-2c780869f3b5, enabling you to link them together to get all the data for the list view execution. The `Sequence`

Field	Details
	field contains the incremental sequence numbers that indicate the order of the multiple events. For more information, see `Sequence`.
`FilterCriteria`	**Type** json **Properties** Nillable **Description** A JSON string that represents the list view's filter criteria at the time the event was captured. **Example** Here's a JSON string that represents filter criteria for an accounts list view. The list view shows only accounts of type "Prospect". `{"whereCondition": {"type":"soqlCondition","field":"Type", "operator":"equals","values":["'Prospect'"]} }`
`ListViewId`	**Type** reference **Properties** Nillable **Description** The ID of the list view associated with this event. If blank, the list view is a default list view (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic) and not explicitly created by a user. For example, 00BB0000001c73kMAA.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular series of list view events. This field is also available in the LoginEvent, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. For example, 0YaB000002knVQLKA2.
`LoginKey`	**Type** string **Properties** Nillable

Field	Details
	Description The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`Name`	**Type** string **Properties** Nillable **Description** The display name of the list view. If blank, the list view is a default list view (such as the list view that displays when a user clicks the Groups tab in Salesforce Classic) and not explicitly created by a user. For example, All Accounts and All Open Leads.
`NumberOfColumns`	**Type** int **Properties** Nillable **Description** The number of columns in the list view.
`OrderBy`	**Type** string **Properties** Nillable **Description** The column that the list view is sorted by. For example, if a list view of accounts is sorted alphabetically by name, the `OrderBy` value is `[Name ASC NULLS FIRST, Id ASC NULLS FIRST]`. If the list is sorted alphabetically by type, the `OrderBy` value is `[Type ASC NULLS FIRST, Id ASC NULLS FIRST]`.
`OwnerId`	**Type** reference **Properties** Nillable **Description** The ID of the org or user who owns the list view. If the list view wasn't saved, this value is the same as `UserId`. For example, 005B0000001vURvIAM.
`PolicyId`	**Type** reference

Field	Details
	Properties Nillable **Description** The ID of the transaction security policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, such as logging in from a recognized device, that activity is approved from the trusted location for as long as the location is trusted. • `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response.

Field	Details
	• `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but will retry. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified. This field isn't populated until all transaction security policies are processed for the real-time event.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The type of entities in the list view. For example, `Opportunity, Lead, Account,` or `Case`. Can also include custom objects.
`Records`	**Type** json **Properties** Nillable **Description** A JSON string that represents the list view's data. For example, `{"totalSize":1,"rows":[{"datacells":["005B0000001vURv","001B000000fewai"]}]}`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`ReplayId`	**Type** string

Field	Details
	Properties Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`RowsProcessed`	**Type** double **Properties** Nillable **Description** The total number of rows returned in the list view. When list data is divided into multiple list view events, this value is the same for all data chunks.
`Scope`	**Type** string **Properties** Nillable **Description** Represents the filter criteria for the list view. Possible values are: • `Delegated`—Records delegated to another user for action; for example, a delegated task. • `Everything`—All records, for example All Opportunities. • `Mine`—Records owned by the user running the list view, for example My Opportunities. • `MineAndMyGroups`—Records owned by the user running the list view, and records assigned to the user's queues. • `MyTerritory`—Records in the territory of the user seeing the list view. This option is available if territory management is enabled for your org. • `MyTeamTerritory`—Records in the territory of the team of the user seeing the list view. This option is available if territory management is enabled for your org. • `Queue`—Records assigned to a queue. • `Team`—Records assigned to a team.
`Sequence`	**Type** int **Properties** Nillable **Description** Incremental sequence number that indicates the order of multiple events that result from a given list view execution.

Field	Details
	When a list view execution returns many records, Salesforce splits this data into chunks based on the size of the records, and then creates multiple correlated ListViewEventStreams. The field values in each of these correlated ListViewEventStreams are the same, except for `Records`, which contains the different data chunks, and `Sequence`, which identifies each chunk in order. Every list view execution has a unique `ExecutionIdentifier` value to differentiate it from other list view executions. To view all the data chunks from a single list view execution, use the `Sequence` and `ExecutionIdentifier` fields in combination. For more information, see `ExecutionIdentifier`.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level will experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.

Field	Details
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** The username in the format of `user@company.com` at the time the event was created.

LoginAsEvent

LoginAsEvent tracks when an admin logs in as another user in your org. In Real-Time Event Monitoring, it captures events for org admins and Experience Cloud sites only. LoginAsEvent is a big object that stores the event data of LoginAsEventStream. This object is available in API version 46.0 and later.

Supported Calls

describeSObjects(), `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`Application`	**Type** string **Properties** Nillable **Description** The application name in English. For example, Salesforce Internal Application, or Microsoft SOAP Toolkit.
`Browser`	**Type** string

Field	Details
	Properties Nillable **Description** The browser name and version if known. Possible values for the browser name are: • Chrome • Firefox • Safari • Unknown For example, "Chrome 77".
`DelegatedOrganizationId`	**Type** string **Properties** Nillable **Description** Organization Id of the admin who performs logs in as another user. For example, 00Dxx0000001gEH
`DelegatedUsername`	**Type** string **Properties** Nillable **Description** Username of the admin who logs in as another user. For example, `admin@company.com`
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time and date of the event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique identifier for each record in LoginAsEvent. Use this field as the primary key in your queries.

Field	Details
`LoginAsCategory`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Represents how the user logs in as another user. Possible values are: • OrgAdmin—An administrator logs in to Salesforce as an individual user. Depending on your org settings, the individual user grants login access to the administrator. • Community—A user who has been granted access to a Salesforce Experience Cloud site logs in.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** The ID from the LoginHistory entity associated with this login event. Tracks a user session so you can correlate user activity with a particular login instance. For example, 0Yaxx0000000019.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, 8gHOMQu+xvjCmRUt.
`LoginType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The event's type of login. For example, "Application."
`Platform`	**Type** string **Properties** Nillable **Description** The platform name and version that are used during the login event. If no platform name is available, "Unknown" is returned. Platform names are in English. For example, "Mac OSX".

Field	Details
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For LoginAsEvent, this field is usually null because the event is captured before a session is created.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • HIGH_ASSURANCE - A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • LOW - The user's security level for the current session meets the lowest requirements. Note: This low level is available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • STANDARD - The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client logging in. For example, 126.7.4.2.
`TargetUrl`	**Type** string **Properties** Nillable **Description** The URL redirected to after logging in as another user succeeds.

Field	Details
`UserId`	**Type** reference **Properties** Nillable **Description** Unique ID that identifies the user who is being logged in as by the admin. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** Username of the user who is being logged in as by the admin, in the format of `someuser@company.com`.
`UserType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license of the user who is being logged in as by the admin. Each `UserType` is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values are: • `CsnOnly`—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • `CspLitePortal`—CSP Lite Portal license. Users whose access is limited because they're organization customers and access the application through a customer portal or an Experience Cloud site. • `CustomerSuccess`—Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. • `Guest`—Users whose access is limited so that your customers can view and interact with your site without logging in. • `PowerCustomerSuccess`—Power Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • `PowerPartner`—Power Partner license. Users whose access is limited because they're partners and typically access the application through a partner portal or site. • `SelfService`—Users whose access is limited because they're organization customers and access the application through a self-service portal.

Field	Details
	• `Standard`—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses, and admins for this org.

Standard SOQL Usage

Currently, the only supported SOQL function on LoginAsEvent is `WHERE`, and you can only use comparison operators (=, <, >, <=, and >=) on the final expression in a `WHERE` clause. The != operator isn't supported.

Note: Date functions such as `convertTimezone()` aren't supported. For example, `SELECT CALENDAR_YEAR(EventDate), Count(EventIdentifier) FROM LoginAsEvent GROUP BY CALENDAR_YEAR(EventDate)` returns an error. You can use date literals in your queries and some date and date/time functions like `TODAY`, `YESTERDAY`, and `LAST_n_DAYS:1`. However, these functions use comparison operators behind the scenes. This means you can only use them in the final expression of a `WHERE` clause.

LoginAsEvent allows filtering over two ordered fields: `EventDate` and `EventIdentifier`. There's a catch here; your query doesn't work unless you use the correct order and combination of these fields. The following list provides some examples of valid and invalid queries:

- **Unfiltered**
  - **Valid**—Contains no `WHERE` clause, so no special rules apply.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginHistoryId, UserId
FROM LoginAsEvent
```

- **Filtered on** `EventDate`
  - **Valid**—You can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginHistoryId, UserId
FROM LoginAsEvent
WHERE EventDate<=2014-11-27T14:54:16.000Z
```

  - **Valid**—You can filter on `EventDate` using date literals.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginHistoryId, UserId
FROM LoginAsEvent
WHERE EventDate<=TODAY
```

- **Filtered on** `EventDate` **and** `EventIdentifier`
  - **Valid**—Successful queries on LoginAsEvent filter over both fields.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginHistoryId, UserId
FROM LoginAsEvent
WHERE EventDate=2014-11-27T14:54:16.000Z and
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

- **Invalid**—Queries on LoginAsEvent with `EventDate` and standard date literals.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginHistoryId, UserId
FROM LoginAsEvent
WHERE EventDate=TODAY and EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

- **Invalid**—Filtering only on `EventDate` with <= or >= operator and `EventIdentifier` field isn't supported.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginHistoryId, UserId
FROM LoginAsEvent
WHERE EventDate<=2014-11-27T14:54:16.000Z and
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

Async SOQL Usage

With Async SOQL, you can filter on any field in LoginAsEvent and use any comparison operator in your query.

Example: Get yesterday's LoginAs events where an Org Admin is logging into the portal as another user.

```
SELECT DelegatedUsername, DelegatedOrganizationId, EventDate, LoginAsCategory,
LoginHistoryId, LoginType, SourceIp, TargetUrl, UserId, Username, UserType FROM
LoginAsEvent WHERE EventDate=Yesterday AND LoginAsCategory='OrgAdmin'
```

SEE ALSO:

Big Objects Implementation Guide

LoginAsEventStream

LoginAsEvent tracks when an admin logs in as another user in your org. In Real-Time Event Monitoring, it captures events for org admins and Experience Cloud site only. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/LoginAsEventStream`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`Application`	**Type** string **Properties** Nillable **Description** The application name in English. For example, Salesforce Internal Application, or Microsoft SOAP Toolkit.
`Browser`	**Type** string **Properties** Nillable **Description** The browser name and version if known. Possible values for the browser name are: • Chrome • Firefox • Safari • Unknown For example, “Chrome 77”.
`DelegatedOrganizationId`	**Type** string **Properties** Nillable **Description** Organization Id of the user who is logging in as another user. For example, 00Dxx0000001gEH
`DelegatedUsername`	**Type** string **Properties** Nillable

Field	Details
	Description Username of the admin who is logging in as another user. For example, `admin@company.com`
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time and date of the event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object. Also, use this field as the primary key in your queries.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginAsCategory`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Represents how the user logs in as another user. Possible values are: • OrgAdmin—An administrator logs in to Salesforce as an individual user. Depending on your org settings, the individual user grants login access to the administrator. • Community—A user who has been granted access to a Salesforce Experience Cloud site logs in.
`LoginHistoryId`	**Type** reference

Field	Details
	Properties Nillable **Description** Tracks a user session so you can correlate user activity with a particular login instance. The ID from the LoginHistory entity associated with this login event. For example, 0Yaxx0000000019.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, 8gHOMQu+xvjCmRUt.
`LoginType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of login used to access the session. See the LoginType field of LoginHistory in the Object Reference guide for the list of possible values.
`Platform`	**Type** string **Properties** Nillable **Description** The platform name and version that are used during the login event. If no platform name is available, "Unknown" is returned. Platform names are in English. For example, "Mac OSX".
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SessionKey`	**Type** string

Field	Details
	Properties Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For LoginAsEvent, this field is usually null because the event is captured before a session is created.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • HIGH_ASSURANCE - A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • LOW - The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • STANDARD - The user's security level for the current session meets the Standard requirements set in the current organization Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client logging in. For example, 126.7.4.2.
`TargetUrl`	**Type** string **Properties** Nillable **Description** The URL redirected to after logging in as another user succeeds.
`UserId`	**Type** reference

Field	Details
	Properties Nillable **Description** Unique ID that identifies the user who is being logged in as by the admin. For example, `005000000000123`.
`Username`	**Type** string **Properties** Nillable **Description** Username of the user who is being logged in as by the admin, in the format of `admin@company.com`.
`UserType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license of the user who is being logged in as by the admin. Each `UserType` is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values: • CsnOnly—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • CspLitePortal—CSP Lite Portal license. Users whose access is limited because they're organization customers and access the application through a customer portal or an Experience Cloud site. • CustomerSuccess—Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. • Guest • PowerCustomerSuccess—Power Customer Success license. Users whose access is limited because they'reare organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • PowerPartner—Power Partner license. Users whose access is limited because they're partners and typically access the application through a partner portal or site. • SelfService • Standard—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses, and admins for this org.

LoginEvent

LoginEvent tracks the login activity of users who log in to Salesforce. You can use LoginEvent in a transaction security policy. LoginEvent is a big object that stores the event data of LoginEventStream. This object is available in API version 36.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

- Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Note: LoginEvent doesn't track login activity after login rates exceed the limit. This condition applies to all users, including integration users and internal users who log in to Salesforce.

Fields

Field	Details
`AdditionalInfo`	**Type** string **Properties** Nillable **Description** JSON serialization of additional information that's captured from the `HTTP` headers during a login request. For example, `{"field1": "value1","field2": "value2"}`. See Working with `AdditionalInfo` on page 437.
`ApiType`	**Type** string **Properties** Nillable **Description** The type of API that's used to log in. Values include: • `SOAP Enterprise` • `SOAP Partner` • `REST API`
`ApiVersion`	**Type** string **Properties** Nillable **Description** The version number of the API. If no version number is available, "Unknown" is returned.

Field	Details
`Application`	**Type** string **Properties** Nillable **Description** The application used to access the org. Possible values include: • `AppExchange` • `Browser` • `Salesforce for iOS` • `Salesforce Developers API Explorer` • `N/A`
`AuthMethodReference`	**Type** string **Properties** Nillable **Description** The authentication method used by a third-party identification provider for an OpenID Connect single sign-on protocol. This field is available in API version 51.0 and later.
`AuthServiceId`	**Type** reference **Properties** Nillable **Description** The 18-character ID for an authentication service for a login event. For example, you can use this field to identify the SAML or authentication provider configuration with which the user logged in.
`Browser`	**Type** string **Properties** Nillable **Description** The browser name and version if known. Possible values for the browser name are: • Chrome • Firefox • Safari • Unknown For example, “Chrome 77”.

Field	Details
`CipherSuite`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The TLS cipher suite used for the login. Values are OpenSSL-style cipher suite names, with hyphen delimiters, for example, `ECDHE-RSA-AES256-GCM-SHA384`. Available in API version 37.0 and later.
`City`	**Type** string **Properties** Nillable **Description** The city where the user's IP address is physically located. This value isn't localized. This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`ClientVersion`	**Type** string **Properties** Nillable **Description** The version number of the login client. If no version number is available, "Unknown" is returned.
`Country`	**Type** string **Properties** Nillable **Description** The country where the user's IP address is physically located. This value isn't localized.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`CountryIso`	**Type** string **Properties** Nillable **Description** The ISO 3166 code for the country where the user's IP address is physically located. For more information, see Country Codes - ISO 3166. This field is available in API version 37.0 and later.

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the transaction security policy, in milliseconds. This field is available in API version 46.0 and later.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The login time of the specified event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique identifier for each record in LoginEvent. Use this field as the primary key in your queries. Available in API version 42.0 and later.
`ForwardedForIp`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The value in the `X-Forwarded-For` header of HTTP requests sent by the client. For logins that use one or more HTTP proxies, the `X-Forwarded-For` header is sometimes used to store the origin IP and all proxy IPs. The `ForwardedForIp` field stores whatever value the client sends, which might not be an IP address. The maximum length is 256 characters. Longer values are truncated. The `ForwardedForIp` field isn't populated for logins completed via OAuth flows or single sign-on (SSO). Available in API version 61.0 and later.
`HttpMethod`	**Type** picklist **Properties** Nillable, Restricted picklist

Field	Details
	Description The HTTP method of the login request; possible values are `GET`, `POST`, and `Unknown`.
`LoginGeoId`	**Type** reference **Properties** Nillable **Description** The Salesforce ID of the LoginGeo object associated with the login user's IP address. For example, 04FB000001TvhiPMAR.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular login instance. This field is also available on the LoginHistory, AuthSession, and other objects, making it easier to trace events back to a user's original authentication. For example, 0YaB000002knVQLKA2.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. This field is available in API version 46.0 and later. For example, lUqjLPQTWRdvRG4.
`LoginLatitude`	**Type** double **Properties** Nillable **Description** The latitude where the user's IP address is physically located.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`LoginLongitude`	**Type** double **Properties** Nillable

Field	Details
	Description The longitude where the user's IP address is physically located.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`LoginSubType`	**Type** picklist **Properties** Nillable, Restricted picklist, **Description** The type of login flow used. See the `LoginSubType` field of LoginHistory in the Object Reference guide for the list of possible values. Label is **Login Subtype**.
`LoginType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of login used to access the session. See the LoginType field of LoginHistory in the Object Reference guide for the list of possible values.
`LoginUrl`	**Type** string **Properties** Nillable **Description** The URL of the login host from which the request is coming. For example, ***`yourInstance`***`.salesforce.com`.
`NetworkId`	**Type** reference **Properties** Nillable **Description** The ID of the Experience Cloud site that the user is logging in to. This field is available if Salesforce Experience Cloud is enabled for your organization.
`Platform`	**Type** string

Field	Details
	Properties Nillable **Description** The operating system name and version that are used during the login event. If no platform name is available, "Unknown" is returned. For example, Mac OSX or iOS/Mac.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction security policy associated with this event. This field is available in API version 46.0 and later. For example, 0NIB000000000KOOAY.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, that activity is approved from the trusted location for as long as the location is trusted. An example of a particular activity is logging in from a recognized device. • `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity.

Field	Details
	• `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response. • `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but will retry. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified. This field is available in API version 46.0 and later.
`PostalCode`	**Type** string **Properties** Nillable **Description** The postal code where the user's IP address is physically located. This value isn't localized.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`RemoteIdentifier`	**Type** string

Field	Details
	Properties Nillable **Description** Reserved for future use.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For LoginEvent, this field is often null because the event is captured before a session is created. For example, vMASKIU6AxEr+Op5. This field is available in API version 46.0 and later.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level isn't available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels. This field is available in API version 42.0 and later.
`SourceIp`	**Type** string **Properties** Nillable **Description** The IP address of the incoming client request that first reaches Salesforce during a login. For example, `126.7.4.2`. For clients that redirect through one or more HTTP proxies, this field stores the IP

Field	Details
	address of the first proxy to reach Salesforce. To better identify the origin IP for these cases, check the `ForwardedForIp` field instead.
`Status`	**Type** string **Properties** Nillable **Description** Displays the status of the attempted login. Status is either success or a reason for failure.
`Subdivision`	**Type** string **Properties** Nillable **Description** The name of the subdivision where the user's IP address is physically located. In the U.S., this value is usually the state name (for example, Pennsylvania). This value isn't localized.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`TlsProtocol`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The TLS protocol version used for the login. Available in API version 37.0 and later. Valid values are: • `TLS 1.0` • `TLS 1.1` • `TLS 1.2` • `TLS 1.3` • `Unknown`
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005000000000123`.
`Username`	**Type** string

Field	Details
	Properties Nillable **Description** The username in the format of `user@company.com`.
`UserType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license. Each `UserType` is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values are: • `CsnOnly`—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • `CspLitePortal`—CSP Lite Portal license. Users whose access is limited because they're organization customers and access the application through a customer portal or Experience Cloud site. • `CustomerSuccess`—Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. • `Guest` • `PowerCustomerSuccess`—Power Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • `PowerPartner`—Power Partner license. Users whose access is limited because they're partners and typically access the application through a partner portal or site. • `SelfService` • `Standard`—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses. This field is available only in the Real-Time Event Monitoring in API version 42.0 and later.

Working with `AdditionalInfo`

`AdditionalInfo` enables you to extend the login event with custom data that can be queried later. For example, you can capture a correlation ID when a user logs in from an external system that shares that unique ID. This process enables tracking logins across systems. To store data with LoginEvent, begin all `AdditionalInfo` field names with `x-sfdc-addinfo-{`***`fieldname`***`}`. For example, a valid field assignment is `x-sfdc-addinfo-correlation_id = ABC123` where `x-sfdc-addinfo-correlation_id` is the field name and `ABC123` is the field value.

When defining field names, note the following:

- `x-sfdc-addinfo-` is case-insensitive. `x-sfdc-addinfo-{`***`field name`***`}` is the same as `X-SFDC-ADDINFO-{`***`FIELD NAME`***`}`.

- Fields can contain only alphanumeric and "_" (underscore) characters.
- Field names must be from 2 and 29 characters in length, excluding `x-sfdc-addinfo-`.
- Field names that don't start with `x-sfdc-addinfo-` are ignored.
- Field names that contain invalid characters after `x-sfdc-addinfo-` can cause an HTTP 400 Bad Request error.
- Only the first 30 valid field names are stored in `AdditionalInfo`. Field names aren't necessarily stored in the same order in which they were passed to authentication.

When determining field values, keep the following in mind:

- You can't use existing API field names as `AdditionalInfo` names in the HTTP header. If the `AdditionalInfo` name conflicts with an object's API name, the field value isn't stored. For example, the HTTP header `X-SFDC-ADDINFO-UserId='abc123'` doesn't get stored in `AdditionalInfo`.
- Additional field values can contain only alphanumeric, "_" and "-" characters.
- Field values must be 255 characters in length or fewer. If a field value exceeds 255 characters, only the first 255 characters are stored and the rest are truncated.
- Field values that contain invalid characters are saved with a field header of Empty String ("").
- Only the first 30 valid field names are stored in the `AdditionalInfo` field. They aren't guaranteed to be stored in the same order that they were passed into the authentication.
- When `AggregationFieldName` is `SourceIp`, you can't filter on `AggregationFieldValue` if its value is *`Salesforce.com IP`*.

How to Pass Additional Information by Using HTTP with cURL

Here's an example of passing additional information via the command line.

```
curl https://yourInstance.salesforce.com/services/oauth2/token -d "grant_type=password"
-d
"client_id=3MVG9PhR6g6B7ps4RF_kNPoWSxVQstrazijsE8njPtkpUzVPPffzy8
jIoRE6q9rPznNtlsqbP9ob8kUfMjXXX" -d "client_secret=4180313776440635XXX" -d
"username=user@company.com" -d "password=123456" -H "X-PrettyPrint:1" -H
"x-sfdc-addinfo-correlationid:
d18c5a3f-4fba-47bd-bbf8-6bb9a1786624"
```

How to Pass Additional Information in Java

Here's an example of passing additional information in Java.

```
//adding additional info headers ..
Map<String, String> httpHeaders = new HashMap<String,String>();
httpHeaders.put("x-sfdc-addinfo-fieldname1" /* additional info field*/ ,
"d18c5a3f-4fba-47bd-bbf8-6bb9a1786624" /* value*/);
httpHeaders.put("x-sfdc-addinfo-fieldname2" /* additional info field*/ ,
"d18c5a3f-4fba-47bd-bbf8-6bb9a1786624" /* value*/);
ConnectorConfig config = new ConnectorConfig();
config.setUsername(userId);
config.setPassword(passwd);
config.setAuthEndpoint(authEndPoint);
config.setProxy(proxyHost, proxyPort);
//setting additional info headers
for (Map.Entry<String, String> entry : httpHeaders.entrySet()) {
config.setRequestHeader(entry.getKey(), entry.getValue());
```

```
}
// Set the username and password if your proxy must be authenticated
9
LoginEvent
config.setProxyUsername(proxyUsername);
config.setProxyPassword(proxyPassword);
try {
EnterpriseConnection connection = new EnterpriseConnection(config);
// etc.
} catch (ConnectionException ce) {
ce.printStackTrace();
}
```

Standard SOQL Usage

Currently, the only supported SOQL function on LoginEvent is `WHERE`, and you can only use comparison operators (=, <, >, <=, and >=) on the final expression in a `WHERE` clause. The `!=` operator isn't supported.

Note: Date functions such as `convertTimezone()` aren't supported. For example, `SELECT CALENDAR_YEAR(EventDate), Count(EventIdentifier) FROM LoginEvent GROUP BY CALENDAR_YEAR(EventDate)` returns an error. You can use date literals in your queries and some date and date/time functions like `TODAY`, `YESTERDAY`, and `LAST_n_DAYS:1`. However, these functions use comparison operators behind the scenes, which means you can only use them in the final expression of a `WHERE` clause.

LoginEvent allows filtering over two ordered fields: `EventDate` and `EventIdentifier`. There's a catch here; your query doesn't work unless you use the correct order and combination of these fields. The following list provides some examples of valid and invalid queries:

- **Unfiltered**
  - **Valid**—Contains no `WHERE` clause, so no special rules apply.

    ```
    SELECT Application, Browser, EventDate, EventIdentifier, LoginUrl, UserId
    FROM LoginEvent
    ```

- **Filtered on** `EventDate`
  - **Valid**—You can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.

    ```
    SELECT Application, Browser, EventDate, EventIdentifier, LoginUrl, UserId
    FROM LoginEvent
    WHERE EventDate<=2014-11-27T14:54:16.000Z
    ```

  - **Valid**—You can filter on `EventDate` using date literals.

    ```
    SELECT Application, Browser, EventDate, EventIdentifier, LoginUrl, UserId
    FROM LoginEvent
    WHERE EventDate<=TODAY
    ```

- **Filtered on** `EventDate` **and** `EventIdentifier`
  - **Valid**—Successful queries on LoginEvent filter over both fields.

    ```
    SELECT Application, Browser, EventDate, EventIdentifier, LoginUrl, UserId
    FROM LoginEvent
    ```

```
WHERE EventDate=2014-11-27T14:54:16.000Z and
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

- **Invalid**—Queries on LoginEvent with `EventDate` and standard date literals.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginUrl, UserId
FROM LoginEvent
WHERE EventDate=TODAY and EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

- **Invalid**—Filtering only on `EventDate` with <= or >= operator and `EventIdentifier` field isn't supported.

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginUrl, UserId
FROM LoginEvent
WHERE EventDate<=2014-11-27T14:54:16.000Z and
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

Async SOQL Usage

With Async SOQL, you can filter on any field in LoginEvent and use any comparison operator in your query.

Example: Get Yesterday's Successful Logins

```
SELECT Application, Browser, EventDate, EventIdentifier, LoginUrl, UserId FROM LoginEvent
WHERE EventDate<Yesterday AND Status='Success'
```

SEE ALSO:

LoginEventStream

Big Objects Implementation Guide

LoginEventStream

LoginEventStream tracks login activity of users who log in to Salesforce. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/LoginEventStream`

Special Access Rules

- Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.
- LoginEventStream doesn't track login activity, including login rate exceeding the limit, for integration or internal users who log in to Salesforce.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AdditionalInfo`	**Type** string **Properties** Nillable **Description** JSON serialization of additional information that's captured from the `HTTP` headers during a login request. For example, `{"field1": "value1","field2": "value2"}`.
`ApiType`	**Type** string **Properties** Nillable **Description** The type of API that's used to log in. Values include: • `SOAP Enterprise` • `SOAP Partner` • `REST API`
`ApiVersion`	**Type** string **Properties** Nillable **Description** The version number of the API. If no version number is available, "Unknown" is returned.
`Application`	**Type** string

Field	Details
	Properties Nillable **Description** The application used to access the org. Possible values include: • `AppExchange` • `Browser` • `Salesforce for iOS` • `Salesforce Developers API Explorer` • `N/A`
`AuthMethodReference`	**Type** string **Properties** Nillable **Description** The authentication method used by a third-party identification provider for an OpenID Connect single sign-on protocol. This field is available in API version 51.0 and later.
`AuthServiceId`	**Type** string **Properties** Nillable **Description** The 18-character ID for an authentication service for a login event. For example, you can use this field to identify the SAML or authentication provider configuration with which the user logged in.
`Browser`	**Type** string **Properties** Nillable **Description** The browser name and version if known. Possible values for the browser name are: • Chrome • Firefox • Safari • Unknown For example, “Chrome 77”.
`CipherSuite`	**Type** picklist

Field	Details
	Properties Nillable, Restricted picklist **Description** The TLS cipher suite used for the login. Values are OpenSSL-style cipher suite names, with hyphen delimiters, for example, `ECDHE-RSA-AES256-GCM-SHA384`. Available in API version 37.0 and later.
`City`	**Type** string **Properties** Nillable **Description** The city where the user's IP address is physically located. This value isn't localized. This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`ClientVersion`	**Type** string **Properties** Nillable **Description** The version number of the login client. If no version number is available, "Unknown" is returned.
`Country`	**Type** string **Properties** Nillable **Description** The country where the user's IP address is physically located. This value isn't localized.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`CountryIso`	**Type** string **Properties** Nillable **Description** The ISO 3166 code for the country where the user's IP address is physically located. For more information, see Country Codes - ISO 3166.

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the transaction security policy, in milliseconds.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The login time of the specified event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** (none) **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object. Also, use this field as the primary key in your queries. Available in API version 42.0 and later.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ForwardedForIp`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The value in the `X-Forwarded-For` header of HTTP requests sent by the client. For logins that use one or more HTTP proxies, the `X-Forwarded-For` header is sometimes used to store the origin IP and all proxy IPs.

Field	Details
	The `ForwardedForIp` field stores whatever value the client sends, which might not be an IP address. The maximum length is 256 characters. Longer values are truncated. The `ForwardedForIp` field isn't populated for logins completed via OAuth flows or single sign-on (SSO). Available in API version 61.0 and later.
`HttpMethod`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The HTTP method of the login request; possible values are `GET`, `POST`, and `Unknown`.
`LoginGeoId`	**Type** string **Properties** Nillable **Description** The Salesforce ID of the LoginGeo object associated with the login user's IP address. For example, 04FB000001TvhiPMAR.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular login instance. This field is also available on the LoginHistory, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. For example, 0YaB000002knVQLKA2.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`LoginLatitude`	**Type** double

Field	Details
	Properties Nillable **Description** The latitude where the user's IP address is physically located.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`LoginLongitude`	**Type** double **Properties** Nillable **Description** The longitude where the user's IP address is physically located.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`LoginSubType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of login flow used. See the `LoginSubType` field of LoginHistory in the Object Reference guide for the list of possible values. Label is **Login Subtype**.
`LoginType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of login used to access the session. See the LoginType field of LoginHistory in the Object Reference guide for the list of possible values.
`LoginUrl`	**Type** string **Properties** Nillable **Description** The URL of the login host from which the request is coming. For example, ***`yourInstance`***`.salesforce.com`.

Field	Details
`NetworkId`	**Type** string **Properties** Nillable **Description** The ID of the Experience Cloud site that the user is logging in to. This field is available if Salesforce Experience Cloud is enabled for your organization.
`Platform`	**Type** string **Properties** Nillable **Description** The operating system name and version that are used during the login event. If no platform name is available, "Unknown" is returned. For example, Mac OSX or iOS/Mac.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction security policy associated with this event. For example, 0NIB000000000KOOAY.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger.

Field	Details
	• `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, that activity is approved from the trusted location for as long as the location is trusted. An example of a particular activity is logging in from a recognized device. • `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response. • `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but will retry. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified.
`PostalCode`	**Type** string **Properties** Nillable **Description** The postal code where the user's IP address is physically located. This value isn't localized.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c.

Field	Details
	This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`RemoteIdentifier`	**Type** string **Properties** Nillable **Description** Reserved for future use.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements.

Field	Details
	Note: This low level isn't available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The IP address of the incoming client request that first reaches Salesforce during a login. For example, `126.7.4.2`. For clients that redirect through one or more HTTP proxies, this field stores the IP address of the first proxy to reach Salesforce. To better identify the origin IP for these cases, check the `ForwardedForIp` field instead.
`Status`	**Type** string **Properties** Nillable **Description** Displays the status of the attempted login. Status is either success or a reason for failure.
`Subdivision`	**Type** string **Properties** Nillable **Description** The name of the subdivision where the user's IP address is physically located. In the U.S., this value is usually the state name (for example, Pennsylvania). This value isn't localized.This field is available in API version 47.0 and later. Note: Due to the nature of geolocation technology, the accuracy of this field can vary.
`TlsProtocol`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The TLS protocol version used for the login. Valid values are: • `TLS 1.0`

Field	Details
	• TLS 1.1 • TLS 1.2 • TLS 1.3 • Unknown
UserId	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, 005000000000123.
Username	**Type** string **Properties** Nillable **Description** The username in the format of user@company.com.
UserType	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license. Each UserType is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values are: • CsnOnly—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • CspLitePortal—CSP Lite Portal license. Users whose access is limited because they're organization customers and access the application through a customer portal or an Experience Cloud site. • CustomerSuccess—Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. • Guest • PowerCustomerSuccess—Power Customer Success license. Users whose access is limited because they're organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • PowerPartner—Power Partner license. Users whose access is limited because they're partners and typically access the application through a partner portal or site. • SelfService

Field	Details
	• `Standard`—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses.

SEE ALSO:

LoginEvent

LogoutEvent

Tracks user UI logouts. A logout event records a successful user logout from your org's UI. LogoutEvent is a big object that stores the event data of LogoutEventStream. This object is available in API version 46.0 and later.

Use LogoutEvent data to implement custom logic during logout. For example, you can revoke all refresh tokens for a user at logout.

Note: LogoutEvent records logouts, not timeouts. Timeouts don't cause a LogoutEventStream object to be published. An exception is when a user is automatically logged out of the org after their session times out because the org has the **Force logout on session timeout** setting enabled. In this case, a logout event is recorded. However, if users close their browser during a session, regardless of whether the **Force logout on session timeout** setting is enabled, a logout event isn't recorded.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field Name	Details
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time when the specified logout event was captured. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort

Field Name	Details
	Description The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. It starts with a login event and ends with either a logout event or the user session expiring.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. You can use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Indicates the session-level security of the session that the user is logging out of for this event. Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level will experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.

Field Name	Details
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client logging out. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** Represents the ID of the user associated with the logout event.
`Username`	**Type** string **Properties** Nillable **Description** Represents the username of the user associated with the logout event.

Standard SOQL Usage

Currently, the only supported SOQL function on LogoutEvent is `WHERE`, and you can only use comparison operators (=, <, >, <=, and >=) on the final expression in a `WHERE` clause. The != operator isn't supported.

Note: Date functions such as `convertTimezone()` aren't supported. For example, `SELECT CALENDAR_YEAR(EventDate), Count(EventIdentifier) FROM LogoutEvent GROUP BY CALENDAR_YEAR(EventDate)` returns an error. You can use date literals in your queries and some date and date/time functions like `TODAY`, `YESTERDAY`, and `LAST_n_DAYS:1`. However, these functions use comparison operators behind the scenes. This means you can only use them in the final expression of a `WHERE` clause.

LogoutEvent allows filtering over two ordered fields: `EventDate` and `EventIdentifier`. There's a catch here; your query won't work unless you use the correct order and combination of these fields. The following list provides some examples of valid and invalid queries:

- **Unfiltered**
  - **Valid**—Contains no `WHERE` clause, so no special rules apply.

```
SELECT EventDate, EventIdentifier, SourceIp, UserId
FROM LogoutEvent
```

- **Filtered on** `EventDate`

  - **Valid**—You can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.

```
SELECT EventDate, EventIdentifier, SourceIp, UserId
FROM LogoutEvent
WHERE EventDate<=2014-11-27T14:54:16.000Z
```

  - **Valid**—You can filter on `EventDate` using date literals.

```
SELECT EventDate, EventIdentifier, SourceIp, UserId
FROM LogoutEvent
WHERE EventDate<=TODAY
```

- **Filtered on `EventDate` and `EventIdentifier`**
  - **Valid**—Successful queries on LogoutEvent filter over both fields.

```
SELECT EventDate, EventIdentifier, SourceIp, UserId
FROM LogoutEvent
WHERE EventDate=2014-11-27T14:54:16.000Z and
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

  - **Invalid**—Queries on LogoutEvent with `EventDate` and standard date literals.

```
SELECT EventDate, EventIdentifier, SourceIp, UserId
FROM LogoutEvent
WHERE EventDate=TODAY and EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

  - **Invalid**—Filtering only on `EventDate` with <= or >= operator and `EventIdentifier` field isn't supported.

```
SELECT EventDate, EventIdentifier, SourceIp, UserId
FROM LogoutEvent
WHERE EventDate<=2014-11-27T14:54:16.000Z and
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

Async SOQL Usage

With Async SOQL, you can filter on any field in LogoutEvent and use any comparison operator in your query.

Example: Get Yesterday's Successful Logouts

```
SELECT EventDate, EventIdentifier, SourceIp, UserId FROM LogoutEvent WHERE
EventDate<Yesterday
```

SEE ALSO:

Big Objects Implementation Guide

LogoutEventStream

Tracks user UI logout. A logout event records a successful user logout from your org's UI. This object is read only, and you can't retrieve it using a SOQL query. This object is available in API version 41.0 and later.

When LogoutEventStream is enabled, Salesforce publishes logout events, and you can add an Apex trigger to subscribe to those events. You can then implement custom logic during logout. For example, you can revoke all refresh tokens for a user at logout.

Note: LogoutEventStream records logouts, not timeouts. Timeouts don't cause a LogoutEventStream object to be published. An exception is when a user is automatically logged out of the org after their session times out because the org has the **Force logout on session timeout** setting enabled. In this case, a logout event is recorded. However, if users close their browser during a session, regardless of whether the **Force logout on session timeout** setting is enabled, a logout event isn't recorded.

Supported Calls

`describeSObjects()`

Special Access Rules

As of Summer '20 and later, only users with the Customize Application user permission can access this object.

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/LogoutEventStream`

Event Delivery Allocation Enforced

No

Fields

Field Name	Details
`EventDate`	**Type** datetime **Properties** Nillable **Description** Represents when the event started. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string

Field Name	Details
	Properties Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. It starts with a login event and ends with either a logout event or the user session expiring.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the EventIdentifier of the related event.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SessionKey`	**Type** string

Field Name	Details
	Properties Nillable **Description** The user's unique session ID. You can use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Indicates the session-level security of the session that the user is logging out of for this event. Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client logging out. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** Represents the ID of the user associated with the logout event.

Field Name	Details
`Username`	**Type** string **Properties** Nillable **Description** Represents the username of the user associated with the logout event.

Usage

In this example, the subscriber inserts a custom logout event record during logout.

```
trigger LogoutEventTrigger on LogoutEventStream (after insert) {
  LogoutEventStream event = Trigger.new[0];
  LogoutEvent__c record = new LogoutEvent__c();
  record.EventIdentifier__c = event.EventIdentifier;
  record.UserId__c = event.UserId;
  record.Username__c = event.Username;
  record.EventDate__c = event.EventDate;
  record.RelatedEventIdentifier__c = event.RelatedEventIdentifier;
  record.SessionKey__c = event.SessionKey;
  record.LoginKey__c = event.LoginKey;
  insert(record);
}
```

MobileEmailEvent

Tracks your users' email activity in a Salesforce mobile app with Enhanced Mobile Security. This object is available in API version 47.0 and later.

Use the Mobile Security SDK to publish these events. Learn more with the Mobile Application Security help documentation.

Supported Calls

`create()`,`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✔
Flows	✔
Processes	✔
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/MobileEmailEvent`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security and Salesforce Event Monitoring add-on subscriptions and the Enforce Enhanced Mobile App Security user permission.

As of Summer '20 and later, only authenticated internal and external users can access this object.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the app.
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string **Properties** Create **Description** Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.

Field	Details
`EmailAddress`	**Type** string **Properties** Create **Description** Email address of the email recipient.
`EventDate`	**Type** dateTime **Properties** Create, Nillable **Description** The date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string **Properties** Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object, if any. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`OsName`	**Type** string **Properties** Create

Field	Details
	Description Name of the operating system.
`OsVersion`	**Type** string **Properties** Create **Description** Version number of the operating system.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`UserId`	**Type** reference **Properties** Create, Namepointing **Description** ID of the user who triggered the event.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

MobileEmailEventStore

Tracks your users' email activity in a Salesforce mobile app with Enhanced Mobile App Security. MobileEmailEventStore is a standard object that stores the event data of MobileEmailEvent. This object is available in API version 48.0 and later.

Use the Mobile Security SDK to have this event generated. See Mobile Application Security in Salesforce Help.

Supported Calls

`create()`, `describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security add-on subscription and the Enforce Enhanced Mobile App Security user permission. Accessing this object also requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the app.
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string **Properties** Create **Description** Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.
`EmailAddress`	**Type** string

Field	Details
	Properties Create **Description** Email address of the email recipient.
`EventDate`	**Type** dateTime **Properties** Create, Filter, Sort **Description** The date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string **Properties** Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`OsName`	**Type** string **Properties** Create **Description** Name of the operating system.
`OsVersion`	**Type** string **Properties** Create **Description** Version number of the operating system.

Field	Details
`UserId`	**Type** reference **Properties** Create **Description** Id of the user who triggered the event. This field is a polymorphic relationship field. **Relationship Name** User **Refers To** User
`Username`	**Type** string **Properties** Create, Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

MobileEnforcedPolicyEvent

Tracks enforcement of Enhanced Mobile Security policy events on a Salesforce mobile app with Enhanced Mobile Security. Events are created on first launch of the mobile app and user rechecks, and are batched and published when the app is in the background. This object is available in API version 47.0 and later.

Use the Mobile Security SDK to publish these events. Learn more with the Mobile Application Security help documentation.

Supported Calls

`create()`, `describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓

Subscriber	Supported?
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/MobileEnforcedPolicyEvent`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security and Salesforce Event Monitoring add-on subscriptions and the Enforce Enhanced Mobile App Security user permission.

As of Summer '20 and later, only authenticated internal and external users can access this object.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the application.
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string **Properties** Create

Field	Details
	Description Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.
`EnforcedAction`	**Type** json **Properties** Create **Description** Action that the policy enforced. Possible values are: • `Warn` • `Error` • `Critical Error`
`EventDate`	**Type** dateTime **Properties** Create, Nillable **Description** Date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string **Properties** Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Nillable

Field	Details
	Description The unique ID of the event, which is shared with the corresponding storage object, if any. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`OsName`	**Type** string **Properties** Create **Description** Operating system name iOS or Android.
`OsVersion`	**Type** string **Properties** Create **Description** Operating system version number.
`PolicyResults`	**Type** json **Properties** Create **Description** Collection of the results of all policies enforced at the time of the event.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

Field	Details
`UserId`	**Type** reference **Properties** Create, Namepointing **Description** ID of the user for whom policies were enforced.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

MobileEnfPolicyEventStore

Tracks enforcement of Enhanced Mobile App Security policy events on a Salesforce mobile app with Enhanced Mobile App Security. Events are created on first launch of the mobile app and user rechecks, and are batched and published when the app is in the background. MobileEnfPolicyEventStore is a standard object that stores the event data of MobileEnforcedPolicyEvent. This object is available in API version 48.0 and later.

Use the Mobile Security SDK to have this event generated. See Mobile Application Security in Salesforce Help.

Supported Calls

`create()`, `describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security add-on subscription and the Enforce Enhanced Mobile App Security user permission. Accessing this object also requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the app.

Field	Details
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string **Properties** Create **Description** Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.
`EnforcedAction`	**Type** json **Properties** Create **Description** Action that the policy enforced. Possible values are: • `Warn` • `Error` • `Critical Error`
`EventDate`	**Type** dateTime **Properties** Create, Filter, Sort **Description** Date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string

Field	Details
	Properties Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`OsName`	**Type** string **Properties** Create **Description** Operating system name.
`OsVersion`	**Type** string **Properties** Create **Description** Operating system version number.
`PolicyResults`	**Type** json **Properties** Create **Description** Collection of the results of all policies enforced at the time of the event.
`UserId`	**Type** reference **Properties** Create **Description** Id of the user for whom policies were enforced.

Field	Details
	This field is a polymorphic relationship field. **Relationship Name** User **Refers To** User
`Username`	**Type** string **Properties** Create, Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

MobileScreenshotEvent

Tracks your users' screenshots in a Salesforce mobile app with Enhanced Mobile Security. This object is available in API version 47.0 and later.

Use the Mobile Security SDK to publish these events. Learn more with the Mobile Application Security help documentation.

Supported Calls

`create()`, `describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/MobileScreenshotEvent`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security and Salesforce Event Monitoring add-on subscriptions and the Enforce Enhanced Mobile App Security user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the application.
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string **Properties** Create **Description** Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.

Field	Details
`EventDate`	**Type** dateTime **Properties** Create, Nillable **Description** Date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string **Properties** Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object, if any. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`OsName`	**Type** string **Properties** Create **Description** Name of the operating system.
`OsVersion`	**Type** string **Properties** Create

Field	Details
	Description Version number of the operating system.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`ScreenDescription`	**Type** string **Properties** Create **Description** Description of what was viewable on the screen when the user took a screenshot, such as Chatter Feed or Record Detail View.
`UserId`	**Type** reference **Properties** Create, Namepointing **Description** ID of the user who triggered the event.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

MobileScreenshotEventStore

Tracks your users' screenshots in a Salesforce mobile app with Enhanced Mobile App Security. MobileScreenshotEventStore is a standard object that stores the event data of MobileScreenshotEvent. This object is available in API version 48.0 and later.

Use the Mobile Security SDK to have this event generated. See Mobile Application Security in Salesforce Help.

Supported Calls

`create()`, `describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security add-on subscription and the Enforce Enhanced Mobile App Security user permission. Accessing this object also requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the app.
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string **Properties** Create **Description** Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.
`EventDate`	**Type** dateTime

Field	Details
	Properties Create, Filter, Sort **Description** The date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string **Properties** Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`OsName`	**Type** string **Properties** Create **Description** Name of the operating system.
`OsVersion`	**Type** string **Properties** Create **Description** Version number of the operating system.
`ScreenDescription`	**Type** string **Properties** Create

Field	Details
	Description Description of what was viewable on the screen when the user took a screenshot, such as Chatter Feed or Record Detail View.
`UserId`	**Type** reference **Properties** Create **Description** Id of the user who triggered the event. This field is a polymorphic relationship field. **Relationship Name** User **Refers To** User
`Username`	**Type** string **Properties** Create, Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

MobileTelephonyEvent

Tracks your users' phone calls and text messages in a Salesforce mobile app with Enhanced Mobile Security. This object is available in API version 47.0 and later.

Use the Mobile Security SDK to publish these events. Learn more with the Mobile Application Security help documentation.

Supported Calls

`create()`, `describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	✓
Flows	✓
Processes	✓
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/MobileTelephonyEvent`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security and Salesforce Event Monitoring add-on subscriptions and the Enforce Enhanced Mobile App Security user permission.

As of Summer '20 and later, only authenticated internal and external users can access this object.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the application.
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string

Field	Details
	Properties Create **Description** Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.
`EventDate`	**Type** dateTime **Properties** Create, Nillable **Description** Date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string **Properties** Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object, if any. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.

Field	Details
`Operation`	**Type** picklist **Properties** Create, Restricted picklist **Description** Type of operation that triggered the event. Possible values are: • `PhoneCall` • `SMS`
`OsName`	**Type** string **Properties** Create **Description** Name of the operating system.
`OsVersion`	**Type** string **Properties** Create **Description** Version number of the operating system.
`PhoneNumber`	**Type** string **Properties** Create **Description** Phone number for the recipient of the phone call or text message.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

Field	Details
`UserId`	**Type** reference **Properties** Create, Namepointing **Description** ID of the user who triggered the event.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

MobileTelephonyEventStore

Tracks your users' phone calls and text messages in a Salesforce mobile app with Enhanced Mobile App Security. MobileTelephonyEventStore is a standard object that stores the event data of MobileTelephonyEvent. This object is available in API version 48.0 and later.

Use the Mobile Security SDK to have this event generated. See Mobile Application Security in Salesforce Help.

Supported Calls

`create()`, `describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires the Enhanced Mobile App Security add-on subscription and the Enforce Enhanced Mobile App Security user permission. Accessing this object also requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`AppPackageIdentifier`	**Type** string **Properties** Create **Description** Generic package identifier for the app.

Field	Details
`AppVersion`	**Type** string **Properties** Create **Description** Version number of the application.
`DeviceIdentifier`	**Type** string **Properties** Create **Description** Unique identifier for the device. Generated by Apple® or Google™.
`DeviceModel`	**Type** string **Properties** Create **Description** Model name of the device.
`EventDate`	**Type** dateTime **Properties** Create, Filter, Sort **Description** The date of the mobile event. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventDescription`	**Type** string **Properties** Create, Nillable **Description** Description of the mobile event.
`EventIdentifier`	**Type** string **Properties** Create, Filter, Sort

Field	Details
	Description The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`Operation`	**Type** picklist **Properties** Create, Restricted picklist **Description** Type of operation that triggered the event. Possible values are: • `PhoneCall` • `SMS`
`OsName`	**Type** string **Properties** Create **Description** Name of the operating system.
`OsVersion`	**Type** string **Properties** Create **Description** Version number of the operating system.
`PhoneNumber`	**Type** string **Properties** Create **Description** Phone number for the recipient of the phone call or text message.
`UserId`	**Type** reference **Properties** Create

Field	Details
	Description Id of the user who triggered the event. This field is a polymorphic relationship field. **Relationship Name** User **Refers To** User
`Username`	**Type** string **Properties** Create, Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`WebkitVersion`	**Type** string **Properties** Create, Nillable **Description** Version of WebKit™ used to render web components.

PermissionSetEvent

Tracks changes to permission sets and permission set groups. This event initiates when a permission is added to, or removed from a permission set. This event also initiates when a permission set containing a critical permission is assigned or unassigned. This object is available in API version 52.0 and later.

Supported Calls

`describeSObjects()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	

Subscriber	Supported?
Processes	
Pub/Sub API	✔
Streaming API (CometD)	✔

Event Delivery Allocation Enforced

No

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventSource`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The source of the event. Possible values are:

Field	Details
	• `API`—The user made changes to a permission set or permission set group from an API call. • `Classic`—The user made changes to a permission set or permission set group from a page in the Salesforce Classic UI. • `Lightning`—The user made changes to a permission set or permission set group from a page in the Lightning Experience UI.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message.
`HasExternalUsers`	**Type** boolean **Properties** Nillable **Description** When true, external users are impacted by the operation that triggered a permission change. The default value is false.
`ImpactedUserIds`	**Type** json **Properties** Nillable **Description** A comma-separated list of IDs of the users affected by the event. A maximum of 1,000 user IDs are included. For example, if a permission set assigned to two users is updated, the users' IDs are recorded in this field.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular series of permission set events. This field is also available in the LoginEvent, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. For example, `0YaB000002knVQLKA2`. This is a relationship field.

Field	Details
	Relationship Name LoginHistory **Relationship Type** Lookup **Refers To** LoginHistory
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`Operation`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of operation that triggers a permission change. Possible values are: • `AssignedToUsers`—A permission set or permission set group is assigned to one or more users. • `CriticalPerms`—This deprecated value indicates the critical permissions are enabled. • `PermsDisabled`—Permissions are disabled. • `PermsEnabled`—Permissions are enabled. • `UnassignedFromUsers`—A permission set or permission set group is unassigned from one or more users.
`ParentIdList`	**Type** json **Properties** Nillable **Description** The IDs of the affected permission sets or permission set groups.
`ParentNameList`	**Type** json

Field	Details
	Properties Nillable **Description** The names of the affected permission sets or permission set groups.
`PermissionExpirationList`	**Type** json **Properties** Nillable **Description** A comma separated list of timestamps from the PermissionSetAssignment.ExpirationDate field that specifies when added permissions will be revoked. This value is null when no expiration timestamp is specified or permissions are removed for the impacted users.
`PermissionList`	**Type** json **Properties** Nillable **Description** The list of permissions that are enabled or disabled in the event.
`PermissionType`	**Type** string **Properties** Nillable **Description** The type of permission that is updated in the event. Possible values are: • `ObjectPermission` • `UserPermission`
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction security policy associated with this event. For example, `0NIB000000000KOOAY`. This is a relationship field. **Relationship Name** Policy

Field	Details
	Relationship Type Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `EndSession`—The user's session is terminated. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, such as logging in from a recognized device, that activity is approved from the trusted location for as long as the location is trusted. • `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response.

Field	Details
	• `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but will retry. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, `bd76f3e7-9ee5-4400-9e7f-54de57ecd79c`. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.

Field	Details
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level isn't available or used in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`UserCount`	**Type** string **Properties** Nillable **Description** The number of users affected by the event. This field has a maximum value of 1,000. If the user appears more than 1,000 times, the value remains at 1,000.
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005000000000123`. This is a polymorphic relationship field.

Field	Details
	Relationship Name User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string **Properties** Nillable **Description** The username in the format of `user@company.com` at the time the event was created.

PermissionSetEventStore

Tracks changes to permission sets and permission set groups. This event initiates when a permission is added to, or removed from a permission set. This event also initiates when a permission set containing a critical permission is assigned or unassigned. PermissionSetEventStore is a big object that stores the event data of PermissionSetEvent. This object is available in API version 52.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds.
`EventDate`	**Type** dateTime

Field	Details
	Properties Filter, Sort **Description** The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds is the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventSource`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The source of the event. Possible values are: • `API`—The user changed a permission set or permission set group from an API call. • `Classic`—The user made changes to a permission set or permission set group from a page in the Salesforce Classic UI. • `Lightning`—The user made changes to a permission set or permission set group from a page in the Lightning Experience UI.
`HasExternalUsers`	**Type** boolean **Properties** Nillable **Description** When true, external users are impacted by the operation that triggered a permission change. The default value is false.
`ImpactedUserIds`	**Type** json **Properties** Nillable **Description** A comma-separated list of IDs of the users affected by the event. A maximum of 1,000 user IDs are included.

Field	Details
	For example, if a permission set assigned to two users is updated, the users' IDs are recorded in this field.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so you can correlate user activity with a particular series of permission set events. This field is also available in the LoginEvent, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. For example, `0YaB000002knVQLKA2`. This is a relationship field. **Relationship Name** LoginHistory **Relationship Type** Lookup **Refers To** LoginHistory
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `lUqjLPQTWRdvRG4`.
`Operation`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The type of operation that triggers a permission change. Possible values are: • `AssignedToUsers`—A permission set or permission set group is assigned to one or more users. • `CriticalPerms`—This deprecated value indicates the critical permissions that are enabled. • `PermsDisabled`—Permissions are disabled. • `PermsEnabled`—Permissions are enabled.

Field	Details
	• `UnassignedFromUsers`—A permission set or permission set group is unassigned from one or more users.
`ParentIdList`	**Type** json **Properties** Nillable **Description** The IDs of the affected permission sets or permission set groups.
`ParentNameList`	**Type** json **Properties** Nillable **Description** The names of the affected permission sets or permission set groups.
`PermissionExpirationList`	**Type** json **Properties** Nillable **Description** A comma separated list of timestamps from the PermissionSetAssignment.ExpirationDate field that specifieswhen added permissions will be revoked. This value is null when no expiration timestamp is specified or permissions are removed for the impacted users.
`PermissionList`	**Type** json **Properties** Nillable **Description** The list of permissions that are enabled or disabled in the event.
`PermissionType`	**Type** string **Properties** Nillable **Description** The type of permission that is updated in the event. Possible values are: • `ObjectPermission` • `UserPermission`

Field	Details
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction security policy associated with this event. For example, `0NIB000000000KOOAY`. This is a relationship field. **Relationship Name** Policy **Relationship Type** Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `EndSession`—The user's session is terminated. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, such as logging in from a recognized device, that activity is approved from the trusted location for as long as the location is trusted.

Field	Details
	• `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response. • `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but will retry. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, `bd76f3e7-9ee5-4400-9e7f-54de57ecd79c`. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, `vMASKIU6AxEr+Op5`.

Field	Details
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level isn't available or used in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`UserCount`	**Type** string **Properties** Nillable **Description** The number of users affected by the event. This field has a maximum value of 1,000. If the user appears more than 1,000 times, the value remains at 1,000.
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005000000000123`. This is a polymorphic relationship field.

Field	Details
	Relationship Name User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string **Properties** Nillable **Description** The username in the format of `user@company.com` at the time the event was created.

ReportAnomalyEvent

Tracks anomalies in how users run or export reports, including unsaved reports. This object is available in API version 49.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	✓
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/ReportAnomalyEvent`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginKey`	**Type** string **Properties** Nillable

Field	Details
	Description The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event. A relationship field. **Relationship Name** Policy **Relationship Type** Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Error` - The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`ReplayId`	**Type** string

Field	Details
	Properties Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`Report`	**Type** string **Properties** Nillable **Description** The report ID for the report for which this anomaly event was detected. For example, `00OD00000011eVCMAY`. If this anomaly resulted from a user executing an unsaved report, the value of this field is null.
`Score`	**Type** double **Properties** Nillable **Description** A number from 0 through 100 that represents the anomaly score for the report execution or export tracked by this event. The anomaly score shows how the user's current report activity is different from their typical activity. A low score indicates that the user's current report activity is similar to their usual activity. A high score indicates that it's different.
`SecurityEventData`	**Type** textarea **Properties** Nillable **Description** The set of features about the report activity that triggered this anomaly event. Let's say, for example, that a user typically downloads 10 accounts but then they deviate from that pattern and download 1,000 accounts. This event is triggered and the contributing features are captured in this field. Potential features include row count, column count, average row size, the day of week, and the browser's user agent used for the report activity. The data captured in this field also shows how much a particular feature contributed to this anomaly event being triggered, represented as a percentage. The data is in JSON format.

Field	Details
	Example This example shows that the average row count contributed more than 95% to the anomaly being triggered. Other anomalous features, such as the autonomous system, day of the week the report was run, the browser used, and the number of columns, contributed much less. `[{ "featureName": "rowCount", "featureValue": "1937568", "featureContribution": "95.00 %" }, { "featureName": "autonomousSystem", "featureValue": "Bigleaf Networks, Inc.", "featureContribution": "1.62 %" }, { "featureName": "dayOfWeek", "featureValue": "Sunday", "featureContribution": "1.42 %" }, { "featureName": "userAgent", "featureValue": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36}", "featureContribution": "1.21 %" }, { "featureName": "periodOfDay", "featureValue": "Evening", "featureContribution": ".09 %" }, { "featureName": "averageRowSize", "featureValue": "744", "featureContribution": "0.08 %" }, { "featureName": "screenResolution", "featureValue": "900x1440", "featureContribution": "0.07 %" }]`
`SessionKey`	**Type** string **Properties** Nillable

Field	Details
	Description The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`. Session information contained in the fields SessionKey, LoginKey, SessionLevel, and SourceIp isn't captured in any report resulting from an asynchronous operation.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the report anomaly that caused this event to be created. **Example** • `Report was exported from an infrequent network (BigLeaf Networks Inc.)` • `Report was generated with an unusually high number of rows (111141)`
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`. A polymorphic relationship field. **Relationship Name** User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string

Field	Details
	Properties Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

ReportAnomalyEventStore

Tracks anomalies in how users run or export reports, including unsaved reports. ReportAnomalyEventStore is an object that stores the event data of ReportAnomalyEvent. This object is available in API version 49.0 and later.

Supported Calls

`describeLayout()describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`EvaluationTime`	**Type** double **Properties** Filter, Nillable, Sort **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** Required. The time when the anomaly was reported. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string

Field	Details
	Properties Filter, Group, Sort **Description** Required. The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`LastReferencedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The timestamp for when the current user last viewed a record related to this record.
`LastViewedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The timestamp for when the current user last viewed this record. If this value is null, it's possible that this record was referenced (`LastReferencedDate`) and not viewed.
`LoginKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`PolicyId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist

Field	Details
	Properties Filter, Group, Nillable, Restricted picklist, Sort **Description** The result of the transaction policy. Possible values are: • `Error` - The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`Report`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The report ID for the report for which this anomaly event was detected. For example, `00OD00000011eVCMAY`. If this anomaly resulted from a user executing an unsaved report, the value of this field is null.
`ReportAnomalyEventNumber`	**Type** string **Properties** Autonumber, Defaulted on create, Filter, idLookup, Sort **Description** The unique number automatically assigned to the event when it's created. You can't change the format or value for this field.
`Score`	**Type** double **Properties** Filter, Nillable, Sort **Description** A number from 0 through 100 that represents the anomaly score for the report execution or export tracked by this event. The anomaly score shows how the user's current report

Field	Details
	activity is different from their typical activity. A low score indicates that the user's current report activity is similar to their usual activity, a high score indicates that it's different.

SecurityEventData	**Type**
	textarea
	Properties
	Nillable
	Description
	The set of features about the report activity that triggered this anomaly event.
	Let's say, for example, that a user typically downloads 10 accounts but then they deviate from that pattern and download 1,000 accounts. This event is triggered and the contributing features are captured in this field. Potential features include row count, column count, average row size, the day of week, and the browser's user agent used for the report activity. The data captured in this field also shows how much a particular feature contributed to this anomaly event being triggered, represented as a percentage. The data is in JSON format.
	Example
	This example shows that the average row count contributed more than 95% to the anomaly being triggered. Other anomalous attributes, such as the autonomous system, day of the week the report was run, the browser used, and the number of columns, contributed much less.
	`[`
	`{`
	`"featureName": "rowCount",`
	`"featureValue": "1937568",`
	`"featureContribution": "95.00 %"`
	`},`
	`{`
	`"featureName": "autonomousSystem",`
	`"featureValue": "Bigleaf Networks, Inc.",`
	`"featureContribution": "1.62 %"`
	`},`
	`{`
	`"featureName": "dayOfWeek",`
	`"featureValue": "Sunday",`
	`"featureContribution": "1.42 %"`
	`},`
	`{`
	`"featureName": "userAgent",`
	`"featureValue": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36}",`
	`"featureContribution": "1.21 %"`
	`},`
	`{`
	`"featureName": "periodOfDay",`
	`"featureValue": "Evening",`
	`"featureContribution": ".09 %"`
	`},

Field	Details

	<pre>{
	"featureName": "averageRowSize",
	"featureValue": "744",
	"featureContribution": "0.08 %"
	},
	{
	"featureName": "screenResolution",
	"featureValue": "900x1440",
	"featureContribution": "0.07 %"
	}
	]</pre>

SessionKey	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
SourceIp	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The source IP address of the client that logged in. For example, `126.7.4.2`. Session information contained in the fields SessionKey, LoginKey, SessionLevel, and SourceIp isn't captured in any report resulting from an asynchronous operation.
Summary	**Type** textarea **Properties** Nillable **Description** A text summary of the report anomaly that caused this event to be created. **Example** • `Report was exported from an infrequent network (BigLeaf Networks Inc.)` • `Report was generated with an unusually high number of rows (111141)`
UserId	**Type** reference

Field	Details
	Properties Filter, Group, Nillable, Sort **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The origin username in the format of `user@company.com` at the time the event was created.

Associated Object

This object has the following associated object. It's available in the same API version as this object.

ReportAnomalyEventStoreFeed
Feed tracking is available for the object.

ReportEvent

Tracks when reports are run in your org. You can use ReportEvent in a transaction security policy. ReportEvent is a big object that stores the event data of ReportEventStream. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`ColumnHeaders`	**Type** string **Properties** Nillable **Description** Comma-separated values of column headers of the report. Values listed are object names, field names, and field values except where aliases are used. For example, [Opportunity.Name,

Field	Details
	Opportunity.Type, Opportunity.Owner.UserRole.RollupDescription, Opportunity.Account.Name, Opportunity.Account.NumberOfEmployees, AGE].
`DashboardId`	**Type** reference **Properties** Nillable **Description** The ID of the dashboard that the report was part of. For example, 01ZB0000000PmoQ. This is a relationship field. **Relationship Name** Dashboard **Relationship Type** Lookup **Refers To** Dashboard
`DashboardName`	**Type** string **Properties** Nillable **Description** The title of the dashboard that the report was part of.
`Description`	**Type** string **Properties** Nillable **Description** The description of the report.
`DisplayedFieldEntities`	**Type** string **Properties** Nillable **Description** The API values of the fields that are displayed on the report, including the names of the entities of the grouped column fields. For example, `[ACCOUNTS, OWNERS]`.
`EvaluationTime`	**Type** double

Field	Details
	Properties Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** The time when the specified report event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Sort **Description** The unique ID of the event. For example, 0a4779b0-0da1-4619-a373-0a36991dff90.
`EventSource`	**Type** picklist **Properties** Nillable, Restricted Picklist **Description** The source of the event. Possible values are: • `API`—The user generated the report from an API call. • `Classic`—The user generated the report from the Salesforce Classic UI. • `Lightning`—The user generated the report from Lightning Experience.
`ExecutionIdentifier`	**Type** string **Properties** Nillable **Description** When report data is divided into multiple report events, use this unique identifier to correlate the multiple data chunks. For example, if each chunk has the same `ExecutionIdentifier` of a50a4025-84f2-425d-8af9-2c780869f3b5,you can link them together to get all the data for the report execution. The `Sequence` field contains the incremental sequence numbers that indicate the order of the multiple events.

Field	Details
	For more information, see `Sequence`.
`ExportFileFormat`	**Type** string **Properties** Nillable **Description** If the user exported the report, this value indicates the format of the exported report. Possible values are: • CSV • Excel
`Format`	**Type** picklist **Properties** Defaulted on create, Nillable, Restricted picklist **Description** The format of the report. Possible values are: • `Matrix` • `MultiBlock` • `Summary` • `Tabular`
`GroupedColumnHeaders`	**Type** string **Properties** Nillable **Description** Comma-separated values of grouped column fields in summary, matrix, and joined reports. For example, `[USERNAME, ACCOUNT.NAME, TYPE, DUE_DATE, LAST_UPDATE, ADDRESS1_STATE]`.
`IsScheduled`	**Type** boolean **Properties** Defaulted on create **Description** If TRUE, the report was scheduled. If FALSE, the report wasn't scheduled.
`LoginHistoryId`	**Type** reference

Field	Details
	Properties Nillable **Description** Tracks a user session so that you can correlate user activity with a particular series of report events. This field is also available on the LoginEvent, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. This value is null if the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report. For example, 0YaB000002knVQLKA2. This is a relationship field. **Relationship Name** LoginHistory **Relationship Type** Lookup **Refers To** LoginHistory
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. This value is null if the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report. For example, lUqjLPQTWRdvRG4.
`Name`	**Type** string **Properties** Nillable **Description** The display name of the report. The value is null for report previews.
`NumberOfColumns`	**Type** int **Properties** Nillable **Description** The number of columns in the report.
`Operation`	**Type** picklist

Field	Details
	Properties Nillable, Restricted Picklist **Description** The context in which the report executed, such as from a UI (Classic, Lightning, Mobile), through an API (synchronous, asynchronous, Apex), or through a dashboard. Session information contained in the fields `SessionKey`, `LoginKey`, `SessionLevel`, and `SourceIp` isn't captured in any report resulting from an asynchronous operation. Possible values are: • `ChartRenderedInEmbeddedAnalyticsApp`—Report executed from a rendered chart in an embedded Analytics app. • `ChartRenderedOnHomePage`—Report executed from a rendered chart on the home page. • `ChartRenderedOnVisualforcePage`—Report executed from a rendered chart on a VisualForce Page. • `DashboardComponentPreviewed`—Report executed from a Lightning dashboard component preview. • `DashboardComponentUpdated`—Report executed when a user refreshed a dashboard component. • `ProbeQuery`—Report executed from a probe query. • `ReportAddedToCampaign`—Report was added from an Add to Campaign action. • `ReportExported`—Report executed from a printable view or report export that wasn't asynchronous nor an API export. • `ReportExportedAsynchronously`—Report was exported asynchronously. • `ReportExportedUsingExcelConnector`—Report was exported using the Excel connector. • `ReportOpenedFromMobileDashboard`—Report executed when a user clicked a dashboard component on a mobile device and drilled down to a report. • `ReportPreviewed`—Report executed when a user got preview results while using the report builder. • `ReportResultsAddedToEinsteinDiscovery`—Report executed synchronously from Einstein Discovery. • `ReportResultsAddedToWaveTrending`—Report executed when a user trended a report in CRM Analytics. • `ReportRunAndNotificationSent`—Report executed through the notifications API. • `ReportRunFromClassic`—Report executed from the Run Report option of Salesforce Classic. • `ReportRunFromLightning`—Report executed from the Run option in Lightning Experience from a non-mobile browser. • `ReportRunFromMobile`—Report executed from the Run Report option of the mobile Salesforce app.

Field	Details
	• `ReportRunFromReportingSnapshot`—Report executed through Snapshot Analytics. • `ReportRunFromRestApi`—Report executed from REST API. • `ReportRunUsingApexAsynchronousApi`—Report executed from the asynchronous Apex API. • `ReportRunUsingApexSynchronousApi`—Report executed from the synchronous Apex API. • `ReportRunUsingAsynchronousApi`—Report executed from an asynchronous API. • `ReportRunUsingSynchronousApi`—Report executed from a synchronous API. • `ReportScheduled`—Report was scheduled. • `Test`—Report execution resulted from a test. • `Unknown`—Report execution origin is unknown.
`OwnerId`	**Type** reference **Properties** Nillable **Description** The ID of the folder, organization, or user who owns the report. If the report wasn't saved, this value is the same as `UserId`. For example, 005B0000001vURv. This is a polymorphic relationship field. **Relationship Name** Owner **Relationship Type** Lookup **Refers To** Folder, Organization, User
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event. This is a relationship field. **Relationship Name** Policy

Field	Details
	Relationship Type Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user wasn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, that activity is approved from the trusted location for as long as the location is trusted. Logging in from a recognized device is an example. • `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response.

Field	Details
	• `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but retries. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified. This field isn't populated until all transaction security policies are processed for the real-time event.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The entities in the SOQL query. For example, Opportunity, Lead, Account, or Case. Can also include custom objects. For relationship queries, the value of this field contains all entities involved in the query. If the query returns 0 records, then the value of this field is null. **Examples** • For `SELECT Contact.FirstName, Contact.Account.Name from Contact`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Account.Name, (SELECT Contact.FirstName, Contact.LastName FROM Account.Contacts) FROM Account`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Id, Name, Account.Name FROM Contact WHERE Account.Industry = 'media'`, the value of `QueriedEntities` is `Account, Contact`.
`Records`	**Type** json **Properties** Nillable **Description** A JSON string that represents the report's data. For example, `{"totalSize":1,"rows":[{"datacells":["005B0000001vURv","001B000000fewai"]}]}`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable

Field	Details
	Description Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`ReportId`	**Type** reference **Properties** Nillable **Description** The ID of the report associated with this event. For example, 00OB00000032FHdMAM. This is a relationship field. **Relationship Name** Report **Relationship Type** Lookup **Refers To** Report
`RowsProcessed`	**Type** double **Properties** Nillable **Description** The total number of rows returned in the report. When report data is divided into multiple report events, this value is the same for all data chunks. For more information, see `ExecutionIdentifier`.
`Scope`	**Type** string **Properties** Nillable **Description** Defines the scope of the data on which the user ran the report. For example, users can run the report against all opportunities, opportunities they own, or opportunities their team owns. Possible values are: • `user`—User owns the objects the report was run against.

Field	Details
	• `team`—Team owns the objects the report was run against. • `organization`—Report was run against all applicable objects.
`Sequence`	**Type** int **Properties** Nillable **Description** Incremental sequence number that indicates the order of multiple events that result from a given report execution. When a report execution returns many records, Salesforce splits this data into chunks based on the size of the records, and then creates separate multiple ReportEventStreams. The field values in each of these correlated ReportEventStreams are the same, except for `Records` and `Sequence`. `Records` contains the different data chunks. `Sequence` identifies each chunk in order. Every report execution has a unique `ExecutionIdentifier` value to differentiate it from other report executions. To view all the data chunks from a single report execution, use the `Sequence` and `ExecutionIdentifier` fields in combination. When a report executes, we provide the first 1,000 events with data in the `Records` field. Use the `ReportId` field to view the full report. For more information, see `ExecutionIdentifier`.
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. This value is null if the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report. For example, vMASKIU6AxEr+Op5.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level.

Field	Details
	• `LOW`—The user's security level for the current session meets the lowest requirements. This low level isn't available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels. This value is null if the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, 126.7.4.2.
`UserId`	**Type** reference **Properties** Filter, Sort **Description** The origin user's unique ID. For example, `005B0000001vURv`. This is a polymorphic relationship field. **Relationship Name** User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

Standard SOQL Usage

Currently, the only supported SOQL function on ReportEvent is `WHERE`, and you can only use comparison operators (=, <, >, <=, and >=) on the final expression in a `WHERE` clause. The `!=` operator isn't supported.

Date functions such as `convertTimezone()` aren't supported. For example, `SELECT CALENDAR_YEAR(EventDate), Count(EventIdentifier) FROM ReportEvent GROUP BY CALENDAR_YEAR(EventDate)` returns an error. You can use date literals in your queries and some date and date/time functions like `TODAY`, `YESTERDAY`, and `LAST_n_DAYS:1`. However, these functions use comparison operators behind the scenes, so you can only use them in the final expression of a `WHERE` clause.

ReportEvent allows filtering over three ordered fields: `UserId` (Beta), `EventDate`, and `EventIdentifier`. There's a catch here: Your query doesn't work unless you use the correct order and combination of these fields.

Valid filters for ReportEvent queries are:

- `UserId` alone
- `EventDate` alone
- `UserId` with `EventDate`
- `EventDate` with `EventIdentifier`
- `EventDate` can have a range filter when the order of the filter is `UserId`, `EventDate`.
- `EventIdentifier` can have a range query when the order is `EventDate`, `EventIdentifier`.

Note: As a beta feature, the UserId filter in ReportEvent is a preview and isn't part of the "Services" under your Main Services Agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is and isn't supported, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature.

This list provides some examples of valid and invalid queries.

- **Unfiltered query**
  - **Valid**—Contains no `WHERE` clause, so no special rules apply.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
```

- **Filter on** `UserId` (Beta)
  - **Valid**—You can filter solely on `UserId` (Beta). You can include a range query when you filter on `UserId` (Beta) alone.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
WHERE UserId='005B0000001vURv'<=TODAY
```

  - **Valid**—Filter on `UserId` (Beta) and `EventDate`. EventDate can also have a range filter if the order of the filter is UserId (Beta), EventDate.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
WHERE UserId='005B0000001vURv' AND EventDate<=TODAY
```

  - **Valid**— Filter on `UserId` (Beta), and sort the results.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
WHERE UserId = '005B0000001vURv'
ORDER BY EventDate DESC
```

  - **Invalid**—Filtering on `UserId` (Beta) and `EventIdentifier` field isn't supported.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
WHERE UserId='005B0000001vURv' AND
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

- **Filter on** `EventDate`
  - **Valid**—You can filter on `EventDate` using date literals. Or you can include a range query when you filter on `EventDate` alone.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
WHERE EventDate<=TODAY
```

  - **Invalid**—Filtering on `EventDate` with standard date literals isn't supported.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
WHERE EventDate=TODAY AND EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

  - **Invalid**—Filtering on `EventDate` with <= or >= operator and `EventIdentifier` field isn't supported.

```
SELECT DashboardId,Description,DisplayedFieldEntities,EventDate,Format,UserId
FROM ReportEvent
WHERE EventDate<=2014-11-27T14:54:16.000Z AND
EventIdentifier='f0b28782-1ec2-424c-8d37-8f783e0a3754'
```

Async SOQL Usage

With Async SOQL, you can filter on any field in ReportEvent and use any comparison operator in your query.

Example: Find all reports that users ran against Patent__c

```
SELECT EventDate, EventIdentifier, PolicyOutcome, EvaluationTime, ReportId, Name FROM
ReportEvent WHERE QueriedEntities='Patent__c'
```

SEE ALSO:

Big Objects Implementation Guide

ReportEventStream

Tracks report-related actions, such as when a user runs or exports a report. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✔
Streaming API (CometD)	✔

Subscription Channel

`/event/ReportEventStream`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`ColumnHeaders`	**Type** string **Properties** Nillable **Description** Comma-separated values of column headers of the report. Values listed are object names, field names, and field values except where aliases are used. For example, [Opportunity.Name, Opportunity.Type, Opportunity.Owner.UserRole.RollupDescription, Opportunity.Account.Name, Opportunity.Account.NumberOfEmployees, AGE].
`DashboardId`	**Type** string **Properties** Nillable

Field	Details
	Description The ID of the dashboard that the report was part of. For example, 01ZB0000000PmoQ.
`DashboardName`	**Type** string **Properties** Nillable **Description** The title of the dashboard that the report was part of.
`Description`	**Type** string **Properties** Nillable **Description** The description of the report.
`DisplayedFieldEntities`	**Type** string **Properties** Nillable **Description** The API values of the fields that are displayed on the report, including the names of the entities of the grouped column fields. For example, `[ACCOUNTS, OWNERS]`.
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the specified report event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.

Field	Details
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event. For example, 0a4779b0-0da1-4619-a373-0a36991dff90.
`EventSource`	**Type** picklist **Properties** Nillable, Restricted Picklist **Description** The source of the event. Possible values are: • `API`—The user generated the report from an API call. • `Classic`—The user generated the report from the Salesforce Classic UI. • `Lightning`—The user generated the report from Lightning Experience.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`ExecutionIdentifier`	**Type** string **Properties** Nillable **Description** When report data is divided into multiple report events, use this unique identifier to correlate the multiple data chunks. For example, if each chunk has the same `ExecutionIdentifier` of a50a4025-84f2-425d-8af9-2c780869f3b5, you can link them together to get all the data for the report execution. The `Sequence` field contains the incremental sequence numbers that indicate the order of the multiple events. For more information, see `Sequence`.
`ExportFileFormat`	**Type** string **Properties** Nillable

Field	Details
	Description If the user exported the report, this value indicates the format of the exported report. Possible values are: • CSV • Excel
`Format`	**Type** picklist **Properties** Defaulted on create, Nillable, Restricted picklist **Description** The format of the report. Possible values are: • `Matrix` • `MultiBlock` • `Summary` • `Tabular`
`GroupedColumnHeaders`	**Type** string **Properties** Nillable **Description** Comma-separated values of grouped column fields in summary, matrix, and joined reports. For example, `[USERNAME, ACCOUNT.NAME, TYPE, DUE_DATE, LAST_UPDATE, ADDRESS1_STATE]`.
`IsScheduled`	**Type** boolean **Properties** Defaulted on create **Description** If TRUE, the report was scheduled. If FALSE, the report wasn't scheduled.
`LoginHistoryId`	**Type** reference **Properties** Nillable **Description** Tracks a user session so that you can correlate user activity with a particular series of report events. This field is also available on the LoginHistory, AuthSession, and LoginHistory objects, making it easier to trace events back to a user's original authentication. This value is null if

Field	Details
	the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report. For example, 0YaB000002knVQLKA2.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. This value is null if the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report. For example, lUqjLPQTWRdvRG4.
`Name`	**Type** string **Properties** Nillable **Description** The display name of the report. The value is null for report previews.
`NumberOfColumns`	**Type** int **Properties** Nillable **Description** The number of columns in the report.
`Operation`	**Type** picklist **Properties** Nillable, Restricted Picklist **Description** The context in which the report executed, such as from a UI (Classic, Lightning, Mobile), through an API (synchronous, asynchronous, Apex), or through a dashboard. Possible values are: • `ChartRenderedInEmbeddedAnalyticsApp`—Report executed from a rendered chart in an embedded Analytics app. • `ChartRenderedOnHomePage`—Report executed from a rendered chart on the home page. • `ChartRenderedOnVisualforcePage`—Report executed from a rendered chart on a VisualForce Page.

Field	Details
	• `DashboardComponentPreviewed`—Report executed from a Lightning dashboard component preview. • `DashboardComponentUpdated`—Report executed when a user refreshed a dashboard component. Because the report resulted from an asynchronous operation, session information, contained in the fields `SessionKey`, `LoginKey`, `SessionLevel`, and `SourceIp`, isn't captured. • `ProbeQuery`—Report executed from a probe query. • `ReportAddedToCampaign`—Report was added from an Add to Campaign action. • `ReportExported`—Report executed from a printable view or report export that wasn't asynchronous nor an API export. • `ReportExportedAsynchronously`—Report was exported asynchronously. • `ReportExportedUsingExcelConnector`—Report was exported using the Excel connector. • `ReportOpenedFromMobileDashboard`—Report executed when a user clicked a dashboard component on a mobile device and drilled down to a report. • `ReportPreviewed`—Report executed when a user got preview results while using the report builder. • `ReportResultsAddedToEinsteinDiscovery`—Report executed synchronously from Einstein Discovery. • `ReportResultsAddedToWaveTrending`—Report executed when a user trended a report in Einstein Analytics. • `ReportRunAndNotificationSent`—Report executed through the notifications API. • `ReportRunFromClassic`—Report executed from the Run Report option of Salesforce Classic. • `ReportRunFromLightning`—Report executed from the Run option in Lightning Experience from a non-mobile browser. • `ReportRunFromMobile`—Report executed from the Run Report option of the mobile Salesforce app. • `ReportRunFromReportingSnapshot`—Report executed through Snapshot Analytics. • `ReportRunFromRestApi`—Report executed from REST API. • `ReportRunUsingApexAsynchronousApi`—Report executed from the asynchronous Apex API. • `ReportRunUsingApexSynchronousApi`—Report executed from the synchronous Apex API. • `ReportRunUsingAsynchronousApi`—Report executed from an asynchronous API. • `ReportRunUsingSynchronousApi`—Report executed from a synchronous API. • `ReportScheduled`—Report was scheduled. • `Test`—Report execution resulted from a test. • `Unknown`—Report execution origin is unknown.

Field	Details
`OwnerId`	**Type** string **Properties** Nillable **Description** The ID of the folder, organization, or user who owns the report. This value is blank if the report wasn't saved. For example, 005B0000001vURvIAM.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Block`—The user was blocked from performing the operation that triggered the policy. • `Error`—The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `FailedInvalidPassword`—The user entered an invalid password. • `FailedPasswordLockout`—The user entered an invalid password too many times. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user wasn't blocked from performing the operation. • `NoAction`—The policy didn't trigger. • `Notified`—A notification was sent to the recipient. • `TwoFAAutomatedSuccess`—Salesforce Authenticator approved the request for access because the request came from a trusted location. After users enable location services in Salesforce Authenticator, they can designate trusted locations. When a user trusts a location for a particular activity, such as logging in from a recognized device, that activity is approved from the trusted location for as long as the location is trusted.

Field	Details
	• `TwoFADenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator. • `TwoFAFailedGeneralError`—An error caused by something other than an invalid verification code, too many verification attempts, or authenticator app connectivity. • `TwoFAFailedInvalidCode`—The user provided an invalid verification code. • `TwoFAFailedTooManyAttempts`—The user attempted to verify identity too many times. For example, the user entered an invalid verification code repeatedly. • `TwoFAInitiated`—Salesforce initiated identity verification but hasn't yet challenged the user. • `TwoFAInProgress`—Salesforce challenged the user to verify identity and is waiting for the user to respond or for Salesforce Authenticator to send an automated response. • `TwoFANoAction`—The policy specifies multi-factor authentication (formerly called two-factor authentication) as an action, but the user is already in a high-assurance session. • `TwoFARecoverableError`—Salesforce can't reach the authenticator app to verify identity, but will retry. • `TwoFAReportedDenied`—The user denied the approval request in the authenticator app, such as Salesforce Authenticator, and also flagged the approval request to report to an administrator. • `TwoFASucceeded`—The user's identity was verified. This field isn't populated until all transaction security policies are processed for the real-time event.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The entities in the SOQL query. For example, Opportunity, Lead, Account, or Case. Can also include custom objects. For relationship queries, the value of this field contains all entities involved in the query. If the query returns 0 records, then the value of this field is null. **Examples** • For `SELECT Contact.FirstName, Contact.Account.Name from Contact`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Account.Name, (SELECT Contact.FirstName, Contact.LastName FROM Account.Contacts) FROM Account`, the value of `QueriedEntities` is `Account, Contact`. • For `SELECT Id, Name, Account.Name FROM Contact WHERE Account.Industry = 'media'`, the value of `QueriedEntities` is `Account, Contact`.
`Records`	**Type** json

Field	Details
	Properties Nillable **Description** A JSON string that represents the report's data. For example, `{"totalSize":1,"rows":[{"datacells":["005B0000001vURv","001B000000fewai"]}]}`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the `EventIdentifier` of the related event. For example, bd76f3e7-9ee5-4400-9e7f-54de57ecd79c. This field is populated only when the activity that this event monitors requires extra authentication, such as multi-factor authentication. In this case, Salesforce generates more events and sets the `RelatedEventIdentifier` field of the new events to the value of the `EventIdentifier` field of the original event. Use this field with the `EventIdentifier` field to correlate all the related events. If no extra authentication is required, this field is blank.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`ReportId`	**Type** string **Properties** Nillable **Description** The ID of the report associated with this event. For example, 00OB00000032FHdMAM.
`RowsProcessed`	**Type** double **Properties** Nillable

Field	Details
	Description The total number of rows returned in the report. When report data is divided into multiple report events, this value is the same for all data chunks. For more information, see `ExecutionIdentifier`.
`Scope`	**Type** string **Properties** Nillable **Description** Defines the scope of the data on which the user ran the report. For example, users can run the report against all opportunities, opportunities they own, or opportunities their team owns. Possible values are: • `user`—User owns the objects the report was run against. • `team`—Team owns the objects the report was run against. • `organization`—Report was run against all applicable objects.
`Sequence`	**Type** int **Properties** Nillable **Description** Incremental sequence number that indicates the order of multiple events that result from a given report execution. When a report execution returns many records, Salesforce splits this data into chunks based on the size of the records, and then creates multiple correlated ReportEventStreams. The field values in each of these correlated ReportEventStreams are the same, except for `Records`, which contains the different data chunks, and `Sequence`, which identifies each chunk in order. Every report execution has a unique `ExecutionIdentifier` value to differentiate it from other report executions. To view all the data chunks from a single report execution, use the `Sequence` and `ExecutionIdentifier` fields in combination. When a report executes, we provide the first 1,000 events with data in the `Records` field. Use the `ReportId` field to view the full report. For more information, see `Sequence`.
`SessionKey`	**Type** string **Properties** Nillable

Field	Details
	Description The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. This value is null if the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report. For example, vMASKIU6AxEr+Op5.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. This low level isn't available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels. This value is null if the event that was generated was from a dashboard refresh, a multi-block report, or a scheduled report.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, 126.7.4.2.
`UserId`	**Type** reference **Properties** Nillable **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string

Field	Details
	Properties Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

SessionHijackingEvent

Tracks when unauthorized users gain ownership of a Salesforce user's session with a stolen session identifier. To detect such an event, Salesforce evaluates how significantly a user's current browser fingerprint diverges from the previously known fingerprint using a probabilistically inferred significance of change. This object is available in API version 49.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	✓
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/SessionHijackingEvent`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`CurrentIp`	**Type** string **Properties** Nillable **Description** The IP address of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousIp` field for the previous IP address. If the IP address didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousIp` field value. For example, `126.7.4.2`.
`CurrentPlatform`	**Type** string **Properties** Nillable **Description** The platform of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousPlatform` field for the previous platform. If the platform didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousPlatform` field value. For example, `MacIntel` or `Win32`.
`CurrentScreen`	**Type** string **Properties** Nillable **Description** The screen of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousScreen` field for the previous screen. If the screen didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousScreen` field value. For example, `(900.0,1440.0)` or `(720,1280)`.
`CurrentUserAgent`	**Type** textarea **Properties** Nillable **Description** The user agent of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session

Field	Details
	hijacking attack has occurred. See the `PreviousUserAgent` field for the previous user agent. If the user agent didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousUserAgent` field value. For example, `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36`.
`CurrentWindow`	**Type** string **Properties** Nillable **Description** The browser window of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousWindow` field for the previous window. If the window didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousWindow` field value. For example, `(1200.0,1920.0)`.
`EvaluationTime`	**Type** double **Properties** Nillable **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the anomaly was detected. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.

Field	Details
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`PolicyId`	**Type** reference **Properties** Nillable **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event. A relationship field. **Relationship Name** Policy **Relationship Type** Lookup **Refers To** TransactionSecurityPolicy
`PolicyOutcome`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The result of the transaction policy. Possible values are: • `Error` - The policy caused an undefined error when it executed.

Field	Details
	• `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient. This field isn't populated until all transaction security policies are processed for the real-time event.
`PreviousIp`	**Type** string **Properties** Nillable **Description** The IP address of the previous fingerprint. The IP address of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentIp` field for the newly observed IP address. For example, `128.7.5.2`.
`PreviousPlatform`	**Type** string **Properties** Nillable **Description** The platform of the previous fingerprint. The platform of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentPlatform` field for the newly observed platform. For example, `Win32` or `iPhone`.
`PreviousScreen`	**Type** string **Properties** Nillable **Description** The screen of the previous fingerprint. The screen of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentScreen` field for the newly observed screen. For example, `(1200.0,1920.0)`.
`PreviousUserAgent`	**Type** textarea

Field	Details
	Properties Nillable **Description** The user agent of the previous fingerprint. The user agent of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentUserAgent` field for the newly observed user agent. For example, `Mozilla/5.0 (iPhone; CPU iPhone OS 13_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko)`.
`PreviousWindow`	**Type** string **Properties** Nillable **Description** The browser window of the previous fingerprint. The window of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentWindow` field for the newly observed window. For example, `(1600.0,1920.0)`.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.
`Score`	**Type** double **Properties** Nillable **Description** Specifies how significant the new browser fingerprint deviates from the previous one. The score is a number from 0.0 (lowest amount of deviation) through 1.0 (highest amount of deviation). The event exposes five field pairs (such as `CurrentIp` and `PreviousIp`) to view the before and after data for the five most interesting browser features that contributed to this anomaly. See the `SecurityEventData` field for all contributing features in JSON format. Salesforce detects session hijacking by comparing browser fingerprints in a given user session and evaluating how significantly a newly observed fingerprint deviates from the existing

Field	Details
	one. A large deviation score (0.8 or more) between two intra-session fingerprints indicates that two different browsers are active in the same session. The presence of two active browsers usually means that session hijacking has occurred.
`SecurityEventData`	**Type** string **Properties** Nillable **Description** The set of browser fingerprint features about the session hijacking that triggered this event. See the Threat Detection documentation for the list of possible features. For example, let's say that a user's current browser fingerprint diverges from their previously known fingerprint. If Salesforce concludes their session was hijacked, it fires this event and the contributing features are captured in this field in JSON format. Each feature describes a particular browser fingerprint property, such as the browser user agent, window, or platform. The data includes the current and previous values for each feature. **Example** `[{ "featureName": "userAgent", "featureContribution": "0.45 %", "previousValue": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142", "currentValue": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36." }, { "featureName": "ipAddress", "featureContribution": "0.23 %", "previousValue": "201.17.237.77", "currentValue": "182.64.210.144" }, { "featureName": "platform", "featureContribution": "0.23 %", "previousValue": "Win32", "currentValue": "MacIntel" }, { "featureName": "screen", "featureContribution": "0.23 %", "previousValue":"(1050.0,1680.0)", "currentValue": "(864.0,1536.0)" }, { "featureName": "window",`

Field	Details

	<pre>"featureContribution": "0.17 %",
	"previousValue": "1363x1717",
	"currentValue": "800x1200"
	}
	]</pre>

SessionKey	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
SourceIp	**Type** string **Properties** Nillable **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
Summary	**Type** textarea **Properties** Nillable **Description** A text summary of the threat that caused this event to be created. The summary lists the browser fingerprint features that most contributed to the threat detection along with their contribution to the total score. **Example** • `Changes to (userAgent, platform, ipAddress) were not expected based on this user's profile. These top 3 deviations contributed (1, 1, 0.922) to the total score, respectively` • `Changes to (ipAddress, userAgent, platform, languages, color) were not expected based on this user's profile. These top 5 deviations contributed (1, 0.695, 0.695, 0.25, 0.223) to the total score, respectively`
UserId	**Type** reference **Properties** Nillable

Field	Details
	Description The origin user's unique ID. For example, `005000000000123`. A polymorphic relationship field. **Relationship Name** User **Relationship Type** Lookup **Refers To** User
`Username`	**Type** string **Properties** Nillable **Description** The origin username in the format of `user@company.com` at the time the event was created.

SessionHijackingEventStore

Tracks when unauthorized users gain ownership of a Salesforce user's session with a stolen session identifier. To detect such an event, Salesforce evaluates how significantly a user's current browser fingerprint diverges from the previously known fingerprint using a probabilistically inferred significance of change. SessionHijackingEventStore is an object that stores the event data of SessionHijackingEvent. This object is available in API version 49.0 and later.

Supported Calls

`describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Fields

Field	Details
`CurrentIp`	**Type** string **Properties** Filter, Group, Nillable, Sort

Field	Details
	Description The IP address of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousIp` field for the previous IP address. If the IP address didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousIp`field value. For example, `126.7.4.2.`
`CurrentPlatform`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The platform of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousPlatform` field for the previous platform. If the platform didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousPlatform` field value. For example, `MacIntel` or `Win32`.
`CurrentScreen`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The screen of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousScreen` field for the previous screen. If the screen didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousScreen` field value. For example, `(900.0,1440.0)` or `(720,1280)`.
`CurrentUserAgent`	**Type** textarea **Properties** Nillable **Description** The user agent of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousUserAgent` field for the previous user agent. If the user agent didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousUserAgent` field value. For example, `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36`.

Field	Details
`CurrentWindow`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The browser window of the newly observed fingerprint that deviates from the previous fingerprint. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `PreviousWindow` field for the previous window. If the window didn't contribute to the observed fingerprint deviation, the value of this field is the same as the `PreviousWindow` field value. For example, `(1200.0,1920.0)`.
`EvaluationTime`	**Type** double **Properties** Filter, Nillable, Sort **Description** The amount of time it took to evaluate the policy in milliseconds. This field isn't populated until all transaction security policies are processed for the real-time event.
`EventDate`	**Type** dateTime **Properties** Filter, Sort **Description** Required. The time when the anomaly was detected. For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Filter, Group, Sort **Description** Required. The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`LastReferencedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The timestamp for when the current user last viewed a record related to this record.

Field	Details
`LastViewedDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The timestamp for when the current user last viewed this record. If this value is null, it's possible that this record was referenced (`LastReferencedDate`) and not viewed.
`LoginKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, lUqjLPQTWRdvRG4.
`PolicyId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** The ID of the transaction policy associated with this event. For example, 0NIB000000000KOOAY. This field isn't populated until all transaction security policies are processed for the real-time event.
`PolicyOutcome`	**Type** picklist **Properties** Filter, Group, Nillable, Restricted picklist, Sort **Description** The result of the transaction policy. Possible values are: • `Error` - The policy caused an undefined error when it executed. • `ExemptNoAction`—The user is exempt from transaction security policies, so the policy didn't trigger. • `MeteringBlock`—The policy took longer than 3 seconds to process, so the user was blocked from performing the operation. • `MeteringNoAction`—The policy took longer than 3 seconds to process, but the user isn't blocked from performing the operation. • `NoAction` - The policy didn't trigger. • `Notified` - A notification was sent to the recipient.

Field	Details
	This field isn't populated until all transaction security policies are processed for the real-time event.
`PreviousIp`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The IP address of the previous fingerprint. The IP address of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentIp` field for the newly observed IP address. For example, `128.7.5.2`.
`PreviousPlatform`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The platform of the previous fingerprint. The platform of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentPlatform` field for the newly observed platform. For example, `Win32` or `iPhone`.
`PreviousScreen`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The screen of the previous fingerprint. The screen of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentScreen` field for the newly observed screen. For example, `(1200.0,1920.0)`.
`PreviousUserAgent`	**Type** textarea **Properties** Nillable **Description** The user agent of the previous fingerprint. The user agent of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentUserAgent` field for the newly observed user agent. For example, `Mozilla/5.0 (iPhone; CPU`

Field	Details
	`iPhone OS 13_0 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko).`
`PreviousWindow`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The browser window of the previous fingerprint. The window of the newly observed fingerprint deviates from this value. The difference between the current and previous values is one indicator that a session hijacking attack has occurred. See the `CurrentWindow` field for the newly observed window. For example, `(1600.0,1920.0)`.
`Score`	**Type** double **Properties** Filter, Nillable, Sort **Description** Specifies how significant the new browser fingerprint deviates from the previous one. The score is a number from 0.0 (lowest amount of deviation) through 1.0 (highest amount of deviation). The event exposes five field pairs (such as `CurrentIp` and `PreviousIp`) to view the before and after data for the five most interesting browser features that contributed to this anomaly. See the `SecurityEventData` field for all contributing features in JSON format. Salesforce detects session hijacking by comparing browser fingerprints in a given user session and evaluating how significantly a newly observed fingerprint deviates from the existing one. A large deviation score (0.8 or more) between two intra-session fingerprints indicates that two different browsers are active in the same session. The presence of two active browsers usually means that session hijacking has occurred.
`SecurityEventData`	**Type** textarea **Properties** Nillable **Description** The set of browser fingerprint features about the session hijacking that triggered this event. See the Threat Detection documentation for the list of possible features. For example, let's say that a user's current browser fingerprint diverges from their previously known fingerprint. If Salesforce concludes their session was hijacked, it fires this event and the contributing features are captured in this field in JSON format. Each feature describes a particular browser fingerprint property, such as the browser user agent, window, or platform. The data includes the current and previous values for each feature.

Field	Details

	Example
	<pre>[
	{
	"featureName": "userAgent",
	"featureContribution": "0.45 %",
	"previousValue": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142",
	
	"currentValue": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36."
	},
	{
	"featureName": "ipAddress",
	"featureContribution": "0.23 %",
	"previousValue": "201.17.237.77",
	"currentValue": "182.64.210.144"
	},
	{
	"featureName": "platform",
	"featureContribution": "0.23 %",
	"previousValue": "Win32",
	"currentValue": "MacIntel"
	},
	{
	"featureName": "screen",
	"featureContribution": "0.23 %",
	"previousValue":"(1050.0,1680.0)",
	"currentValue": "(864.0,1536.0)"
	},
	{
	"featureName": "window",
	"featureContribution": "0.17 %",
	"previousValue": "1363x1717",
	"currentValue": "800x1200"
	}
	]</pre>

SessionHijackingEventNumber	**Type** string **Properties** Autonumber, Defaulted on create, Filter, idLookup, Sort **Description** The unique number assigned by the system after the event is received in Salesforce. This ID is different than the replayID field on the streaming event SessionHijackingEvent. You can't change the format or value for this field.

Field	Details
`SessionKey`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started. For example, vMASKIU6AxEr+Op5.
`SourceIp`	**Type** string **Properties** Filter, Group, Nillable, Sort **Description** The source IP address of the client that logged in. For example, `126.7.4.2`.
`Summary`	**Type** textarea **Properties** Nillable **Description** A text summary of the threat that caused this event to be created. The summary lists the browser fingerprint features that most contributed to the threat detection along with their contribution to the total score. **Example** • `Changes to (userAgent, platform, ipAddress) were not expected based on this user's profile. These top 3 deviations contributed (1, 1, 0.922) to the total score, respectively` • `Changes to (ipAddress, userAgent, platform, languages, color) were not expected based on this user's profile. These top 5 deviations contributed (1, 0.695, 0.695, 0.25, 0.223) to the total score, respectively`
`UserId`	**Type** reference **Properties** Filter, Group, Nillable, Sort **Description** The origin user's unique ID. For example, `005000000000123`.
`Username`	**Type** string

Field	Details
	Properties Filter, Group, Nillable, Sort **Description** The origin username in the format of `user@company.com` at the time the event was created.

Associated Object

This object has the following associated object. It's available in the same API version as this object.

SessionHijackingEventStoreFeed
Feed tracking is available for the object.

UriEvent

Detects when a user creates, accesses, updates, or deletes a record in Salesforce Classic only. Doesn't detect record operations done through a Visualforce page or Visualforce page views. UriEvent and is a big object that stores the event data of UriEventStream. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`, `query()`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Note: UriEvent doesn't track Setup events.

Fields

Field	Details
`EventDate`	**Type** dateTime **Properties** Filter. Sort **Description** The time when the specified URI event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string

Field	Details
	Properties Filter, Sort **Description** The unique ID of the event. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`.
`LoginKey`	**Type** string **Properties** Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, `8gHOMQu+xvjCmRUt`.
`Message`	**Type** string **Properties** Nillable **Description** The failure message if the operation being performed on the entity failed (`OperationStatus`=FAILURE).
`Name`	**Type** string **Properties** Nillable **Description** The value of the record being viewed/edited.
`Operation`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The operation being performed on the entity. For example, `Read, Create, Update`. or `Delete`. Create and update operations are captured in pairs; that is, expect two event records for each operation. The first record represents the start of the operation, and the second record represents whether the operation was successful or not.

Field	Details
	If there isn't a second event recorded for a create or update operation, then the user canceled the operation, or the operation failed with client-side validation (for example, when a required field is empty).
`OperationStatus`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Whether the operation performed on the entity (such as create) succeeded or failed. When the operation starts, the value is always INITIATED. Possible values are: • `Failure`—The operation failed. • `Initiated`—The operation started. Note: Create and update operations can generate an extra `OperationStatus=Initiated` event after an operation fails. Ignore this extra record. • `Success`—The operation succeeded.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The API name of the objects referenced by the URI.
`RecordId`	**Type** reference **Properties** Nillable **Description** The ID of the record being viewed or edited. For example, `001RM000003cjx6YAA`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the EventIdentifier of the related event.
`SessionKey`	**Type** string

Field	Details
	Properties Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level will experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client logging in. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005RM000001ctYJYAY`.
`UserName`	**Type** string

Field	Details
	Properties Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`UserType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license. Each `UserType` is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values are: • `CsnOnly`—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • `CspLitePortal`—CSP Lite Portal license. Users whose access is limited because they are organization customers and access the application through a customer portal or Experience Cloud site. • `CustomerSuccess`—Customer Success license. Users whose access is limited because they are organization customers and access the application through a customer portal. • `Guest` • `PowerCustomerSuccess`—Power Customer Success license. Users whose access is limited because they are organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • `PowerPartner`—Power Partner license. Users whose access is limited because they are partners and typically access the application through a partner portal or site. • `SelfService` • `Standard`—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses.

Standard SOQL Usage

UriEvent allows filtering over two fields: `EventDate` and `EventIdentifier`. The only supported SOQL functions on the UriEvent object are `WHERE`, `ORDER BY`, and `LIMIT`. In the `WHERE` clause, you can only use comparison operators (<, >, <=, and >=). The != operator isn't supported. In the `ORDER BY` clause, you can only use EventDate `DESC`. Ascending order isn't supported with EventDate, and EventIdentifier sorting isn't supported.

Note: Date functions such as `convertTimeZone()` aren't supported—for example, `SELECT CALENDAR_YEAR(EventDate), Count(Id) FROM UriEvent GROUP BY CALENDAR_YEAR(EventDate)` returns an error. You can use date literals in your queries and some date/time functions like `TODAY()`, `YESTERDAY()`, and `LAST_n_DAYS:1`. However, these functions use comparison operators behind the scenes. Therefore you can only use them in the final expression in the `WHERE` clause.

The following list provides some examples of valid queries:

- **Unfiltered**
  - **Valid**—Contains no `WHERE` clause, so no special rules apply.

```
SELECT EntityType, UserName, UserType
FROM UriEvent
```

- **Filtered on** `EventDate`—you can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.
  - **Valid**—you can filter solely on `EventDate`, but single filters on other fields fail. You can also use a comparison operator in this query type.

```
SELECT EntityType, UserName, UserType
FROM UriEvent
WHERE EventDate>=2014-11-27T14:54:16.000Z
```

Async SOQL Usage

With Async SOQL, you can filter on any field in UriEvent and use any comparison operator in your query.

Find who is accessing Opportunities and related Contacts

```
SELECT EventDate, EventIdentifier, UserName, UserType, Name, EntityType, Operation,
LoginKey, SessionKey FROM UriEvent WHERE RecordId='001B000000AkcHxIAJ'
```

SEE ALSO:

LightningUriEvent

Big Objects Implementation Guide

UriEventStream

Detects when a user creates, accesses, updates, or deletes a record in Salesforce Classic only. Doesn't detect record operations done through a Visualforce page or Visualforce page views. This object is available in API version 46.0 and later.

Supported Calls

`describeSObjects()`

Supported Subscribers

Subscriber	Supported?
Apex Triggers	
Flows	
Processes	
Pub/Sub API	✓
Streaming API (CometD)	✓

Subscription Channel

`/event/UriEventStream`

Special Access Rules

Accessing this object requires either the Salesforce Shield or Salesforce Event Monitoring add-on subscription and the View Real-Time Event Monitoring Data user permission.

Note: UriEventStream doesn't track Setup events.

Event Delivery Allocation Enforced

No

Fields

Field	Details
`EventDate`	**Type** dateTime **Properties** Nillable **Description** The time when the specified URI event was captured (after query execution takes place). For example, `2020-01-20T19:12:26.965Z`. Milliseconds are the most granular setting.
`EventIdentifier`	**Type** string **Properties** Nillable **Description** The unique ID of the event, which is shared with the corresponding storage object. For example, `0a4779b0-0da1-4619-a373-0a36991dff90`. Use this field to correlate the event with its storage object.
`EventUuid`	**Type** string **Properties** Nillable **Description** A universally unique identifier (UUID) that identifies a platform event message. This field is available in API version 52.0 and later.
`LoginKey`	**Type** string

Field	Details
	Properties Nillable **Description** The string that ties together all events in a given user's login session. The session starts with a login event and ends with either a logout event or the user session expiring. For example, 8gHOMQu+xvjCmRUt
`Message`	**Type** string **Properties** Nillable **Description** The failure message if the operation being performed on the entity failed (`OperationStatus=Failure`).
`Name`	**Type** string **Properties** Nillable **Description** The value of the record being viewed or edited.
`Operation`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The operation being performed on the entity. For example, `Read, Create, Update`, or `Delete`. Create and update operations are captured in pairs; that is, expect two event records for each operation. The first record represents the start of the operation, and the second record represents whether the operation was successful or not. The two records are correlated by `RelatedEventIdentifier`. If there isn't a second event recorded for a create or update operation, then the user canceled the operation, or the operation failed with client-side validation (for example, when a required field is empty).
`OperationStatus`	**Type** picklist **Properties** Nillable, Restricted picklist

Field	Details
	Description Whether the operation performed on the entity (such as create) succeeded or failed. When the operation starts, the value is always INITIATED. Possible values are: • `Failure`—The operation failed. • `Initiated`—The operation started. Note: Create and update operations can generate an extra `OperationStatus=Initiated` event after an operation fails. Ignore this extra record. • `Success`—The operation succeeded.
`QueriedEntities`	**Type** string **Properties** Nillable **Description** The API name of the objects referenced by the URI.
`RecordId`	**Type** string **Properties** Nillable **Description** The id of the record being viewed or edited. For example, `001RM000003cjx6YAA`.
`RelatedEventIdentifier`	**Type** string **Properties** Nillable **Description** Represents the EventIdentifier of the related event.
`ReplayId`	**Type** string **Properties** Nillable **Description** Represents an ID value that is populated by the system and refers to the position of the event in the event stream. Replay ID values aren't guaranteed to be contiguous for consecutive events. A subscriber can store a replay ID value and use it on resubscription to retrieve missed events that are within the retention window.

Field	Details
`SessionKey`	**Type** string **Properties** Nillable **Description** The user's unique session ID. Use this value to identify all user events within a session. When a user logs out and logs in again, a new session is started.
`SessionLevel`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** Session-level security controls user access to features that support it, such as connected apps and reporting. Possible values are: • `HIGH_ASSURANCE`—A high assurance session was used for resource access. For example, when the user tries to access a resource such as a connected app, report, or dashboard that requires a high-assurance session level. • `LOW`—The user's security level for the current session meets the lowest requirements. Note: This low level is not available, nor used, in the Salesforce UI. User sessions through the UI are either standard or high assurance. You can set this level using the API, but users assigned this level will experience unpredictable and reduced functionality in their Salesforce org. • `STANDARD`—The user's security level for the current session meets the Standard requirements set in the org's Session Security Levels.
`SourceIp`	**Type** string **Properties** Nillable **Description** The source IP address of the client logging in. For example, `126.7.4.2`.
`UserId`	**Type** reference **Properties** Nillable **Description** The user's unique ID. For example, `005RM000001ctYJYAY`. This is a polymorphic relationship field.

Field	Details
	Relationship Name User **Relationship Type** Lookup **Refers To** User
`UserName`	**Type** string **Properties** Nillable **Description** The username in the format of `user@company.com` at the time the event was created.
`UserType`	**Type** picklist **Properties** Nillable, Restricted picklist **Description** The category of user license. Each `UserType` is associated with one or more UserLicense records. Each UserLicense is associated with one or more profiles. Valid values are: • `CsnOnly`—Users whose access to the application is limited to Chatter. This user type includes Chatter Free and Chatter moderator users. • `CspLitePortal`—CSP Lite Portal license. Users whose access is limited because they are organization customers and access the application through a customer portal or an Experience Cloud site. • `CustomerSuccess`—Customer Success license. Users whose access is limited because they are organization customers and access the application through a customer portal. • `Guest` • `PowerCustomerSuccess`—Power Customer Success license. Users whose access is limited because they are organization customers and access the application through a customer portal. Users with this license type can view and edit data they directly own or data owned by or shared with users below them in the customer portal role hierarchy. • `PowerPartner`—Power Partner license. Users whose access is limited because they are partners and typically access the application through a partner portal or site. • `SelfService`

Field	Details
	• `Standard`—Standard user license. This user type also includes Salesforce Platform and Salesforce Platform One user licenses.

SEE ALSO:

LightningUriEventStream