Salesforce Order Management Developer Guide

Version 66.0, Spring ’26

Summer ’26

Last updated: March 31, 2026

CONTENTS

CHAPTER 1 Salesforce Order Management Developer Guide

In this chapter ...

Customize Salesforce Order Management and integrate it with your storefront, inventory, and fulfillment systems.

Order Management Developer Resources

In addition to the *Order Management Developer Guide*, these documentation resources can help you customize Salesforce and the Order Management app.

Connect in Apex Classes

- FulfillmentOrder Class
- OmnichannelInventoryService Class
- OrderPaymentSummary Class
- OrderSummary Class
- OrderSummaryCreation Class
- Repricing Class
- ReturnOrder Class
- Routing Class

Connect REST API Resources

- B2B and B2B2C Commerce Resources
- Salesforce Omnichannel Inventory Resources
- Salesforce Order Management Resources

Salesforce Flow

- Flow Builder
- Salesforce Omnichannel Inventory Flow Core Actions
- Salesforce Order Management Flow Core Actions
- Salesforce Omnichannel Inventory Actions
- Salesforce Order Management Actions
- Presentation: Advanced Salesforce Order Management Flows (Video)
- Example Custom Flow: Remorse Delay
- Example Custom Flow: Bulk Order Cancel

Platform Events

FOStatusChangedEvent
FulfillOrdItemQtyChgEvent
OrderStatusChangedEvent
OrderSummaryCreatedEvent
OrderSumStatusChangedEvent
PendingOrdSumProcEvent
ProcessExceptionEvent

Implementation

Salesforce Order Management Implementation Guide for B2C Commerce (PDF)
Salesforce Order Management Implementation Guide for B2B and B2B2C Commerce (PDF)
Order Management with B2C Commerce Add Item Playbook (PDF)

Distributed Order Management Routing Package

Distributed Order Management Routing Package Documentation
Omnichannel Inventory Downtime Playbook

Commerce Payments

Data Models for CCS Order and Invoice Objects
CreditMemo
CreditMemoLine
Invoice
InvoiceLine
Data Models for Commerce Payment Objects
Payment
PaymentAuthorization

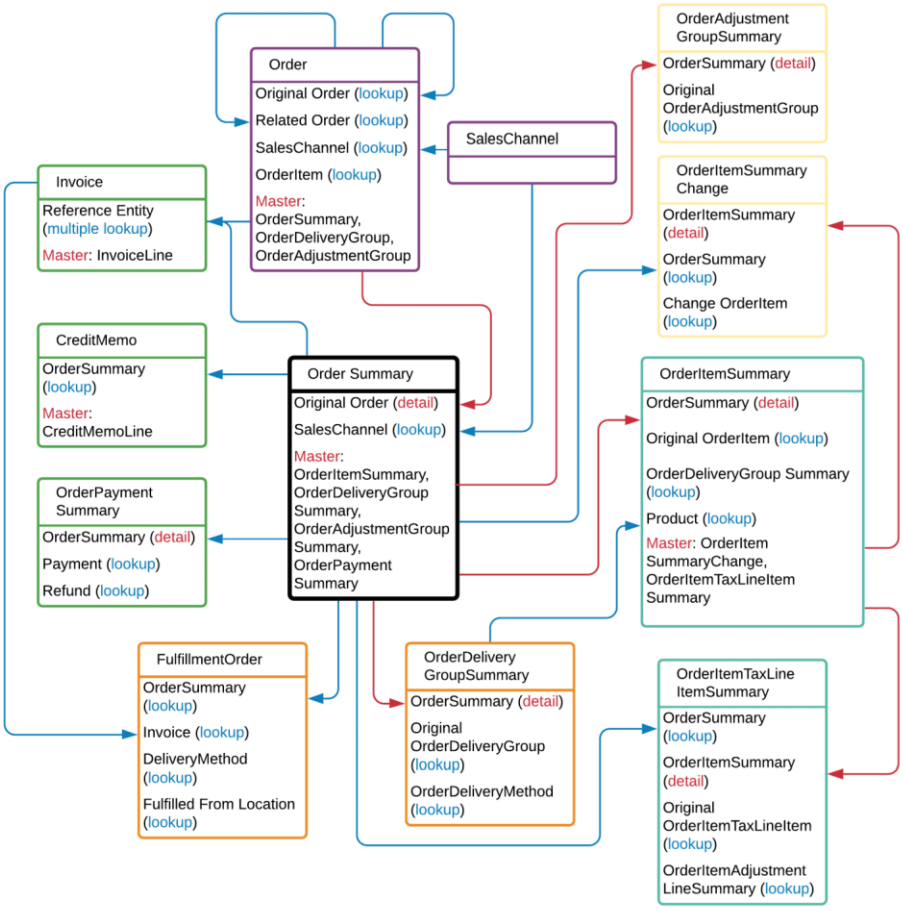
PaymentGateway
PaymentGatewayLog
PaymentGatewayProvider
PaymentGroup
PaymentLineInvoice
PaymentMethod
Commerce Payments Connect in REST Resources
Commerce Payments Resources
CommercePayments Apex Namespace
CommercePayments Namespace

Other Developer Resources

Order Management Partner Pocket Guide
Object Reference for Salesforce and Lightning Platform
Standard Lightning Page Components

Order Summary Entity Relationships

In Salesforce Order Management, each order is represented by an order summary and a number of other records linked to the order summary. This diagram illustrates some of the relationships between the OrderSummary object and other objects used in Salesforce Order Management.



To view a diagram of all object relationships in your organization, from Setup, in the Quick Find box, enter `Schema Builder`, and then select **Schema Builder**. For information on using Schema Builder, see Design Your Own Data Model in *Extend Salesforce with Clicks, not Code*.

Salesforce B2C Commerce Storefront Order Data Map

These tables illustrate how data in a Salesforce B2C Commerce order packet maps to records in Salesforce Order Management. If you're implementing your own storefront integration, this map can help you understand the order data requirements.

Integration Notes

General:

- B2C Commerce sends orders in New or Open status to Salesforce Order Management. To delay sending an order, for example, to perform a fraud check, keep it in Created status.
- B2C Commerce sends order data to Salesforce Order Management according to a set frequency and when the packet of pending orders reaches a certain size. The process is similar to the one used for Commerce Cloud Order Management. In addition, the integration contacts B2C Commerce every 5 minutes to request any pending order data.

- The B2C Integration can't create an order with a combined total of more than 200 OrderItems and OrderItemAdjustmentLineItems. For example, it can create an order with 150 OrderItems and 45 OrderItemAdjustmentLineItems, but can't create an order with 110 OrderItems and 100 OrderItemAdjustmentLineItems. To import an order that exceeds this limit, use Bulk API 2.0. For information about manually importing orders into Order Management, see Importing Order Data.
- The integration associates sites in B2C Commerce with SalesChannels in Order Management. It identifies the SalesChannel for an order by searching for a SalesChannelName value that matches the `site/site-id` value in the order data.
- For details of the B2C Commerce order data format, see the Order XSD Schema in the *B2C Commerce Infocenter*.
- The B2C Integration service user is an internal Salesforce user. If you set internal organization-wide sharing defaults for any objects accessed by the integration, set them to **Public Read/Write** access.

  Note: For information about organization-wide sharing defaults, see Organization-Wide Sharing Defaults in Salesforce Help.

- If the Salesforce org ID or login URL changes, or if you run DBInit on your B2C Commerce storefront, the integration requires updating. To reprovision the integration, open a case with Salesforce Customer Support as described in the *Order Management Implementation Guide*.
- If you refresh a sandbox org, or change the Tenant ID of a B2C Commerce instance, repeat the implementation process as described in the *Order Management Implementation Guide*. For information on sandbox orgs, see the sandbox topics in Salesforce Help.
- To deactivate the integration, open a case with Salesforce Support.
- To stop sending orders from a site, log in to Business Manager, navigate to **Merchant Tools** > **Site Preferences** > **Order**, and turn off the **Include in Order Management** setting.

Shopper Accounts:

- When the `order/billing-address/company-name` value in the order data is not null, Order Management always uses a standard account and contact to represent the shopper.
- If you enable Person Accounts, don't change the names or API names of the two default Account record types. The B2C Integration uses those types to represent shopper accounts.
- You can configure the integration to identify existing shoppers by applying your org's duplicate and matching rules for Accounts, Contacts, and Person Accounts. To enable this feature, in Setup, in Order Management Settings, turn on **B2C Integration Data Matching Rules**. For more information on configuring duplicate management, see Manage Duplicate Records in Salesforce Help.

  Note: If your org's matching rules identify multiple potential duplicate records, the integration selects the one with the highest match score. If multiple records tie for the highest match score, you can't guarantee which is selected.

  Note: Applying complex custom matching rules can affect the integration's performance.

- The integration can't look up any encrypted fields on Salesforce objects. Encrypting any of the fields it uses to look up records causes order ingestion to fail. If you want to encrypt the Account.Name, Account.PersonEmail, or Contact.Email field, then you must implement custom duplicate and matching rules that don't use any encrypted fields. For more information on encryption restrictions, see General Shield Platform Encryption Considerations in Salesforce Help.
- For detailed information on the storage and identification of shopper accounts, see Order Management Shopper Records in *Salesforce Order Management* in Salesforce Help.

International Considerations

- If the `order/taxation` value for an order is *`net`*, then the integration creates an OrderItemTaxLineItem record for each OrderItem and OrderItemAdjustmentLineItem record.
- If Salesforce state and country/territory picklists aren't enabled:
  - The `state-code` values in order data must match a standard 2-character ISO state or province code. The integration copies them to State fields on Salesforce records.

  - The `country-code` values in order data must match a standard 2-character ISO country/territory code. The integration copies them to Country fields on Salesforce records.

- If Salesforce state and country/territory picklists are enabled:
  - The `state-code` values in order data must match the state code of an entry in the Salesforce state picklist. The integration copies them to StateCode fields on Salesforce records. The corresponding State fields are set to the matching integration values from the picklist.
  - The `country-code` values in order data must match the country/territory code of an entry in the Salesforce country/territory picklist. The integration copies them to CountryCode fields on Salesforce records. The corresponding Country fields are set to the matching integration values from the picklist.

Payments

- Order data includes a paired payment instrument and payment transaction for each payment associated with the order.
- If an `order/payments/payment/transaction-type` value in the order data starts with *`auth`*, then the integration creates a PaymentAuthorization record for that transaction. If the value is *`sale`* or *`capture`*, then it creates a Payment record. These checks aren't case-sensitive.
- For a credit or debit card payment type, the `payments/payment/credit-card/card-type` value in the order data must match a value in the CardType picklist on the CardPaymentMethod object.
- To handle a payment method that doesn't match a CardPaymentMethod CardType or DigitalWallet Type, create a custom payment method as described later in this section.
- If you use Salesforce Payments, the integration sets different values for some of the fields on AlternativePaymentMethod, CardPaymentMethod, and DigitalWallet. The details are described in the map tables.
- The integration recognizes certain custom attributes on `order/payments/payment` and copies them to standard fields on the PaymentGatewayLog record in Salesforce. To take advantage of this feature, create and populate custom attributes with the following exact names on the Order Payment Transaction object in B2C Commerce:
  - `authCode` (value copied to PaymentGatewayLog.GatewayAuthCode)
  - `avsResultCode` (value copied to PaymentGatewayLog.GatewayAvsCode)
  - `approvalStatus` (value copied to PaymentGatewayLog.GatewayResultCode)
- The integration can copy other custom attributes on `order/payments/payment` to standard fields on AlternativePaymentMethod, CardPaymentMethod, DigitalWallet, Payment, and PaymentAuthorization records in Salesforce.
- An OrderPaymentSummary for a B2C Commerce order only has a FullName value if the payment method is a credit card. Other payment methods don't include the data value that the integration uses to set the FullName.

Promotions

- The integration creates an OrderAdjustmentGroup record for each order-level `price-adjustments/price-adjustment/promotion-id` and for each item-level `price-adjustments/price-adjustment/promotion-id` that applies to more than one item in the order. For each OrderAdjustmentGroup, it creates an OrderItemAdjustmentLineItem record for each OrderItem in the order that the associated promotion applies to. If an OrderItem is affected by multiple promotions, then it can have an OrderItemAdjustmentLineItem for each one.
- If the `order/taxation` value for an order is *`net`*, then the integration creates an OrderItemTaxLineItem record for each OrderItemAdjustmentLineItem record.

For information on the Salesforce state and country/territory picklists, see Let Users Select State and Country from Picklists in *Salesforce Help*.

Set Up Payment Gateways

Configure a payment gateway adapter for each merchant account. If you use multiple currencies or payment methods through a single payment processor, you can optionally set up multiple payment gateways for that processor. By default, the Order Management integration with Salesforce B2C Commerce only supports credit card and digital wallet payment types, but you can create your own customizations.

You can use Apex and the Connect REST API to set up payment gateways. See commercepayments namespace and Use Cases for the commercepayments Namespace in the *Apex Developer Guide*, and Commerce Payments Resources in the *Connect REST API Developer Guide*.

Payment Method Processing

The B2C Commerce integration processes each storefront payment method in the order data as follows:

1. Does the payment method ID in the order data match the following non-case-sensitive regular expression? If so, then create a DigitalWallet record in Salesforce.

   ```
   paypal|visa_checkout|pay_by_check|.*(apple|google|android|amazon|ali).*(pay)*
   ```

2. Does the payment method ID in the order data match the Gateway Provider Payment Method Type of a GtwyProvPaymentMethodType record in Salesforce? If so, then create a record in Salesforce according to the GtwyProvPaymentMethodType's Payment Method Type.
3. Does the card type in the order data match an entry in the Card Type picklist on the CardPaymentMethod object in Salesforce? If so, then create a CardPaymentMethod record in Salesforce.

   Note: You can't customize the Card Type picklist.

4. Return an error message that the payment method isn't supported.

To support a different payment type, set up a custom payment method as described in the next section.

Create Custom Payment Methods

If you use a storefront payment method that meets both of the following criteria, configure a custom payment method for it.

- The default regular expression doesn't identify it as a DigitalWallet method.
- It doesn't match any of the standard card types for a CardPaymentMethod.

To configure a custom payment method as an AlternativePaymentMethod, first create a RecordType for the AlternativePaymentMethod object. Give the record type a name that represents the payment method. For information on creating record types, see Create Record Types in *Salesforce Help*.

Define a custom payment method by creating a GtwyProvPaymentMethodType (Gateway Provider Payment Method Type) record. The gateway provider payment method type associates the custom payment method with a payment gateway and defines it as an AlternativePaymentMethod, CardPaymentMethod, or DigitalWallet.

Note: You can't create or access a **GtwyProvPaymentMethodType record** in the Salesforce UI. To create one, insert a record using a tool like Postman or Apex code. The API name of the object is GtwyProvPaymentMethodType, at the URL `/services/data/v`***`version`***`/sobjects/GtwyProvPaymentMethodType`.

Set the GtwyProvPaymentMethodType fields as described here.

Comments
Optional description.

Developer Name
The unique API name of the record.

Master Label
A human-readable name for the record.

Gateway Provider Payment Method Type
The name of the payment method in the storefront. This value must exactly match the payment method ID used in B2C Commerce.

Payment Gateway Provider Id
Reference to the Payment Gateway Provider record associated with the payment processor for the payment method.

Payment Method Type
`AlternativePaymentMethod` `CardPaymentMethod`, or `DigitalWallet`. To use AlternativePaymentMethod, first create a corresponding RecordType record.

Record Type Id
If using an AlternativePaymentMethod, this value is a reference to the RecordType record.

Here's an example of a gateway provider payment method definition:

```
{
  "DeveloperName" : "BankTransfer"
  "MasterLabel" : "Bank Transfer",
  "GtwyProviderPaymentMethodType" : "directBanking",
  "PaymentGatewayProviderId" : "0cJaa0000000001E67",
  "PaymentMethodType" : "AlternativePaymentMethod",
  "RecordTypeId" : "012aa000000008A34F"
}
```

Integrate Custom B2C Commerce Storefront Data

The integration can pass custom data on certain B2C Commerce objects to Order Management. To set up the transfer, add custom attributes to objects in B2C Commerce, and add matching custom fields to the corresponding objects in Order Management. The integration compares the B2C Commerce attribute ID and the Order Management field name at the API level. (It doesn't consider their UI labels.) If they match and the data types match, the integration includes those values when it creates Order Management records. If you also create a matching field on the corresponding summary object, Order Management includes the custom data when it creates a summary record.

You can add custom attributes to product options that exist as product line items, but not to product options that exist as attributes on parent products. You can't add an attribute to an attribute.

If you add a custom field to an object used in a flow, customize the flow to handle it.

Important: A change order is an Order record. If you add a required custom field to the Order object, update the service flows to set that field when they create a change order. Otherwise, the flow fails.

If order data includes a custom attribute value, but the corresponding Salesforce object has no matching custom field, the integration ignores that attribute. If a custom field exists in Salesforce, and order data doesn't include a matching custom attribute value, the integration ignores it.

Important: If a custom field in Salesforce is required, and it corresponds to a custom storefront attribute, then order data must include a value for that attribute. Without that value, the integration can't create the corresponding Salesforce record. In that case, the integration fails with an error.

Note: At the API level, custom field names in Salesforce always end in `__c`. Don't include it in the names of the matching attributes on your storefront objects. However, if you include a custom namespace in a custom field name, also include it in the name of the

matching storefront object attribute. For example, if the API name of your custom field in Salesforce is *mynamespace_FieldName__c*, name your matching storefront object attribute *mynamespace_FieldName*. Limit your custom field names to 40 characters (37 plus *__c*).

You can support product bundles by passing that information as custom data. To set up product bundle associations, define a custom guid attribute for the line item object in your storefront. In Salesforce, define a matching field on OrderItem and OrderItemSummary. When creating an order, assign the same value to the custom attribute on line items that belong together. You can define another custom attribute to identify a line item as part of a bundle. Then build logic into your fulfillment process to handle them accordingly.

Note: When using custom attributes to support bundles, customize your return and cancel flows to handle them according to your business rules. For example, design your cancel process so that when you cancel a bundle product, any associated bundle products are also canceled. Likewise, you can design your return process to disallow the return of only some of the products in a bundle.

You can match custom attributes and fields on these sets of objects. For High-Scale Order Integration (HSOI), custom fields are only necessary on the summary objects. HSOI directly maps to these fields. When using HSOI, don't add custom fields to standard objects. Setup is streamlined by keeping custom fields in a single location.

B2C Commerce	Salesforce Order Management
Product line item, gift certificate line item, and shipping line item	OrderItem and OrderItemSummary
Order	Order and OrderSummary
Order payment instrument and order payment transaction	AlternativePaymentMethod, CardPaymentMethod, DigitalWallet, Payment, PaymentAuthorization, and PaymentGatewayLog (see note)
Shipment	OrderDeliveryGroup and OrderDeliveryGroupSummary
Buyer Address on Order	Account (including Person Account) and Contact

Note: You can't add custom fields to the PaymentGatewayLog object. However, the integration recognizes certain custom attributes on `order/payments/payment` and copies them to standard fields on the PaymentGatewayLog record in Salesforce. To take advantage of this feature, create and populate custom attributes with the following exact names on the Order Payment Transaction object in B2C Commerce:

- `authCode` (value copied to PaymentGatewayLog.GatewayAuthCode)
- `avsResultCode` (value copied to PaymentGatewayLog.GatewayAvsCode)
- `approvalStatus` (value copied to PaymentGatewayLog.GatewayResultCode)

The integration supports the following data types for matching custom attributes and fields:

- Boolean
- Currency
- Datetime (Date isn't supported)
- Double
- Email
- Multipicklist
- Phone
- Picklist

- String
- TextArea
- URL

The Order Management B2C Service permission set provides the integration with access to Salesforce records. When you add a custom field to a Salesforce object, update that permission set by adding read and edit access to the new field. You also must add edit access to the permission set that provides access for your Order Management users.

1. From Setup, in the Quick Find box, enter `Permission Sets`, and then select **Permission Sets**.
2. Select **Order Management B2C Service**.
3. In the Apps section, click **Object Settings**.
4. Select the object that has the custom field.
5. Click **Edit**.
6. In the Field Permissions section, select the **Edit Access** checkbox for the custom field.
7. Click **Save**.
8. Return to the list of permission sets by selecting **Permission Sets** in the Setup navigation menu.
9. Select the permission set that controls access for your Order Management users. Normally, it's called **OM Console**.
10. In the Apps section, click **Object Settings**.
11. Select the object that has the custom field.
12. Click **Edit**.
13. In the Field Permissions section, select the **Edit Access** checkbox for the custom field.
14. Click **Save**.

Sync Data with Manual Data Map Refresh

Sync the order data map in Salesforce Order Management with your B2C Commerce org, for example after adding new custom attributes. Trigger the data map refresh to instantly access new schema changes instead of waiting for the next scheduled sync.

1. Click App Launcher, and then select **Administration** > **Global Preferences** > **Salesforce Order Management Configuration**.
2. Click **Refresh Data Map**.
3. Save your changes.

Account Object (Standard or PersonAccount)

Order Management always uses standard accounts to represent shoppers that have company names. If an order includes an `order/billing-address/company-name` value, and no matching account record exists, then the integration creates a standard account, regardless of the Person Accounts for Shoppers setting.

Changing the Person Accounts for Shoppers admin setting doesn't affect existing shopper data. However, it changes the way that the B2C Commerce integration stores new shopper data and whether it recognizes existing shopper data.

Note: If the B2C Integration Data Matching Rules setting is on, this behavior can vary depending on your org's matching rules.

- While Order Management is configured to use standard accounts: the integration recognizes existing shoppers that are stored as person accounts. It associates their new orders with their existing person accounts. If an existing shopper has records of both types, it associates new orders with their standard account and contact.

- While Order Management is configured to use person accounts: the integration doesn't recognize existing shoppers that are stored as standard accounts and contacts. If one of those shoppers places an order, the integration creates a person account record for them and associates the new order with it. Order Management treats the accounts as separate shoppers. If an existing shopper has records of both types, the integration associates new orders with their person account.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/billing-address/company-name or order/billing-address/first-name and order/billing-address/last-name	Name	If `order/billing-address/company-name` has a value in the order data packet, then it's copied to this value. If it has no value in the order data, this value is set as follows: • If this record is a person account, this value isn't set. • If this record isn't a person account, this value is set to *order/billing-address/first-name* + " " + *order/billing-address/last-name*.
order/billing-address/first-name and order/billing-address/second-name	FirstName	This value is only set for person accounts. It's set to *order/billing-address/first-name* + " " + *order/billing-address/second-name*.
order/billing-address/last-name	LastName	This value is only set for person accounts.
order/customer/customer-email	PersonEmail	This value is only set for person accounts.
order/billing-address/title	PersonTitle	This value is only set for person accounts.
order/billing-address/address1, order/billing-address/address2, and order/billing-address/address3	BillingStreet	This value is only set for person accounts. It's set to *order/billing-address/address1* + " " + *order/billing-address/address2* + " " + *order/billing-address/address3*.
order/billing-address/city	BillingCity	This value is only set for person accounts.
order/billing-address/postal-code	BillingPostalCode	This value is only set for person accounts.
order/billing-address/state-code	BillingState	This value is only set for person accounts. Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/state-code*.

B2C Commerce XSD Value	Salesforce Object Field	Notes
		It must be a standard 2-character ISO state or province code. • Picklists enabled—This value is set to the picklist integration value corresponding to the state code that matches *order/billing-address/state-code*.
order/billing-address/state-code	BillingStateCode	This value is only set for person accounts, and only when state and country/territory picklists are enabled on your org. *order/billing-address/state-code* must match the state code of an entry in the Salesforce state picklist.
order/billing-address/country-code	BillingCountry	This value is only set for person accounts. Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/country-code*. It must be a standard 2-character ISO country/territory code. • Picklists enabled—This value is set to the picklist integration value corresponding to the country/territory code that matches *order/billing-address/country-code*.
order/billing-address/country-code	BillingCountryCode	This value is only set for person accounts, and only when state and country/territory picklists are enabled on your org. *order/billing-address/country-code* must match the country/territory code of an entry in the Salesforce country/territory picklist.
order/billing-address/phone	Phone	This value is set for both person accounts and regular accounts.

AlternativePaymentMethod Object

The integration checks the `payment_method` value of each payment instrument in an order. If it matches the Gateway Provider Payment Method Type of a GtwyProvPaymentMethodType record, then the integration creates a record according to the GtwyProvPaymentMethodType's Payment Method Type value. If the Payment Method Type is AlternativePaymentMethod, then the integration creates an AlternativePaymentMethod record using the associated RecordType.

Note: To use AlternativePaymentMethod, first create a RecordType and GtwyProvPaymentMethodType for your custom payment type as described in the Create Custom Payment Methods section.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	RecordTypeId	This value is set to the ID of the RecordType record assigned to the GtwyProvPaymentMethodType.
order/payments/payment/credit-card/card-token	GatewayToken	
order/customer/customer-email	Email	
N/A	Status	This picklist value is always set to `Active`.
order/billing-address/company-name	CompanyName	
payment instrument payment_type or payment_method	Type	The value depends on whether you're using Salesforce Payments. • Salesforce Payments—This value is set to the payment instrument `payment_type`. • Other Payment Processor—This value is set to the payment instrument `payment_method`.
payment instrument payment_type or payment_method	PaymentMethodType	The value depends on whether you're using Salesforce Payments. If the payment instrument value isn't present, a PaymentMethodType isn't set.
payment instrument payment_bank	PaymentMethodSubType	The value depends on whether you're using Salesforce Payments. If the payment instrument value isn't present, a PaymentMethodSubType isn't set.
payment instrument payment_account_last_digits	PaymentMethodDetails	
order/billing-address/address1, order/billing-address/address2, and order/billing-address/address3	PaymentMethodStreet	This value is set to `order/billing-address/address1 + " " + order/billing-address/address2 + " " + order/billing-address/address3`.
order/billing-address/city	PaymentMethodCity	
order/billing-address/state-code	PaymentMethodState	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to

B2C Commerce XSD Value	Salesforce Object Field	Notes
		order/billing-address/state-code. It must be a standard 2-character ISO state or province code. • Picklists enabled—This value is set to the picklist integration value corresponding to the state code that matches *order/billing-address/state-code*.
order/billing-address/state-code	PaymentMethodStateCode	This value is only set when state and country/territory picklists are enabled on your org. *order/billing-address/state-code* must match the state code of an entry in the Salesforce state picklist.
order/billing-address/postal-code	PaymentMethodPostalCode	
order/billing-address/country-code	PaymentMethodCountry	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/country-code*. It must be a standard 2-character ISO country/territory code. • Picklists enabled—This value is set to the picklist integration value corresponding to the country/territory code that matches *order/billing-address/country-code*.
order/billing-address/country-code	PaymentMethodCountryCode	This value is only set when state and country/territory picklists are enabled on your org. *order/billing-address/country-code* must match the country/territory code of an entry in the Salesforce country/territory picklist.
order/billing-address/phone	Phone	
orderremoteHost	IpAddress	
N/A	AccountId	This value is set to the ID of the Account or Person Account record associated with the shopper.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	PaymentGatewayId	This value is set to the ID of the PaymentGateway record associated with the PaymentGatewayProvider record assigned to the GtwyProvPaymentMethodType.
N/A	ProcessingMode	This value is always set to `External`. It specifies that an external payment provider handles payment transactions.
order/payments/payment/*custom_attribute* or a custom attribute on the payment type	*custom_attribute_name*	If the Salesforce AlternativePaymentMethod object has a custom field matching a custom attribute on the storefront payment type or order payment transaction object, the value is copied to the AlternativePaymentMethod record. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

CardPaymentMethod Object

The integration checks the `payment_method` value of each payment instrument in an order against the following regex. If it matches, then the integration creates a DigitalWallet record for the instrument. If it doesn't match, then the integration checks the `payment/credit-card/card-type` value against the CardType picklist on the CardPaymentMethod object. If the picklist contains a match, then the integration creates a CardPaymentMethod record for the instrument. Otherwise, to support the payment method, you must set up a custom payment method for it.

```
paypal|visa_checkout|pay_by_check|.*(apple|google|android|amazon|ali).*(pay)*
```

Note: The regex isn't case-sensitive.

B2C Commerce XSD Value	Salesforce Object Field	Notes
payment instrument payment_brand or order/payments/payment/credit-card/card-type	CardType	The value depends on whether you're using Salesforce Payments. • Salesforce Payments—This value is set to the payment instrument *payment_brand*. • Other Payment Processor—This value is set to *order/payments/payment/credit-card/card-type* The value must match a card type in the CardType picklist on the CardPaymentMethod object. To handle a different card type, create a custom

B2C Commerce XSD Value	Salesforce Object Field	Notes
		payment method as described in the *Order Management Implementation Guide*.
order/payments/payment/credit-card/card-number	InputCardNumber	
order/payments/payment/credit-card/card-holder	CardHolderName	
order/payments/payment/credit-card/expiration-year	ExpiryYear	
order/payments/payment/credit-card/expiration-month	ExpiryMonth	
[order payment method]	CardCategory	This value can be `CreditCard` or `DebitCard`. The default value is `CreditCard`.
N/A	Status	This picklist value is always set to `Active`.
order/payments/payment/credit-card/card-token	GatewayToken	
payment instrument payment_brand or credit_card_type	PaymentMethodType	The value depends on whether you're using Salesforce Payments. If the payment instrument value isn't present, the value is set to `Other`. • Salesforce Payments—This value is set to the payment instrument `payment_brand`. • Other Payment Processor—This value is set to the payment instrument `credit_card_type`.
payment instrument payment_wallet_type or payment_method	PaymentMethodSubType	The value depends on whether you're using Salesforce Payments. • Salesforce Payments—This value is set to the payment instrument `payment_wallet_type`. • Other Payment Processor—This value is set to the payment instrument `payment_method`.
payment instrument payment_account_last_digits or masked_credit_card_number	PaymentMethodDetails	The value depends on whether you're using Salesforce Payments. • Salesforce Payments—This value is set to the payment instrument `payment_account_last_digits`. • Other Payment Processor—This value is set to the payment instrument `masked_credit_card_number`.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/billing-address/address1, order/billing-address/address2, and order/billing-address/address3	PaymentMethodStreet	This value is set to *order/billing-address/address1* + " " + *order/billing-address/address2* + " " + *order/billing-address/address3*.
order/billing-address/city	PaymentMethodCity	
order/billing-address/state-code	PaymentMethodState	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/state-code*. It must be a standard 2-character ISO state or province code. • Picklists enabled—This value is set to the picklist integration value corresponding to the state code that matches *order/billing-address/state-code*.
order/billing-address/state-code	PaymentMethodStateCode	This value is only set when state and country/territory picklists are enabled on your org. *order/billing-address/state-code* must match the state code of an entry in the Salesforce state picklist.
order/billing-address/postal-code	PaymentMethodPostalCode	
order/billing-address/country-code	PaymentMethodCountry	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/country-code*. It must be a standard 2-character ISO country/territory code. • Picklists enabled—This value is set to the picklist integration value corresponding to the country/territory code that matches *order/billing-address/country-code*.
order/billing-address/country-code	PaymentMethodCountryCode	This value is only set when state and country/territory picklists are enabled on

B2C Commerce XSD Value	Salesforce Object Field	Notes
		your org. *order/billing-address/country-code* must match the country/territory code of an entry in the Salesforce country/territory picklist.
N/A	AccountId	This value is set to the ID of the Account or Person Account record associated with the CardPaymentMethod.
N/A	PaymentGatewayId	This value is set to the ID of the PaymentGateway record whose ExternalReference value matches the processor ID value of the order payment instrument.
N/A	ProcessingMode	This value is always set to *External*. It specifies that an external payment provider handles payment transactions.
order/payments/payment/*custom_attribute* or a custom attribute on the payment type	*custom_attribute_name*	If the Salesforce CardPaymentMethod object has a custom field matching a custom attribute on the storefront payment type or order payment transaction object, the value is copied to the CardPaymentMethod record. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

Contact Object

When using person accounts, access shopper contact data using person account records, not contact records. When using standard accounts and contacts, the `BillToContactId` field on these objects points to the associated contact record:

- Credit Memo
- Fulfillment Order
- Invoice
- Order
- Order Summary

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	AccountId	This value is set to the ID of the Account record that represents the shopper.
order/customer/customer-email	Email	
order/billing-address/first-name	FirstName	

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/billing-address/last-name	LastName	
order/billing-address/phone	Phone	

DigitalWallet Object

The integration checks the `payment_method` value of each payment instrument in an order. If it matches the following regex, then the integration creates a DigitalWallet record for the instrument. If it doesn't match, then the integration tries to create an AlternativePaymentMethod or a CardPaymentMethod record for the instrument.

```
paypal|visa_checkout|pay_by_check|.(apple|google|android|amazon|ali).(pay)*
```

Note: The regex isn't case-sensitive.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	Status	This picklist value is always set to *`Active`*.
payment instrument payment_type or payment_method	Type	The value depends on whether you're using Salesforce Payments. • Salesforce Payments—This value is set to the payment instrument *`payment_type`*. • Other Payment Processor—This value is set to the payment instrument *`payment_method`*.
order/payments/payment/credit-card/card-token	GatewayToken	
payment instrument payer_email	Email	
payment instrument payment_type or payment_method	PaymentMethodType	The value depends on whether you're using Salesforce Payments. • Salesforce Payments—This value is set to the payment instrument *`payment_type`*. • Other Payment Processor—This value is set to the payment instrument *`payment_method`*.
payment instrument payment_type or payment_method	PaymentMethodSubType	The value depends on whether you're using Salesforce Payments. • Salesforce Payments—This value is set to the payment instrument *`payment_type`*.

B2C Commerce XSD Value	Salesforce Object Field	Notes
		• Other Payment Processor—This value is set to the payment instrument *payment_method*.
payment instrument payment_reference	PaymentMethodDetails	
order/billing-address/address1, order/billing-address/address2, and order/billing-address/address3	PaymentMethodStreet	This value is set to *order/billing-address/address1* + " " + *order/billing-address/address2* + " " + *order/billing-address/address3*.
order/billing-address/city	PaymentMethodCity	
order/billing-address/state-code	PaymentMethodState	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/state-code*. It must be a standard 2-character ISO state or province code. • Picklists enabled—This value is set to the picklist integration value corresponding to the state code that matches *order/billing-address/state-code*.
order/billing-address/state-code	PaymentMethodStateCode	This value is only set when state and country/territory picklists are enabled on your org. *order/billing-address/state-code* must match the state code of an entry in the Salesforce state picklist.
order/billing-address/postal-code	PaymentMethodPostalCode	
order/billing-address/country-code	PaymentMethodCountry	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/country-code*. It must be a standard 2-character ISO country/territory code. • Picklists enabled—This value is set to the picklist integration value

B2C Commerce XSD Value	Salesforce Object Field	Notes
		corresponding to the country/territory code that matches *order/billing-address/country-code*.
order/billing-address/country-code	PaymentMethodCountryCode	This value is only set when state and country/territory picklists are enabled on your org. *order/billing-address/country-code* must match the country/territory code of an entry in the Salesforce country/territory picklist.
N/A	AccountId	This value is set to the ID of the Account or Person Account record associated with the DigitalWallet.
N/A	PaymentGatewayId	This value is set to the ID of the PaymentGateway record whose ExternalReference value matches the processor ID value of the order payment instrument.
N/A	ProcessingMode	This value is always set to *External*. It specifies that an external payment provider handles payment transactions.
order/payments/payment/*custom_attribute* or a custom attribute on the payment type	*custom_attribute_name*	If the Salesforce DigitalWallet object has a custom field matching a custom attribute on the storefront payment type or order payment transaction object, the value is copied to the DigitalWallet record. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

Order Object

The integration uses both the `order/order-no` and `catalog/catalog-id` values to check for duplicate orders. It looks for an existing order record with an OrderReferenceNumber matching the `order-no` and a SalesChannelId pointing to a SalesChannel that matches the `catalog-id`. If it finds one, it doesn't create a duplicate order.

Note: If the catalog ID for a sales channel changes and an existing order is received with a different catalog ID, the integration considers it a new order and creates a duplicate record.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	Pricebook2Id	This value is set to the ID of the Pricebook2 record for the standard price book.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/customer/customer-name	Name	
N/A	Status	When the Order record is created, this value is set to `Draft`. The last step in the Composite API call that creates the records related to the order sets this value to `Active`.
N/A	EffectiveDate	This value is set to the current datetime when the record is created.
catalog/catalog-id and order/order-no	OrderManagementReferenceIdentifier	This value is set to *B2C realm ID* + "_" + *B2C instance ID* + "@" + *catalog/catalog-id* + "@" + *order/order-no*.
order/order-no	OrderReferenceNumber	
order/billing-address/address1, order/billing-address/address2, and order/billing-address/address3	BillingStreet	This value is set to *order/billing-address/address1* + " " + *order/billing-address/address2* + " " + *order/billing-address/address3*.
order/billing-address/city	BillingCity	
order/billing-address/state-code	BillingState	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/state-code*. It must be a standard 2-character ISO state or province code. • Picklists enabled—This value is set to the picklist integration value corresponding to the state code that matches *order/billing-address/state-code*.
order/billing-address/state-code	BillingStateCode	This value is only set when state and country/territory picklists are enabled on your org. *order/billing-address/state-code* must match the state code of an entry in the Salesforce state picklist.
order/billing-address/postal-code	BillingPostalCode	

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/billing-address/country-code	BillingCountry	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/billing-address/country-code*. It must be a standard 2-character ISO country/territory code. • Picklists enabled—This value is set to the picklist integration value corresponding to the country/territory code that matches *order/billing-address/country-code*.
order/billing-address/country-code	BillingCountryCode	This value is only set when state and country/territory picklists are enabled on your org. *order/billing-address/country-code* must match the country/territory code of an entry in the Salesforce country/territory picklist.
order/billing-address/phone	BillingPhoneNumber	
order/order-date	OrderedDate	
order/customer/customer-email	BillingEmailAddress	
N/A	BillToContactId	This value is set to the ID of the Contact record that represents the shopper. When using person accounts, this value isn't set. In that case, access shopper contact information via the Account instead of the Contact.
N/A	AccountId	This value is set to the ID of the Account or Person Account record that represents the shopper.
catalog/catalog-id	SalesChannelId	This value is set to the ID of the SalesChannel record whose SalesChannelName field matches the `catalog/catalog-id` in the order data packet.
order/taxation	TaxLocaleType	This value is set to *Net* or *Gross* based on the value of order/taxation. If using Net taxation, this value is set to *Net* and the integration creates OrderItemTaxLineItem records for the order. If using Gross taxation,

B2C Commerce XSD Value	Salesforce Object Field	Notes
		this value is set to `Gross` and the integration doesn't create OrderItemTaxLineItem records for the order. If the value isn't set, then the default value is `Net`.
order/currency	CurrencyIsoCode	This value is only set if the Salesforce org has Multicurrency enabled.
order/*`custom_attribute`*	*`custom_attribute_name`*	If the Salesforce Order object has a custom field matching a custom attribute on the storefront order object, the value is copied to the Order record. If the Salesforce OrderSummary object also has a matching custom field, it's copied to both records. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

OrderAdjustmentGroup Object

The integration creates one OrderAdjustmentGroup record for each promotion that applies to an entire order or to multiple items in an order. For an order-level promotion, it assigns the OrderAdjustmentGroup to one OrderItemAdjustmentLineItem record for each OrderItem record in the order. It creates the OrderAdjustmentGroup even if the order only contains one OrderItem. For an item-level promotion that applies to multiple OrderItems, it assigns the OrderAdjustmentGroup to one OrderItemAdjustmentLineItem record for each OrderItem record in the Order that the promotion applies to. If an item-level promotion only applies to one OrderItem, the integration doesn't create an OrderAdjustmentGroup for it.

Note: Shipping adjustments that aren't part of an order-level promotion are treated as item-level promotions. An example of an order-level promotion is "20% off and free shipping."

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/price-adjustments/price-adjustment/promotion-id	Name	
order/price-adjustments/price-adjustment/promotion-id	Description	
N/A	Type	This picklist value depends on the promotion level: • Order-level—This value is set to `Header`. • Item-level—This value is set to `SplitLine`.
N/A	OrderId	This value is set to the ID of the associated original Order record.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	AdjustmentCauseId	This value is set to the ID of the associated Promotion record.

OrderDeliveryGroup Object

The integration creates an OrderDeliveryGroup record for each shipment (also called a LineItemGroup in B2C Commerce) in the order data.

For order changes such as returns and cancellations that include prorated delivery amounts, the proration considers the OrderDeliveryGroups containing OrderItems that are part of the change. For example, consider an order that has three OrderDeliveryGroups, each with multiple OrderItems. A return of two OrderItems, each from a different OrderDeliveryGroup, uses the total delivery charges associated with those two OrderDeliveryGroups. It prorates that amount across the OrderItems in those OrderDeliveryGroups by OrderItem price. The shipping refund equals the prorated delivery amounts for the two returned OrderItems. The return doesn't consider any delivery charges associated with the third OrderDeliveryGroup.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/customer/customer-email	EmailAddress	
order/shipments/shipment/shipping-address/city	DeliverToCity	
order/shipments/shipment/shipping-address/country-code	DeliverToCountry	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/shipments/shipment/shipping-address/country-code*. It must be a standard 2-character ISO country/territory code. • Picklists enabled—This value is set to the picklist integration value corresponding to the country/territory code that matches *order/shipments/shipment/shipping-address/country-code*.
order/shipments/shipment/shipping-address/country-code	DeliverToCountryCode	This value is only set when state and country/territory picklists are enabled on your org. *order/shipments/shipment/shipping-address/country-code* must match the country/territory code of an entry in the Salesforce country/territory picklist.
order/shipments/shipment/shipping-address/title, order/shipments/shipment/shipping-address/first-name, order/shipments/shipment/shipping-address/last-name, and order/shipments/shipment/shipping-address/suffix	DeliverToName	If the DeliveryGroup contains gift certificate items, this value will contain the buyer_name: firstName + ' ' + lastName. Otherwise, the shipping address name is set to:

B2C Commerce XSD Value	Salesforce Object Field	Notes
		order/shipments/shipment/shipping-address/title + " " + *order/shipments/shipment/shipping-address/first-name* + " " + *order/shipments/shipment/shipping-address/last-name* + " " + *order/shipments/shipment/shipping-address/suffix*
order/shipments/shipment/shipping-address/postal-code	DeliverToPostalCode	
order/shipments/shipment/shipping-address/state-code	DeliverToState	Usage depends on whether state and country/territory picklists are enabled in Salesforce. • Picklists not enabled—This value is set to *order/shipments/shipment/shipping-address/state-code* It must be a standard 2-character ISO state or province code. • Picklists enabled—This value is set to the picklist integration value corresponding to the state code that matches *order/shipments/shipment/shipping-address/state-code*
order/shipments/shipment/shipping-address/state-code	DeliverToStateCode	This value is only set when state and country/territory picklists are enabled on your org. *order/shipments/shipment/shipping-address/state-code* must match the state code of an entry in the Salesforce country/territory picklist.
order/shipments/shipment/shipping-address/address1, order/shipments/shipment/shipping-address/address2, and order/shipments/shipment/shipping-address/address3	DeliverToStreet	This value is set to *order/shipments/shipment/shipping-address/address1* + " " + *order/shipments/shipment/shipping-address/address2* + " " + *order/shipments/shipment/shipping-address/address3*
order/shipments/shipment/shipping-address/phone	PhoneNumber	
order/shipments/shipment/gift	IsGift	This value is only populated if order/shipments/shipment/gift=*true*.
order/shipments/shipment/gift-message	GiftMessage	This value is only populated if `order/shipments/shipment/gift=`*true*.
order/shipments/shipment/shipping-method	OrderDeliveryMethodId	This value is set to the ID of the OrderDeliveryMethod record whose `ReferenceNumber` field matches the

B2C Commerce XSD Value	Salesforce Object Field	Notes
		order/shipments/shipment/shipping-method value in the order data.
N/A	OrderId	This value is set to the ID of the associated original Order record.
order/shipments/shipment/*custom_attribute*	*custom_attribute_name*	If the Salesforce OrderDeliveryGroup object has a custom field matching a custom attribute on the storefront shipment object, the value is copied to the OrderDeliveryGroup record. If the Salesforce OrderDeliveryGroupSummary object also has a matching custom field, it's copied to both records. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

OrderItem Object

If an order has a TaxLocaleType of Net, then the integration also creates an OrderItemTaxLineItem record for each OrderItem record.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/product-lineitems/product-lineitem/lineitem-text	Description	If the record represents a shipping charge, this value is set to *Shipping*. Otherwise, it's set to *order/product-lineitems/product-lineitem/lineitem-text*.
N/A	Type	If the record represents a shipping charge, this picklist value is set to *Delivery Charge*. Otherwise, it's set to *Order Product*.
order/product-lineitems/product-lineitem/quantity	Quantity	
order/product-lineitems/product-lineitem/net-price	TotalLineAmount	
N/A	LineNumber	Each order's OrderItems that represent physical products and non-shipping charges are assigned sequential LineNumber values starting at *1*. OrderItems that represent shipping charges are assigned sequential LineNumber values starting at *1000*, so in order detail views, they appear after all other OrderItems. Each delivery group has it's own line numbering, so there may be multiple

B2C Commerce XSD Value	Salesforce Object Field	Notes
		OrderItems with `LineNumber: 1` if there are multiple delivery groups. Order Management doesn't directly support kit items or line item options, so it stores them as OrderItem records with their own LineNumbers. An OrderItem's LineNumber is unrelated to its order line position in B2C Commerce.
order/product-lineitems/product-lineitem/base-price or order/product-lineitems/product-lineitem/net-price and order/product-lineitems/product-lineitem/quantity	UnitPrice	The integration calculates this value to a limited number of decimal places and includes it for reference. Because the calculation can introduce small rounding errors, Order Management doesn't use this value for any processing. The calculation formula depends on the tax locale type of the order: • Net Taxation—`base-price` • Gross Taxation—`net-price` / `quantity`
order/product-lineitems/product-lineitem/base-price or order/product-lineitems/product-lineitem/gross-price and order/product-lineitems/product-lineitem/quantity	GrossUnitPrice	The integration calculates this value to a limited number of decimal places and includes it for reference. Because the calculation can introduce small rounding errors, Order Management doesn't use this value for any processing. The calculation formula depends on the tax locale type of the order: • Net Taxation—`gross-price` / `quantity` • Gross Taxation—`base-price`
order/product-lineitems/product-lineitem/base-price	ListPrice	This value is only set if the Optional Price Books feature is enabled. It's needed when the OrderItem has no associated PriceBookEntry.
N/A	OrderId	This value is set to the ID of the associated original Order record.
N/A	OrderDeliveryGroupId	This value is set to the ID of the associated OrderDeliveryGroup record.
N/A	PricebookEntryId	This value is set to the ID of the associated PriceBookEntry record. If the Optional Price Books feature is enabled, this value isn't set.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	Product2Id	This value is set to the ID of the associated Product2 record.
order/product-lineitems/product-lineitem/*custom_attribute*, order/gift-certificate-lineitems/gift-certificate-lineitem/*custom_attribute*, or order/shipping-lineitems/shipping-lineitem/*custom_attribute*	*custom_attribute_name*	If the Salesforce OrderItem object has a custom field matching a custom attribute on the storefront product-lineitem object, giftcertificate-lineitem object, or shipping-lineitem object, the value is copied to the OrderItem record. If the Salesforce OrderItemSummary object also has a matching custom field, it's copied to both records. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

OrderItemAdjustmentLineItem Object

An OrderItemAdjustmentLineItem record represents the application of a promotion to an OrderItem record. Thus, an OrderItem record has one associated OrderItemAdjustmentLineItem record for each promotion that applies to it.

If an order has a TaxLocaleType of Net, then the integration also creates an OrderItemTaxLineItem record for each OrderItemAdjustmentLineItem record.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/product-lineitems/product-lineitem/product-id and order/price-adjustments/price-adjustment/lineitem-text	Name	This value is set to *order/product-lineitems/product-lineitem/product-id* + "–" + *order/price-adjustments/price-adjustment/lineitem-text*
price-adjustments/price-adjustment/net-price	Amount	
price-adjustments/price-adjustment/tax	TotalTaxAmount	
order/price-adjustments/price-adjustment/lineitem-text	PromotionText	
N/A	OrderItemId	This value is set to the ID of the OrderItem record that the adjustment applies to.
N/A	OrderAdjustmentGroupId	For an order-level promotion, or an item-level promotion that applies to multiple OrderItems, this value is set to the ID of the parent OrderAdjustmentGroup record. For an item-level promotion that applies to only one OrderItem, it isn't set.
N/A	AdjustmentCauseId	This value is set to the ID of the associated Promotion record.

OrderItemTaxLineItem Object

If an order has a TaxLocaleType of Net, then the integration creates an OrderItemTaxLineItem record for each OrderItem and OrderItemAdjustmentLineItem record in the order. If an order has a TaxLocaleType of Gross, then the integration doesn't create any OrderItemTaxLineItem records for it.

B2C Commerce XSD Value	Salesforce Object Field	Notes
See notes	Name	This value is generated based on the `Type` of the associated OrderItem or OrderItemAdjustmentLineItem record and, if applicable, the `StockKeepingUnit` of the associated Product2: • OrderItem: – Order Product—*`StockKeepingUnit`* + " - *`Tax`*" – Delivery Charge—*`"Delivery Charge - Tax"`* • OrderItemAdjustmentLineItem: – Product—*`StockKeepingUnit`* + " - *`Adjustment Tax`*" – Shipping—"*`Delivery Charge - Adjustment Tax`*"
N/A	Type	This picklist value is always set to *`Estimated`*.
order/product-lineitems/product-lineitem/tax	Amount	
order/product-lineitems/product-lineitem/tax-rate	Rate	
order/order-date	TaxEffectiveDate	
N/A	OrderItemId	If the tax applies to an OrderItem record, this value is set to the ID of that order item. If it applies to an OrderItemAdjustmentLineItem record, this value is set to the ID of the order item that the adjustment applies to.
N/A	OrderItemAdjustmentLineItemId	This value is only set for tax that applies to an OrderItemAdjustmentLineItem record.

Payment Object

If an `order/payments/payment/transaction-type` value in the order data is *`sale`* or *`capture`*, then the integration creates a Payment record for that transaction. If the value starts with *`auth`*, then it creates a PaymentAuthorization record. These checks aren't case-sensitive.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	PaymentGatewayId	This value is set to the ID of the PaymentGateway record whose ExternalReference value matches the processor ID value of the order payment instrument associated with the order payment transaction.
N/A	PaymentGatewayLogId	This value is set to the ID of the PaymentGatewayLog record associated with the payment.
order/payments/payment/amount	Amount	
N/A	ProcessingMode	This picklist value is always set to *`External`*.
N/A	Status	This picklist value is always set to *`Processed`*.
order/payments/payment/transaction-id	GatewayRefNumber	This value isn't required in the order data. The default value is *`null`*.
order/customer/customer-email	Email	
order/order-date	Date	
order/payments/payment/transaction-type	Type	
order/billing-address/phone	Phone	
N/A	PaymentGroupId	This value is set to the ID of the PaymentGroup record associated with the payment.
N/A	AccountId	This value is set to the ID of the Account or Person Account record associated with the shopper.
N/A	PaymentMethodId	This value is set to the ID of the CardPaymentMethod or DigitalWallet record associated with the payment.
order/currency	CurrencyIsoCode	This value is only set if the Salesforce org has Multicurrency enabled.
order/payments/payment/*`custom_attribute`* or a custom attribute on the payment type	*`custom_attribute_name`*	If the Salesforce Payment object has a custom field matching a custom attribute

B2C Commerce XSD Value	Salesforce Object Field	Notes
		on the storefront payment type or order payment transaction object, the value is copied to the Payment record. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

PaymentAuthorization Object

If an `order/payments/payment/transaction-type` value in the order data starts with *`auth`*, then the integration creates a PaymentAuthorization record for that transaction. If the value is *`sale`* or *`capture`*, then it creates a Payment record. These checks aren't case-sensitive.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	PaymentGatewayId	This value is set to the ID of the PaymentGateway record whose ExternalReference value matches the processor ID value of the order payment instrument associated with the order payment transaction.
N/A	PaymentGatewayLogId	This value is set to the ID of the PaymentGatewayLog record associated with the payment authorization.
order/payments/payment/amount	Amount	
N/A	ProcessingMode	This picklist value is always set to *`External`*.
N/A	Status	This picklist value is always set to *`Processed`*.
order/payments/payment/transaction-id	GatewayRefNumber	This value isn't required in the order data. The default value is *`null`*.
(custom attribute) order/payments/payment/authCode	GatewayAuthCode	The default Payment Transaction object in B2C Commerce doesn't include an `authCode` attribute. To use it, add it as a custom attribute and populate it before sending the order data. If you name it `authCode`, the integration automatically copies it to the GatewayAuthCode field.
N/A	PaymentGroupId	This value is set to the ID of the PaymentGroup record associated with the payment authorization.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	AccountId	This value is set to the ID of the Account or Person Account record associated with the shopper.
N/A	PaymentMethodId	This value is set to the ID of the CardPaymentMethod or DigitalWallet record associated with the payment authorization.
order/currency	CurrencyIsoCode	This value is only set if the Salesforce org has Multicurrency enabled.
order/payments/payment/*custom_attribute* or a custom attribute on the payment type	*custom_attribute_name*	If the Salesforce PaymentAuthorization object has a custom field matching a custom attribute on the storefront payment type or order payment transaction object, the value is copied to the PaymentAuthorization record. If a custom field is non-nillable, then order data must include a value for the corresponding custom attribute. If the value is missing, the integration fails.

PaymentGatewayLog Object

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	PaymentGatewayId	This value is set to the ID of the PaymentGateway record whose ExternalReference value matches the processor ID value of the order payment instrument associated with the order payment transaction.
N/A	InteractionStatus	This picklist value is always set to *Success*.
order/payments/payment/transaction-type	InteractionType	The default value is *Authorization*.
(custom attribute) order/payments/payment/authCode	GatewayAuthCode	The default Payment Transaction object in B2C Commerce doesn't include an `authCode` attribute. To use it, add it as a custom attribute and populate it before sending the order data. If you name it `authCode`, the integration automatically copies it to the GatewayAuthCode field.
(custom attribute) order/payments/payment/avsResultCode	GatewayAvsCode	The default Payment Transaction object in B2C Commerce doesn't include an `avsResultCode` attribute. To use it, add

B2C Commerce XSD Value	Salesforce Object Field	Notes
		it as a custom attribute and populate it before sending the order data. If you name it `avsResultCode`, the integration automatically copies it to the GatewayAvsCode field.
(custom attribute) order/payments/payment/approvalStatus	GatewayResultCode	The default Payment Transaction object in B2C Commerce doesn't include an `approvalStatus` field. To use it, add it as a custom attribute and populate it before sending the order data. If you name it `approvalStatus`, the integration automatically copies it to the GatewayResultCode field.

PaymentGroup Object

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	SourceObjectId	This value is set to the ID of the original Order record that the payments in the group apply to.
order/currency	CurrencyIsoCode	This value is only set if the Salesforce org has Multicurrency enabled.

PendingOrderSummary Object

PendingOrderSummary is only used by High Scale Orders. It holds order data that's been ingested into ZOS and is waiting for its Salesforce records to be created. You can trigger creation of the records, including the Summary records, by importing a PendingOrderSummary from the Pending Order Summaries page in the Salesforce UI.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	AccountId	This value is set to the ID of the Account or Person Account record that represents the shopper.
N/A	BillToContactId	This value is set to the ID of the Contact record that represents the shopper. When using person accounts, this value isn't set. In that case, access shopper contact information via the Account instead of the Contact.
order/customer/customer-email	BillingEmailAddress	

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/billing-address/phone	BillingPhoneNumber	
order/currency	CurrencyIsoCode	This value is only set if the Salesforce org has Multicurrency enabled.
catalog/catalog-id and order/order-no	ExternalReferenceIdentifier	Used internally to prevent duplicate records. This value is case-sensitive. On creation, this value is set to *B2C realm ID* + "_" + *B2C instance ID* + "@" + *catalog/catalog-ID* + "@" + *order/order-no*.
all price, adjustment, and tax values	GrandTotalAmount	Total amount, including adjustments and tax, of the order.
order/order-no	OrderNumber	
order/order-date	OrderedDate	
all data	Payload	This field contains the entire order data payload.
N/A	PayloadType	The datatype of the Payload. Possible values are *JSON_GZIP* or *JSON_RAW*.
catalog/catalog-id	SalesChannelId	This value is set to the ID of the SalesChannel record whose SalesChannelName field matches the `catalog/catalog-id` in the order data packet.
N/A	SalesStoreId	This value is set to the ID of the RetailStore or WebStore record associated with the order.
order/billing-address/first-name and order/billing-address/last-name	ShopperName	This value is set to *order/billing-address/first-name* + " " + *order/billing-address/last-name*.

PricebookEntry Object

Salesforce Order Management doesn't use the PricebookEntry object. It creates a PriceBookEntry record for each Product2 because the OrderItem object requires a `PricebookEntryId` value.

Note: If you turn on the Optional Price Books feature, then Order Management doesn't create price book entries for product records that it creates from order data. If you turn off the feature, then you must add price book entries manually.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	Pricebook2Id	This value is set to the ID of the Pricebook2 record for the standard price book.
N/A	Product2Id	This value is set to the ID of the associated Product2 record.
order/product-lineitems/product-lineitem/base-price	UnitPrice	This value is only set when the PriceBookEntry record is created. It isn't updated when an OrderItem associated with the same Product2 is created. It isn't used for any calculations in Order Management.
order/currency	CurrencyIsoCode	This value is only set if the Salesforce org has Multicurrency enabled.

Product2 Object

The integration identifies products by first searching for a Product2 with a ProductCode value that matches the `product-id` value in the order data. If it doesn't find one, it looks for a Product2 with a StockKeepingUnit value that matches the `product-id` value. If it still doesn't find one, then it creates a Product2 record.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/product-lineitem/lineitem-text	Description	
order/product-lineitem/product-name	Name	
order/product-lineitem/product-id	StockKeepingUnit	If the product is an electronic gift certificate, this value is set to `eGiftCertificate`.
order/product-lineitem/product-id	ProductCode	If the product is an electronic gift certificate, this value is set to `eGiftCertificate`.
N/A	IsActive	The default value is `true`.

Promotion Object

For information on the Promotion object, see the Loyalty Management Developer Guide.

B2C Commerce XSD Value	Salesforce Object Field	Notes
order/price-adjustments/price-adjustment/promotion-id	Name	
order/price-adjustments/price-adjustment/lineitem-text	DisplayName	
order/price-adjustments/price-adjustment/lineitem-text	Description	
N/A	StartDate	The default value is the current date.
N/A	IsCommercePromotion	The default value is `true`.

B2C Commerce XSD Value	Salesforce Object Field	Notes
N/A	IsActive	The default value is `true`.

SalesChannel Object

The integration associates B2C Commerce site IDs with Salesforce SalesChannels.

B2C Commerce XSD Value	Salesforce Object Field	Notes
site/site-id	Description	
site/site-id	SalesChannelName	
N/A	Type	

WebStore Object

The integration looks for a WebStore matching the currency and locale of the order, and creates one if needed. It associates one WebStore with each combination of currency and locale that appears in order data. For example, let's say that you have one site that supports US orders in US dollars. A second site supports UK orders in pounds or Euros, and also supports French orders in Euros. Create four Web Stores to represent them:

- USD - en_US
- GBP - en_GB
- EUR - en_GB
- EUR - fr_FR

B2C Commerce XSD Value	Salesforce Object Field	Notes
catalog/catalog-id and order/currency	Name	For a single-currency org, this value is the order `catalog/catalog-id`, for example `SiteGenesis`. For a multicurrency org, it's the order `catalog/catalog-id` plus the currency associated with the WebStore, for example `SiteGenesis USD`.
order/currency	CurrencyIsoCode	This value is only set if the Salesforce org has Multicurrency enabled.
order/customer-locale	DefaultLanguage	The order locale ID must be a valid value for a WebStore DefaultLanguage. It can't be `default`.
order/taxation	DefaultTaxLocaleType	This value is set to `Net` or `Gross` based on the value of `order/taxation`. If using Net taxation, this value is set to `Net`. If using Gross taxation, this value is set to

B2C Commerce XSD Value	Salesforce Object Field	Notes
		`Gross`. If the value isn't set, then the default value is `Net`.
order/order-no and catalog/catalog-id	ExternalReference	This value is set to `B2C realm ID` + "_" + `B2C instance ID` + "@" + `catalog/catalog-id`.
N/A	LocationId	The system creates a location and sets this value to its ID. It sets the new location's LocationType field to `Virtual` and its ExternalReference field to `null`. If you include any custom validations for the Location object, they must account for these values.
order/currency	SupportedCurrencies	This value is only set if the Salesforce org has Multicurrency enabled. It matches the value of CurrencyIsoCode.
order/customer-locale	SupportedLanguages	The order locale ID must be a valid value for a WebStore DefaultLanguage. It can't be `default`. It matches the value of DefaultLanguage.
N/A	Type	This value is always set to `B2CE`.

Importing Order Data

Salesforce Order Management handles order data imported from your storefront. You can bulk load historical order data from a legacy system into a new Order Management org or implement a custom import process for new orders from your storefront.

Importing Data into Order Management

In Order Management, each order is represented by an OrderSummary record and a set of supporting records, such as OrderItemSummaries and OrderDeliveryGroupSummaries. When you import order data, you create the Order record and supporting records, including OrderItems and OrderDeliveryGroups. Then process the order with the Create Order Summary flow core action or API to create the OrderSummary and its supporting records.

Creating Order Summaries for Imported Orders

You can create order summaries for imported orders in a few ways.

Importing Custom Order Data

To import custom order data, add a custom field to Order, OrderItem, or OrderDeliveryGroup and a matching custom field to the corresponding summary object. When you set the custom value on the base record, the Create Order Summary action or API copies it to the matching field on the corresponding summary record.

Importing Data When Using Salesforce B2C Commerce

If you import data that's used by the B2C Commerce integration, such as product or shopper data, keep it synchronized with your B2C Commerce data. Otherwise, the integration can't recognize it.

High-Scale Orders and Deduplication

High-Scale Orders uses the Pending Order Summaries REST API to create summaries from order summary graphs. Entities included in the request parameter aren't subject to deduplication rules.

Importing Data into Order Management

In Order Management, each order is represented by an OrderSummary record and a set of supporting records, such as OrderItemSummaries and OrderDeliveryGroupSummaries. When you import order data, you create the Order record and supporting records, including OrderItems and OrderDeliveryGroups. Then process the order with the Create Order Summary flow core action or API to create the OrderSummary and its supporting records.

Note: Don't directly create OrderSummary records. Always run the flow core action or call the API.

To import large amounts of order data, use Bulk API 2.0. See the Bulk API 2.0 Developer Guide.

If you're importing small amounts of order data, you can make composite REST API requests. A composite API call can include up to 200 records. That limit applies to all objects, not only the orders. For example, an order with 5 items and an order-level adjustment that applies to each of them includes at least 11 records: 1 order, 5 order items, and 5 order item adjustments. For more information, see Composite Resources in the *REST API Developer Guide*.

When making API calls to import data, keep in mind that they count toward your API limits. You can request a temporary increase for importing historical orders by opening a case with Salesforce Customer Support. Allow at least 2 weeks for Support to arrange the increase.

Managed Orders and Unmanaged Orders

The OrderSummary field OrderLifecycleType specifies whether the order is managed with Order Management features. The default value is Managed. You can configure the Create Order Summary action or API request to set the OrderLifecycleType field to `Managed` or `Unmanaged` when it creates a record, but you can't change it later.

Examples of orders that use the unmanaged lifecycle type include historical orders and orders managed by an external system.

For information about the differences between managed and unmanaged order lifecycles, see Order Lifecycle Management.

Data Requirements for Object Management Import

When importing storefront data, the order in which you create records is important because of the relationships between objects. For example, an OrderItem record requires a Product2, a PriceBookEntry, and an Order. So you can't create an OrderItem record until those three related records exist.

For each imported order, create records in this order.

1. Shopper data
   - Account
   - Contact (not required if the account record type is Person Account)
2. SalesChannel
3. Product2
4. PriceBookEntry (unless using Optional Price Books)
5. Payment data (depending on the type of payment)
   - AlternativePaymentMethod

- CardPaymentMethod
- DigitalWallet
- GtwyProvPaymentMethodType
- PaymentGatewayProvider
- PaymentGateway

6. OrderDeliveryMethod (optional)
7. Order

   The records created before the Order record aren't specific to one order. When creating them, check for duplicates. For information on configuring automatic identification and handling duplicate records, see Manage Duplicate Records.

8. OrderItem (at least one)
9. OrderItemAdjustmentLineItem (optional)
10. OrderItemTaxLineItem (optional)
11. OrderDeliveryGroup (at least one)
12. PaymentGroup (optional)
13. PaymentAuthorization (optional)
14. Payment (optional)

In addition to the normally required fields, Order Management requires values for other fields for some objects. When creating a record for the objects in this table, define values for all the listed fields.

Object	Required Fields
Account	• AccountNumber • BillingCity • BillingCountry • BillingCountryCode • BillingState • BillingStreet • FirstName (if a Person Account) • IsPersonAccount • LastName (if a Person Account) • Name (if not a Person Account)
SalesChannel	• SalesChannelName
Product2	• Description • IsActive • Name • ProductCode • StockKeepingUnit

Object	Required Fields
CardPaymentMethod	• AccountId • CardCategory • CardHolderName • CardType • ExpiryMonth • ExpiryYear • InputCardNumber • ProcessingMode • Status
Order	• AccountId • BillingCity • BillingCountry • BillingCountryCode • BillingEmailAddress • BillingState • BillingStreet • EffectiveDate • Name • OrderedDate • OrderReferenceNumber • SalesChannelId • Status (Set this field to **Draft** for a new record, and change it to **Active** before creating the OrderSummary.)
OrderItem	• Description • GrossUnitPrice • OrderDeliveryGroupId • OrderId • PricebookEntryId • Product2id • Quantity • TotalLineAmount • Type • UnitPrice
OrderItemAdjustmentLineItem	• Amount • Description • Name

Object	Required Fields
	• OrderItemId
OrderItemTaxLineItem	• Amount • Description • Name • OrderItemAdjustmentLineItemId (if applicable) • OrderItemId • Rate • TaxEffectiveDate • Type
OrderDeliveryGroup	• DeliverToCity • DeliverToCountry • DeliverToName • DeliverToPostalCode • DeliverToState • DeliverToStreet • EmailAddress • OrderDeliveryMethodId • OrderId
PaymentGroup	• SourceObjectId
PaymentAuthorization	• AccountId • Amount • PaymentGatewayId • PaymentGroupId • PaymentMethodId • ProcessingMode • Status
Payment	• AccountId • Amount • Date • Email • PaymentGatewayId • PaymentGroupId • PaymentMethodId • Phone

Object	Required Fields
	• ProcessingMode • Status • Type

Creating Order Summaries for Imported Orders

You can create order summaries for imported orders in a few ways.

For details about each method, see the corresponding developer guides. To learn about using different types of flows, see the Salesforce Flow Developer Center.

Method	Pros	Cons
Flow triggered by a platform event	• Simple to implement • Asynchronous transaction separate from order creation improves performance	• No parallel execution; orders processed one at a time • No control of timing • Asynchronous execution requires complex error handling
Flow triggered by a record change (Order Status: Draft Activated)	• Simple to implement • Allows a multithreaded approach for creating order summaries in parallel with orders • A Composite API call can create an order and order summary in the same transaction, allowing simpler error handling; for example: 1. Create the Order record and set its Status to **Draft**. 2. Create the supporting records, such as order items and order delivery groups. 3. Change the order's Status to **Activated**. That triggers the Create Order Summary flow to run on that order.	• When the order and order summary are created in the same transaction, and the order summary fails, the order also fails • Large transactions • Lowest performance method
Apex triggered by a platform event	• Asynchronous transaction separate from order creation improves performance • Allows a multithreaded approach for creating order summaries in parallel with orders • Best performance method	• No control of timing • Asynchronous execution requires complex error handling

Method	Pros	Cons
Scheduled Apex job	• Allows controlled batch execution	• Subject to Apex limits on the number of records processed • Requires programmatic identification of the orders to process • Runs even when no orders need processing
Direct calls to API resources	• Allows external control • Some clients allow parallel processing	• Subject to API call limits • Requires external app design, authentication, and so on

Importing Custom Order Data

To import custom order data, add a custom field to Order, OrderItem, or OrderDeliveryGroup and a matching custom field to the corresponding summary object. When you set the custom value on the base record, the Create Order Summary action or API copies it to the matching field on the corresponding summary record.

Importing Custom Order Data

For example, your storefront data includes an order value called Shopper Category. You can add a custom field called `ShopperCategory__c` to both Order and OrderSummary. When you import an order, set the `ShopperCategory__c` value on the Order record. When you run the Create Order Summary action or call the API for that order, it copies that value to the `ShopperCategory__c` field on the OrderSummary record.

Order summary creation supports custom fields for these data types:

- Boolean
- Currency
- Datetime (Date isn't supported)
- Double
- Email
- Multipicklist
- Phone
- Picklist
- Reference
- String
- TextArea
- URL

Importing Data When Using Salesforce B2C Commerce

If you import data that's used by the B2C Commerce integration, such as product or shopper data, keep it synchronized with your B2C Commerce data. Otherwise, the integration can't recognize it.

For example, when the integration creates an OrderItem record, it searches for a Product2 record whose ProductCode or StockKeepingUnit matches the product ID in the order data. If you import catalog product data and the ProductCode and StockKeepingUnit values don't match the product ID values in B2C Commerce, the integration can't recognize those products. In that case, the integration can add the wrong products to orders or create duplicate Product2 records.

This table lists some of the data fields that the integration uses to look up records. If you import or create these objects, keep them synchronized with your B2C Commerce data.

Order Management Object Field	B2C Commerce Data
OrderDeliveryMethod.ReferenceID	Shipping Method ID
Product2.ProductCode or Product2.StockKeepingUnit	Product ID
PaymentGateway.ReferenceID	Payment Processor ID
SalesChannel.SalesChannelName	Site ID or Domain ID
• For a person account: Account.PersonEmail • For other accounts: Contact.Email field	Customer Email Address
Account.Name	• For an individual shopper not using a person account: Customer Billing Address First Name and Last Name concatenated • For a business shopper: Customer Billing Address Company Name

The OrderSummary ExternalReferenceIdentifier Field

Starting in API version 55.0, the B2C Commerce integration uses the OrderSummary.ExternalReferenceIdentifier field to prevent duplicate orders. When an order is ingested from B2C Commerce, this field is set to *B2C realm ID* + "_" + *B2C instance ID* + "@" + *B2C Commerce catalog/domain ID* + "@" + *B2C Commerce order number*. If you create other orders in Salesforce, give them unique ExternalReferenceIdentifier values.

Note: If you use High Scale Orders, the PendingOrderSummary.ExternalReferenceIdentifier field is also set.

Note: In API version 55.0, the standard B2C Commerce integration set this value to "SFDC" + "@" + *nanotime* + "@" + *UUID* and High Scale Orders set it to the value used in later versions.

High-Scale Orders and Deduplication

High-Scale Orders uses the Pending Order Summaries REST API to create summaries from order summary graphs. Entities included in the request parameter aren't subject to deduplication rules.

If a request has entity details in the PurchaseSupportDetails section, the Pending Order Summaries REST API uses deduplication logic to determine whether to update an existing record or create a record. Duplicate entities within the request that aren't in the database won't be deduplicated.

Sending orders with inaccurate or outdated data can cause problems that are hard to fix later. Before sending orders to Salesforce Order Management, upload new information for Account, Contact, Product2, PricebookEntry, Promotion, WebStore, SalesChannel, and OrderDeliveryMethod.

This table lists the Pending Order Summaries REST API entities and their corresponding deduplication logic.

Entity	Deduplication Logic
Account, Person Account, Contact	• When you create Account, Person Account, and Contact Duplicate Rules and Matching Rules, the API calls these rules to find duplicates. For Contact, an AccountId is required to use duplication rules. • For Person Account, Person Accounts must be enabled.
Pricebook2	Only Standard Price Book lookup is supported. If a Pricebook2 node is included in the request, the Standard Price Book ID is queried and used.
Product2	Deduplication is based on ProductCode. If an existing Product2 with the same ProductCode is found, it's used.
PricebookEntry	• For a single currency Salesforce org, deduplication is based on Pricebook2Id and Product2Id. • For a multicurrency Salesforce org, deduplication is based on Pricebook2Id, Product2Id, and CurrencyIsoCode. If you don't specify the CurrencyIsoCode, the default CurrencyIsoCode is used.
SalesChannel	Deduplication is based on the SalesChannelName field.
Promotion	Deduplication is based on the Name field.
WebStore	Deduplication is based on the ExternalReference field.
OrderDeliveryMethod	Deduplication is based on the ReferenceNumber field.

SEE ALSO:

Pending Order Summaries

High-Scale Orders

Fulfillment Orders

A FulfillmentOrder record has a set of supporting records, including FulfillmentOrderLineItems, FulfillmentOrderItemAdjustments, and FulfillmentOrderItemTaxes. When you call an Order Management API or the flow core action to create a fulfillment order, it also creates the supporting records.

For managed OrderSummaries, you create a fulfillment order via Apex, Connect API, or flow core action. This method also creates the supporting records, such as fulfillment order line items, and then sets the new fulfillment order's Status to Allocated.

You can also create a fulfillment order externally using a platform API, such as sObject, Composite, or Bulk API 2.0. When you create a fulfillment order for an unmanaged OrderSummary, you can only use a platform API.

When you create a fulfillment order via a platform API, you also create the supporting records. In that case, when you create the fulfillment order, set its Status to Draft. After creating all the supporting records, change its Status to Allocated (or a custom status associated with the Activated Status Category).

In addition to the normally required fields, Order Management requires values for certain fields for each FulfillmentOrder record and supporting record. When creating a record of an object in this table, define values for all the listed fields.

Object	Required Fields
FulfillmentOrder	• AccountId • DeliveryMethodId • FulfilledFromLocationId • FulfilledToCity • FulfilledToCountry • FulfilledToEmailAddress • FulfilledToLatitude • FulfilledToLongitude • FulfilledToName • FulfilledToPhone • FulfilledToPostalCode • FulfilledToState • FulfilledToStreet • OrderId • OrderSummaryId • Status • Type
FulfillmentOrderLineItem	• Description • FulfillmentOrderId • GrossUnitPrice • OrderItemId • OrderItemSummaryId • OriginalQuantity • Product2Id • Quantity • TotalLineAmount • Type • TypeCode • UnitPrice
FulfillmentOrderItemAdjustment	• Amount • Description • FulfillmentOrderId • FulfillmentOrderLineItemId • OrderItemAdjustLineSummaryId
FulfillmentOrderItemTax	• Amount • Description

Object	Required Fields
	• FulfillmentOrderId • FulfillmentOrderItemAdjustId (if applicable) • FulfillmentOrderLineItemId • OrderItemTaxLineItemSummaryId • Rate • TaxEffectiveDate • Type

Location Capacity

You can limit the number of fulfillment orders assigned to a location by defining its fulfillment order capacity. The Location object has three fields for fulfillment order capacity.

Track Fulfillment Order Capacity
Indicates whether the location's capacity is tracked. The default is false.

Fulfillment Order Capacity
The maximum number of fulfillment orders that can be assigned to the location per a user-defined time period. If the value is null, the location's capacity is unlimited.

Assigned Fulfillment Order Count
The number of fulfillment orders currently assigned to the location. To define the time period associated with the maximum capacity, reset this value at a specified interval. For example, to track capacity per day, run an automated job that sets this value to 0 for each location at midnight of the location's time zone.

You can manage location capacity using these Connect API resources and flow core actions.

Hold Fulfillment Order Capacity
When you want to assign a fulfillment order to a location, hold capacity for it at that location. If the location has no capacity, it returns an error. Free capacity is defined as the location's fulfillment order capacity minus the sum of its held capacity and assigned fulfillment orders.

Confirm Held Fulfillment Order Capacity
When you assign a fulfillment order to a location, confirm the location's held capacity. Confirming held capacity increases the location's assigned fulfillment order count and decreases its held capacity.

Release Fulfillment Order Capacity
If you're holding capacity for a fulfillment order at a location and decide not to assign that fulfillment order to that location, release the held capacity. Releasing capacity decreases the location's held capacity without increasing its assigned fulfillment order count.

Get Fulfillment Order Capacity Values
Returns a location's maximum capacity, assigned fulfillment order count, and capacity being held. This function is the only way to retrieve the held quantity, because it isn't represented by a field on the Location object.

Changing a location's `Assigned Fulfillment Order Count` doesn't affect its held capacity. For example, resetting the assigned count at midnight doesn't remove held capacity for unassigned fulfillment orders. To decrease held capacity, you must confirm or release it.

SEE ALSO:

Location Object Reference Topic

Taxation in Order Management

Tax calculations can be complex, especially when supporting multiple currencies. Orders can have different tax types, and taxes can be affected by price adjustments.

Tax Types
Orders can include two types of taxes: sales tax and value-added tax (VAT).

Price Adjustments and Taxes
The OrderItemTaxLineItem object represents a tax applied to an order item or the change in tax associated with an order item adjustment.

SEE ALSO:

Salesforce Standard Objects Reference

Salesforce Order Management Implementation Guide

Tax Types

Orders can include two types of taxes: sales tax and value-added tax (VAT).

Sales tax is paid directly on the sale price of goods and services in countries such as the United States. The tax amount is displayed separately from the product prices. This display type is also referred to as "net."

VAT is an indirect tax on goods and services. The displayed product price includes the tax. If tax fields appear, they usually display "included" instead of the amount. This display type is also referred to as "gross."

To support both types, Order Management objects include fields for price amounts, tax amounts, combined amounts, and whether an order uses net or gross taxation. You configure page layouts and Salesforce Flow screens to show the appropriate fields for your business needs.

For example, to display net taxation in an order summary page layout, include the `TotalAmount` and `TotalTaxAmount` fields. To display gross taxation, include the `GrandTotalAmount` field. An order summary record always includes all three values, and the `GrandTotalAmount` always equals the sum of the `TotalAmount` and the `TotalTaxAmount`.

Default page layouts display net tax amounts. If your org supports only locales that use sales tax, you can use the default layouts. If your org supports a locale that uses VAT, configure your page layouts to display fields with gross tax amounts.

An org that supports multiple currencies can show net tax amounts for some orders and gross tax amounts for others. In that case, create a user profile for service agents in gross tax locales by cloning the standard user profile. Create page layouts that display gross tax amounts instead of net tax amounts, and assign them to the new profile.

For details about price and tax fields, see the *Object Reference for Salesforce and Lightning Platform*.

For information about how the B2C Commerce integration handles taxes, see the Salesforce B2C Commerce Storefront Order Data Map.

SEE ALSO:

Manage Multiple Currencies Help Topic

Taxation in Order Management

Price Adjustments and Taxes

The OrderItemTaxLineItem object represents a tax applied to an order item or the change in tax associated with an order item adjustment.

For example, an order item with a `TotalPrice` of 100 is subject to a 5% sales tax. It has an associated order item tax line item with an `Amount` of 5. Applying a 10% discount creates an order item adjustment line item with an `Amount` of -10 and an order item tax line item with an `Amount` of -0.50. The adjustment order item tax line item includes references to both the order item adjustment line item and the original order item tax line item that it adjusts.

When you apply an adjustment to an order item that has multiple order item tax line items, all tax adjustments are combined in one change order item tax line item. For example, if an order item has two taxes, with Amount *`1`* and Amount *`0.5`*, then applying a 10% discount creates one order item tax line item with Amount *`-0.15`*.

The B2C Commerce integration combines the taxes on each order item into a single amount. To differentiate between multiple taxes on an order item, use custom attributes to represent them.

Here's an example of how applying a discount affects an order item's tax values. Consider an order item with a `TotalPrice` of 100, subject to a 10% sales tax. The records initially look like this.

- OrderItem
  - `TotalPrice` 100
- OrderItemTaxLineItem
  - `Amount` 10
  - `OrderItemId` pointing to the OrderItem
- OrderItemSummary
  - `TotalPrice` 100
  - `TotalTaxAmount` 10
  - `TotalAmtWithTax` 110
  - `OriginalOrderItemId` pointing to the OrderItem
- OrderItemTaxLineItemSummary
  - `Amount` 10
  - `OrderItemSummaryId` pointing to the OrderItem
  - `OriginalOrderItemTaxLineItemId` pointing to the OrderItemTaxLineItem

Then apply a 10% discount to the OrderItem by calling the Adjust Order Item Summaries Submit action and passing these values.

- `orderItemSummaryId`—The ID of the OrderItemSummary
- `adjustmentType`—Percentage
- `discountValue`—-10

These are the new records.

- OrderItemAdjustmentLineItem
  - `Amount` *`-10`*
  - `TotalTaxAmount` *`-1`*
  - `TotalAmtWithTax` *`-11`*
  - `OrderItemId` pointing to the OrderItem
- OrderItemAdjustmentLineSummary

  - `Amount` -10
  - `TotalTaxAmount` -1
  - `TotalAmtWithTax` -11
  - `OrderItemSummaryId` pointing to the OrderItemSummary
  - `OriginalOrderItemAdjustmentLineItemId` pointing to the OrderItemAdjustmentLineItem
- OrderItemTaxLineItem (for the adjustment)
  - `Amount` -1
  - `OrderItemAdjustmentLineItemId` pointing to the OrderItemAdjustmentLineItem
  - `RelatedTaxLineItemId` pointing to the original OrderItemTaxLineItem
- OrderItemTaxLineItemSummary (for the adjustment)
  - `Amount` -1
  - `OrderItemAdjustmentLineSummaryId` pointing to the OrderItemAdjustmentLineSummary
  - `OrderItemSummaryId` pointing to the OrderItemSummary
  - `OriginalOrderItemTaxLineItemId` pointing to the new OrderItemTaxLineItem for the adjustment

The values on the existing records now look like this.

- OrderItem (unchanged)
  - `TotalPrice` 100
- OrderItemTaxLineItem (unchanged)
  - `Amount` 10
  - `OrderItemId` pointing to the OrderItem
- OrderItemSummary
  - `TotalPrice` 90
  - `TotalTaxAmount` 9
  - `TotalAmtWithTax` 99
  - `OriginalOrderItemId` pointing to the OrderItem
  - `TotalAdjustmentAmount` -10
  - `TotalAdjustmentTaxAmount` -1
  - `TotalAdjustmentAmtWithTax` -11
- OrderItemTaxLineItemSummary
  - `Amount` 9 (sum of both OrderItemTaxLineItems)
  - `OrderItemSummaryId` pointing to the OrderItem
  - `OriginalOrderItemTaxLineItemId` pointing to the original OrderItemTaxLineItem

SEE ALSO:

Taxation in Order Management

API Framework for Exchanges with RMA Returns

Expand your exchange capability by increasing the ways that a customer service representative (CSR) can help your customers exchange products. With these exchange capabilities, a CSR can exchange uneven products and can exchange products in an already-fulfilled order. To configure an exchange workflow in your org, you need the SalesforceOrderManagementGrowth license.

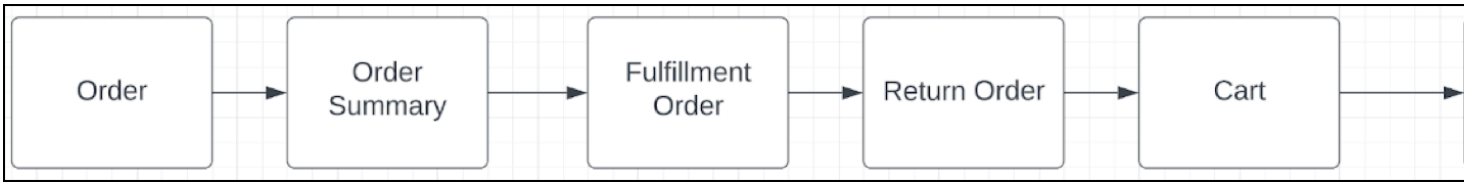
Before a CSR can place uneven or post-fulfillment exchanges, your organization needs to implement an exchange flow. In the Spring '24 release and later, you can configure exchanges using the exchanges invocable actions.

EDITIONS

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

There are two Connect APIs for exchanges, Preview Cart to Exchange Order and Submit Cart to Exchange Order. There's an invocable action for both of these APIs as well. There are also two objects for exchanges, OrderPaymentSummary and OrderPaymentSummaryReference.

A fully implemented exchange flow starts by getting the order summary to gather information for the exchange. The items being returned are selected from the return order. From there, a cart is created with the new items. The exchange order is run to exchange the items in the return order for the items in the cart. Finally, a summary shows whether any additional funds are needed, if a refund is necessary, or if it's an even exchange.



Exchanges supports both EnsureFunds and EnsureRefunds. After implementing exchanges, set `isReservedBalanceAmountConsidered` to `true` for any existing flow that uses EnsureFunds or EnsureRefunds.

EnsureFunds can be called for any OrderSummary regardless of the SourceProcess (Standard, OrderOnBehalfOf, Exchange). The EnsureFunds and ApplyFunds logic now consider shared OrderPaymentSummaries. Both the EnsureFunds and ApplyFunds contracts now contain an optional boolean field, isReservedBalanceAmountConsidered. This field indicates whether the EnsureFunds logic considers the ReservedBalanceAmount for the shared OrderPaymentSummary.

The EnsureRefunds contracts now contain an optional boolean field isReservedBalanceAmountConsidered. This field indicates whether the EnsureRefunds logic considers the ReservedBalanceAmount for the OrderPaymentSummary. EnsureRefunds considers any reserved balance amount that a child exchange order has in reserve to fully fund the exchange Order Summary. The reserved balance amount is available in the payment record and only the extra balance is refunded as part of the EnsureRefunds command.

If you're using the EnsureFunds or EnsureRefunds flows, update the flows to contain the isReservedBalanceAmountConsidered flag.

For additional or excess funds, the formulas that calculate the funds remain the same. All the captures, refunds, pending refunds, and returns are related to the OrderPaymentSummaries in the OrderSummary Relationship tree.

The new algorithm is executed in this order.

1. Get all of the Order Summary IDs that are related to the Order Summary.

   Note: Get the root OrderSummary from the OrderSummaryRelationship and then find all the AssociatedOrderSummaries with that root.

2. Get all the Order Payment Summary entities.
3. Get all OrderSummary totals.

The algorithm then uses the existing formula to calculate the excess funds based on the roll-up fields.

1. Preview Cart for an Exchange Order
   Before submitting an exchange request, we recommend previewing the balance state for the exchange order. Previewing can help you determine what functions are necessary, especially for uneven exchanges.
2. Submit Cart for an Exchange Order
   After you've previewed the cart, it's almost time to submit the exchange. There are a few considerations to keep in mind while creating the submit for the exchange.

SEE ALSO:

Salesforce Help: Exchanges with RMA Returns Overview

Ensure Funds

Ensure Refunds

Preview Cart for an Exchange Order

Before submitting an exchange request, we recommend previewing the balance state for the exchange order. Previewing can help you determine what functions are necessary, especially for uneven exchanges.

EDITIONS

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

There are three outcomes of an uneven exchange.

- Additional funds are needed because the web cart's total amount is larger than the return order amount being returned: `WebCart.GrandTotalAmount() > ReturnOrder.GrandTotalAmount()`
- A refund is needed because where the total amount being refunded is greater than the web cart total: `WebCart.GrandTotalAmount() < ReturnOrder.GrandTotalAmount()`
- The exchange is an even exchange, where the totals equal each other: `WebCart.GrandTotalAmount() = ReturnOrder.GrandTotalAmount()`

To preview the cart, set the `orderSummaryId` (`mainOrderSummaryId`), `exchangeCartId`, and `referenceId` (`returnOrderId`) to be neither null nor empty. They must have valid Salesforce IDs and must represent a valid entity.

The `ReferenceId` must be type `ReturnOrder`. The return order `StatusCategory` must be activated. `ReturnOrderId` (`ReferenceId`) must belong to `MainOrderSummaryId`.

The minimum cart must contain at least one product and delivery group. Make sure that the cart belongs to the same account as the `OrderSummary` `OrderSummary.Account` equals `cart.Account`.

The preview response might look like this example.

```
{
  "changeBalances": {
  "grandTotalAmount": -47.49,
  "totalAdjDeliveryAmtWithTax": 12.94
  "totalAdjDistAmountWithTax": 0,
  "totalAdjProductAmtWithTax": -60.44,
  "totalAdjustedDeliveryAmount": 11.99,
  "totalAdjustedDeliveryTaxAmount": 0.95,
  "totalAdjustedProductAmount": -55.96,
  "totalAdjustedProductTaxAmount": -4.48,
  "totalAdjustmentDistributedAmount": 0,
  "totalAdjustmentDistributedTaxAmount": 0,
  "totalAmount": -43.97,
```

```
"totalExcessFundsAmount": 47.49,
"totalFeeAmount": 0,
"totalFeeTaxAmount": 0,
"totalRefundableAmount": 47.49,
"totalRequiredFundsAmount": 0,
"totalTaxAmount": -3.52
},
"errors": [],
"orderSummaryId": "1OsSB000000UdWz0AK",
"success": true
}
```


SEE ALSO:

API Framework for Exchanges with RMA Returns

Salesforce Help: Exchanges with RMA Returns Overview

Submit Cart for an Exchange Order

After you've previewed the cart, it's almost time to submit the exchange. There are a few considerations to keep in mind while creating the submit for the exchange.

EDITIONS

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

Before creating your submit order, validate that `orderSummaryId` (`MainOrderSummaryId`), `exchangeCartId`, and `referenceId`:

- Aren't null or empty
- Are valid Salesforce IDs
- Represent a valid entity

The `referenceId` must be type `ReturnOrder`. Make sure the `ReturnOrder's StatusCategory` is Activated or Pending.

Make sure the `ReturnOrderId` belongs to the `MainOrderSummaryId`.

The cart for an exchange needs at least one product and at least one delivery group. The cart must belong to the same account as the `OrderSummary`. Make sure the `OrderSummary.Account` equals the `cart.Account`.

Input Fields for the Submit Cart to Exchange Order API Representation

Name	Type	Description	Required or Optional
`mainOrderSummaryId`	String	ID of the main Order Summary.	Required
`exchangeCartId`	String	ID of the cart used for adding items to the exchange order.	Required
`referenceId`	String	Polymorphic Reference ID, which must be related to the main Order Summary. Only Return Order ID is supported.	Required

Name	Type	Description	Required or Optional
`paymentInfoList`	PaymentInfoInputRepresentationList	List of payment information if additional funds are needed for the new exchange order. • PaymentInfoInputRepresentation – String paymentAuthorizationId – Array[String] `paymentIds` – String paymentMethodId(optional) – String lastPaymentGatewayLogId(optional) – String `name`(optional)	Optional
`paymentInfoItems`	Array[PaymentInfoInputRepresentation]	List of payment information if additional funds are needed for the new exchange order.	Optional
SequenceOrderPaymentSummaryInputList	Sequence	Ordered list of `OrderPaymentSummaries` and the reserved balance amounts to apply them to. • [illegible] sequences • String `orderPaymentSummaryId` (required) • Number amount, which uses the total Order Payment Summary balance if it's zero or not specified. • Order Number: If the exchange is created in the order number specified, it creates an exchange with random guide as the order number	Optional

Example

The submit response might look like this example.

```
{
  "changeBalances": {
```

```
"grandTotalAmount": -47.49,
"totalAdjDeliveryAmtWithTax": 12.94
"totalAdjDistAmountWithTax": 0,
"totalAdjProductAmtWithTax": -60.44,
"totalAdjustedDeliveryAmount": 11.99,
"totalAdjustedDeliveryTaxAmount": 0.95,
"totalAdjustedProductAmount": -55.96,
"totalAdjustedProductTaxAmount": -4.48,
"totalAdjustmentDistributedAmount": 0,
"totalAdjustmentDistributedTaxAmount": 0,
"totalAmount": -43.97,
"totalExcessFundsAmount": 47.49,
"totalFeeAmount": 0,
"totalFeeTaxAmount": 0,
"totalRefundableAmount": 47.49,
"totalRequiredFundsAmount": 0,
"totalTaxAmount": -3.52
},
"errors": [],
"exchangeOrderSummaryId": "1OsSB000000UdYb",
"orderSummaryId": "1OsSB000000UdWz0AK",
"success": true
}
```


SEE ALSO:

API Framework for Exchanges with RMA Returns

Salesforce Help: Exchanges with RMA Returns Overview

Payment Sequencing

When ensuring funds or refunds for orders that include multiple payment methods, you can control the sequence in which to apply the amounts to payment methods. For example, if an order is partially paid with a gift card, you can refund the gift card first, regardless of how the payment amounts were originally distributed. Or, if the order is modified between when different fulfillment groups are fulfilled, you can capture the partial payment amount from the gift card first if a future order fulfillment is canceled and preserve the full gift card amount. The default logic is based on matching amounts to OrderPaymentSummary amounts.

Default Payment Sequence to Ensure Funds

When you call Ensure Funds without specifying a sequence of OrderPaymentSummaries, it distributes the funds between the Order PaymentSummaries belonging to the OrderSummary according to this logic. However, if multiple order payment summaries have equal BalanceAmount values, their order of selection is random.

- Examine the invoice and ensure it doesn't exceed the total `BalanceAmount` of all the order payment summaries associated with the order summary. If the BalanceAmount is equal to the invoice balance, apply the funds from the order payment summary to the invoice.
  - If no exact match is found, apply funds from the order payment summary with the largest BalanceAmount to the invoice.
- If the invoice still has a balance, repeat step 1 until no captured funds remain.

  - If the invoice still has a balance and there's an order payment summary with an authorized amount equal to the remaining invoice balance, then capture and apply the funds from that order payment summary to the rest of the invoice balance.
  - If no exact match is found, capture and apply funds from the order payment summary with the largest authorized amount.
- If the invoice still has a balance, repeat step 2 until the full invoice balance is ensured.

Default Payment Sequence to Ensure Refunds

When you call Ensure Refunds without specifying a sequence of OrderPaymentSummaries, it distributes the refund between the OrderPaymentSummaries belonging to the OrderSummary according to this logic. However, if multiple OrderPaymentSummaries have equal amounts, their selection order is random.

- If a credit memo is specified, identify OrderPaymentSummaries with captured amounts that were applied to the corresponding invoice.
  - Examine the OrderPaymentSummaries. If one has a captured amount matching the credit memo amount, apply the refund to that payment.
  - If no exact match is found, look for OrderPaymentSummaries with captured amounts greater than the credit memo amount. If any exist, apply the refund to the smallest.
  - If no greater amounts are found, traverse the OrderPaymentSummaries in order of captured amount, from largest to smallest. Apply the refund to them until it's fully applied.
- If an excess funds amount is specified, identify OrderPaymentSummaries with captured amounts that weren't applied to any invoice.
  - Examine the OrderPaymentSummaries. If one has a captured amount matching the excess funds amount, apply the refund to that payment.
  - If no exact match is found, look for OrderPaymentSummaries with captured amounts greater than the excess funds amount. If any exist, apply the refund to the smallest.
  - If no greater amounts are found, traverse the OrderPaymentSummaries in order of captured amount, from largest to smallest. Apply the refund to them until it's fully applied.

Specify the Sequence of OrderPaymentSummaries

When you call Ensure Funds or Ensure Refunds, you can sequence the distribution of the amounts by including the `isAllowPartial` and `sequences` values.

isAllowPartial
: Controls what happens when the distributed amounts that you specify don't cover the full amount. If the value is true, the remaining amount is skipped. If it belongs to a credit memo, it remains on the credit memo. If the value is false, the default logic is applied to the remaining amount.

sequences
: An ordered list of paired amounts and OrderPaymentSummaries. Each amount is funded or refunded to the OrderPaymentSummary paired with it. The process traverses this list in order and stops when it has funded or refunded the full amount.

Custom Default Sequence Apex Example to Ensure Funds

You can implement your own default payment sequence by wrapping Ensure Funds in code that generates the isAllowPartial and sequences values. Here's an example by using Apex that tries to fund from DigitalWallet payment methods before any other payment type.

- Collect all OrderPaymentSummaries associated with the OrderSummary. Identify the payment type by using the record ID's key prefix.
- Fund $10 from each OrderPaymentSummary that's a DigitalWallet until the full amount is fulfilled.
- If an amount remains, fund it according to the default logic.

  You can customize which payment type to fund from first by changing the amount associated with each type's key prefix. Because the list is ordered, you can, for example, fund $10 from each DigitalWallet, then $10 from each CardPaymentMethod, and then apply the default logic to any remaining amount.

```
public class CreateSequenceOPSEnsureFundsInvocable {

    @InvocableMethod(label='Ensure Funds with Sequence OrderPaymentSummaries')
    public static void createSequenceOrderPaymentSummaryList(List<String> invoiceIds) {

        // DigitalWallet Key Prefix: 1DW
        // CardPaymentMethod Key Prefix: 03O
        // AlternativePaymentMethod Key Prefix: 8Z7
        Map<String, Double> sequencePrefixAndAmounts= new Map<String, Double>();
        sequencePrefixAndAmounts.put('1DW', 10.0);
        sequencePrefixAndAmounts.put('03O', 0.0);
        sequencePrefixAndAmounts.put('8Z7', 0.0);

        // Set is allowed partial to true/false
        Boolean isAllowPartial = true;

        Map<String, Double> sequenceOPSAndAmounts = new Map<String, Double>();

        ConnectApi.EnsureFundsAsyncInputRepresentation ensureFundsInput = new
ConnectApi.EnsureFundsAsyncInputRepresentation();
        ensureFundsInput.invoiceId = invoiceIds.get(0);
        ensureFundsInput.sequences = new
List<ConnectApi.SequenceOrderPaymentSummaryInputRepresentation>();
        ensureFundsInput.isAllowPartial = isAllowPartial;

        // Get the Id and ReferenceEntityID (OrderSummaryID) From the Invoice
        Invoice inv = [SELECT Id, ReferenceEntity.Id FROM Invoice WHERE Id = :invoiceIds
Limit 1];
        string orderSummaryId = inv.ReferenceEntityId;

        // Get list of OrderPaymentSummaries from the orderSummaryID
        List<OrderPaymentSummary> orderPaymentSummaries = [SELECT Id, PaymentMethod.Id,
Type FROM OrderPaymentSummary WHERE OrderSummary.Id = :orderSummaryId];

        Map<Id, OrderPaymentSummary> orderPaymentSummariesMap = new Map<Id,
OrderPaymentSummary>(orderPaymentSummaries);

        //Loop through the sequence of payment methods prefixes
        for(String keyPrefix: sequencePrefixAndAmounts.keySet()){
        // Loop through orderPaymentSummaries creating a map of sequence
            for(OrderPaymentSummary ops: orderPaymentSummaries){
                string paymentMethodId = ops.PaymentMethod.Id;
                if(paymentMethodId.startsWith(keyPrefix)){
                    sequenceOPSAndAmounts.put(ops.Id,
sequencePrefixAndAmounts.get(keyPrefix));
```

```
                    orderPaymentSummariesMap.remove(ops.Id);
                }
            }
        }

        // Create the list of sequences to add to EnsureFundsAsync input
        for(String orderPaymentSummarySequence: sequenceOPSAndAmounts.keySet()){
            ConnectApi.SequenceOrderPaymentSummaryInputRepresentation request = new
ConnectApi.SequenceOrderPaymentSummaryInputRepresentation();
            request.amount = sequenceOPSAndAmounts.get(orderPaymentSummarySequence);
            request.orderPaymentSummaryId = orderPaymentSummarySequence;
            ensureFundsInput.sequences.add(request);
        }

        ConnectApi.EnsureFundsAsyncOutputRepresentation result =
ConnectApi.OrderSummary.ensureFundsAsync(orderSummaryId, ensureFundsInput);

    }

}
```

Custom Default Sequence Apex Example to Ensure Refunds

You can implement your own default payment sequence by wrapping Ensure Refunds in code that generates the `isAllowPartial` and `sequences` values. Here's an example using Apex that tries to refund to DigitalWallet payment methods before any other payment type.

- Collect all OrderPaymentSummaries associated with the OrderSummary. Identify the payment type using the record ID's key prefix.
- Refund $10 to each OrderPaymentSummary that's a DigitalWallet until the full amount has been refunded.
- If an amount remains, refund it according to the default logic.

You can customize which payment type to refund first by changing the amount associated with each type's key prefix. Because the list is ordered, you could, for example, refund $10 to each DigitalWallet, then $10 to each CardPaymentMethod, and then apply the default logic to any remaining amount.

```
public class CreateSequenceOPSRefundsInvocable {

    @InvocableMethod(label='EnsureRefunds with Sequenced OrderPaymentSummaries')
    public static void createSequenceOrderPaymentSummaryList(List<String> creditMemoIds)
{

        // When applying refunds, prefer DigitalWallets
        // DigitalWallet            Key Prefix: 1DW
        // CardPaymentMethod        Key Prefix: 03O
        // AlternativePaymentMethod Key Prefix: 8Z7
        Map<String, Double> sequencePrefixAndAmounts= new Map<String, Double>();
        sequencePrefixAndAmounts.put('1DW', 10.0);
        sequencePrefixAndAmounts.put('03O', 0.0);
        sequencePrefixAndAmounts.put('8Z7', 0.0);

       // Always disallow partial refunds -- if the sequence doesn't cover the full refund,
 then apply the default logic to the remaining amount
        Boolean isAllowPartial = false;
```

```
        Map<String, Double> sequenceOPSAndAmounts = new Map<String, Double>();

        ConnectApi.EnsureRefundsAsyncInputRepresentation ensureRefundsInput = new
ConnectApi.EnsureRefundsAsyncInputRepresentation();
        ensureRefundsInput.creditMemoId = creditMemoIds.get(0);
        ensureRefundsInput.sequences = new
List<ConnectApi.SequenceOrderPaymentSummaryInputRepresentation>();
        ensureRefundsInput.isAllowPartial = isAllowPartial;

        // Get the Id and ReferenceEntityID (OrderSummaryID) From the Credit Memo
        CreditMemo cm = [SELECT Id, ReferenceEntity.Id  FROM CreditMemo  WHERE Id =
:creditMemoIds Limit 1];
        string orderSummaryId = cm.ReferenceEntityId;

        // Get list of OrderPaymentSummaries from the orderSummaryID
        List<OrderPaymentSummary> orderPaymentSummaries = [SELECT Id, PaymentMethod.Id,
Type  FROM OrderPaymentSummary  WHERE OrderSummary.Id = :orderSummaryId];

        Map<Id, OrderPaymentSummary> orderPaymentSummariesMap = new Map<Id,
OrderPaymentSummary>(orderPaymentSummaries);

        //Loop through the sequence of payment methods prefixes
        for(String keyPrefix: sequencePrefixAndAmounts.keySet()){
            // Loop through orderPaymentSummaries creating a map of sequence
            for(OrderPaymentSummary ops: orderPaymentSummaries){
                string paymentMethodId = ops.PaymentMethod.Id;
                if(paymentMethodId.startsWith(keyPrefix)){
                    sequenceOPSAndAmounts.put(ops.Id,
sequencePrefixAndAmounts.get(keyPrefix));
                    orderPaymentSummariesMap.remove(ops.Id);
                }
            }
        }

        // Create the sequence list and add it to the EnsureRefundsAsync input
        for(String orderPaymentSummarySequence: sequenceOPSAndAmounts.keySet()){
                ConnectApi.SequenceOrderPaymentSummaryInputRepresentation request = new
ConnectApi.SequenceOrderPaymentSummaryInputRepresentation();
                request.amount = sequenceOPSAndAmounts.get(orderPaymentSummarySequence);
                request.orderPaymentSummaryId = orderPaymentSummarySequence;
                ensureRefundsInput.sequences.add(request);
        }

        ConnectApi.EnsureRefundsAsyncOutputRepresentation result =
ConnectApi.OrderSummary.ensureRefundsAsync(orderSummaryId, ensureRefundsInput);

    }

}
```

Ensure Funds Payment Sequencing Edge Cases

The rollout of Payment Sequencing for Ensure Funds in Spring '26 introduced a few unexpected customer behaviors, but these behaviors are expected by design. Review these scenarios in case you encounter them when using Ensure Funds.

- **Zero amount and unspecified amounts provided for order payment summary result in same behavior**. Providing zero is treated the same as not specifying an amount. If you provide zero as an amount for any Order Payment Summary in a sequence, Ensure Funds interprets this as if no amount is specified. Thus, the system attempts to take as many available funds as possible instead of strictly applying zero funds.
- **Reserved balance amount doesn't respect sequencing or amount limits.** This is expected behavior as the reserved balance amount is exclusively tied to the order payment summary's order summary and is applied in full before other sources. Other order summaries can't use this reserved portion of funds and this reserved portion is consumed first, regardless of your provided sequence or amount limits.
- **Sequence not strictly reflected when isAllowPartial = false.** This is expected behavior. If a sequence is provided and isAllowPartial is set to false, the system first tries to apply funds according to the sequence. If the funds are insufficient, it falls back to the default strategy. The system ensures the correct final funded amount. The sequence on the invoice page may not exactly match the input sequence.
- **Amount parameter limits pre-captured and available to capture funds.** This is expected behavior. The amount parameter is treated as a total cap across both pre-captured and available to capture funds for the order payment summary.

Salesforce Order Management Lightning Components

Salesforce Order Management includes standard Lightning components.

Display Recipient's Order Product Details

Use the Order Product Summaries by Recipient component to display order product details on an Order Summary record page. The component is available in Salesforce Order Management.

The order product summary fields are defined by the Order Product Summaries related list on the OrderDeliveryGroup object page layout. To modify the columns on the order summary details page, edit the related list on the Order Delivery Group page layout, not the Order Summary page layout.

You can create a custom filter to control which order delivery group summary records are displayed.

Display Order Monetary Totals

Use the Order Summary Totals component to display financial totals on an Order Summary record page. The component is available in Salesforce Order Management.

You can customize the panel title and which values to display.

Order on Behalf Of for External Payments

To support payment flows with non-tokenized payments, configure the External Payment Mode flow. This flow bypasses the tokenized payment flow and calls Payment Authorize directly.

Before creating a flow for external payments, make sure Order on Behalf Of is properly configured. See Configure Order on Behalf Of.

To start creating the flow for external payments, download the External Payments Package.

Set the flow global variable **externalPaymentMode** to true.

Add a screen to the Order on Behalf Of UI flow to manage payments. We provide one you can use, or you can create your own page. If you're creating your own screen, ensure that it lets your representatives capture payment details and billing addresses.

After you've added this screen to your flow, call the **PaymentAuthorize** invocable from the outputs of that screen. Construct the input for the PaymentAuthorize in the flow.

Finally, update the billing details in the webcart using the flow Update Records action. Input BillingDetails from the new payment screen and input the PaymentMethod and PaymentGroup from the PaymentAuthorize Invocable.

Expand Data Sources for Return Insights

Extensions are a mechanism for using Apex to customize the functionality that powers B2B and B2C Commerce storefronts. The custom Apex class that implements an extension is called an extension provider. To extend or override the Product Expand API functionality, use the Commerce Order Management Product Expand Extension.

To start, extend the base class, commerce_ordermanagement.ProductExpandService.

```
public class ProductExpandServiceSample extends commerce_ordermanagement.ProductExpandService
 {
    // Extension code.
}
```

Override the `returnReasons` method of `commerce_ordermanagement.ProductExpandService`, which is used to get the product's return reasons. In this example, the entire method is replaced with custom logic to extract product data from ProductExpandRequest, set the return reasons, and process the ProductExpandResponse. You can also fetch the return reasons from an external service.

```
// Extract product data from the request with custom logic and get the product's return
reasons from an external service.

public override commerce_ordermanagement.ProductExpandResponse
returnReasons(commerce_ordermanagement.ProductExpandRequest productExpandRequest) {

        // Create a product expand response without the default processing.

        List<commerce_ordermanagement.ProductData> productList = new
List<commerce_ordermanagement.ProductData>();

        List<commerce_ordermanagement.Reason> returnReasons = new
List<commerce_ordermanagement.Reason>();

        commerce_ordermanagement.Reason reason1 = new commerce_ordermanagement.Reason();
        commerce_ordermanagement.Reason reason2 = new commerce_ordermanagement.Reason();

        //Set the reason in the response. You can also fetch the reasons from external
systems and set it.
        reason1.setReason('Size is big');
        reason2.setReason('Too big');

        returnReasons.add(reason1);
        returnReasons.add(reason2);

        for(commerce_ordermanagement.ProductData productData :
```

```
productExpandRequest.getProducts()) {
            // Set the return reasons for the products in the request
            productData.setReturnReasons(returnReasons);
            productList.add(productData);
        }

        response.setProductList(productList);
        response.setSucceed(true);

        // Process the product expand response with more custom logic.

        // Return the processed product expand response.
        return response;
    }
```

Instead of replacing the entire method with custom logic, you can call the base method with the super() method. You can still insert custom logic before and after the super() call.

```
public override commerce_ordermanagement.ProductExpandResponse
returnReasons(commerce_ordermanagement.ProductExpandRequest productExpandRequest) {
        // Get the product expand response using the default implementation of the
returnReasons method. Although returnReasons is called unmodified, we can supply it with
a modified version of the request.
        commerce_ordermanagement.ProductExpandResponse productExpandResponse =
super.returnReasons(productExpandRequest);

        // Modify the product return reasons response with custom logic.

        // Return the modified transactional return reasons response.
        return productExpandResponse;
    }
```

To get the return reasons from an external service, define a custom private method.

```
private Map<String, ProductReturnReasonsDataFromExternalService>
getReturnReasonsFromExternalService(Set<String> productIds){
        // Custom logic.
    }
```

You can create extension classes that can:

- Replace the entire default implementation of each base class method with custom logic.
- Add custom logic before calling the default implementation of a base class method.
- Add customer logic after calling the default implementation of a base class method.
- Add custom methods and classes.

To learn how to add your own Apex classes to a Salesforce org, see Adding an Apex Class.

Before you can fill an extension slot with a custom extension provider, create a web store. Register the extension class, and map the extension class to a B2C or D2C Commerce store. If you don't have a web store, create a dummy web store in your Developer Console. Go to Debug, open Execute Anonymous Window, and paste this code.

```
WebStore store = new WebStore(
      Name = 'SiteGenesis',
      // Either NET or GROSS
```

```
      DefaultTaxLocaleType = 'NET',
      // Format expected is instanceId@siteName; example abc_123@SiteGenesis
      ExternalReference = 'bblz_stg@SiteGenesis',
      Type = 'B2CE'
    );
insert store;
```

Click **Execute**.

To fetch the web store ID, execute this SOQL query

```
SELECT Id FROM WebStore LIMIT 1
```

and pick the web store ID from the returned values. Map your web store's ID to the EPN.

To register and map extension classes, we recommend using Salesforce CLI and the Salesforce Commerce plug-in for `sfdx`. To manage extensions, use the latest version of Salesforce CLI. For detailed instructions, install Salesforce CLI and update Salesforce CLI.

To install the plug-in, run `sfdx plugins:install @salesforce-commerce`.

Example commands. The store-ID is the web store ID.

Example:

```
# Register an extension class
     sfdx commerce:extension:register --targetusername user@abc.com --apex-class-name
 ProductExpandServiceSample --extension-point-name
Commerce_Domain_OrderManagement_Product --registered-extension-name
ProductExpandServiceSample

# Map an extension class
sfdx commerce:extension:map --registered-extension-name ProductExpandServiceSample
--store-id <store_id>

# Unmap an extension class
sfdx commerce:extension:unmap --registered-extension-name ProductExpandServiceSample
--store-id <store_id>
```

SEE ALSO:

Salesforce Help: Return Insights

API End-of-Life Policy

Salesforce is committed to supporting each API version for a minimum of 3 years from the date of first release. To improve the quality and performance of the API, versions that are over 3 years old sometimes are no longer supported.

Salesforce notifies customers who use an API version scheduled for deprecation at least 1 year before support for the version ends.