# CX for Industries

## Integrating Consumer Goods Mobile Direct-Store-Delivery

April 2023

CX for Industries
Integrating Consumer Goods Mobile Direct-Store-Delivery

April 2023

F77903-01

Author: Tejaswi Tatavarthi

# Contents

Get Help i

1 About This Guide 1

Audience and Scope 1

Related Guides 1

2 Introduction 3

Integration: Overview 3

3 Integrate Routes 5

Integrate Routes 5

Configure Integration Flows for Routes 6

Tables and Key Fields 6

4 Manage Route Inventory 9

Integrate Route Inventory 9

Configure Integration Flows for Route Inventory 10

Tables and Key Fields for Route Integration 12

5 Integrate Orders 15

Integrate Orders 15

Configure Integration Flows for Orders 16

Tables and Key Fields for Orders 19

6 Integrate Shipments 25

Integrate Shipments 25

Configure Integration Flows for Shipments 26

Tables and Key Fields for Shipments 27

7 Integrate Invoices 29

Integrate Invoices 29

Configure Integration Flows for Invoices 30
Tables and Key Fields for Invoices 30

8 Integrate Credit Lines 33
Integrate Credit Lines 33
Configure Integration Flows for Credit Lines 34
Tables and Key Fields for Credit Lines 35

9 Integrate Payments 39
Integrate Payments 39
Integrate Payment Types 40
Configure Integration for Payments 41
Tables and Key Fields for Payments 41

10 Integrate Store Credits 45
Integrate Store Credits 45
Configure Integration Flows for Store Credits 46
Integrate Tables and Key Fields for Store Credits 47

11 Integrate Empty Containers 51
Integrate Empty Containers 51
Configure Integration Flows for Empty Containers 52
Tables and Key Fields for Empty Containers 53

# Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

## Get Help in the Applications

Use help icons ⓘ to access help in the application. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons.

## Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

## Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

## Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

## Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

## Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

# 1 About This Guide

## Audience and Scope

This guide provides information about how to integrate back-end systems with the Direct-Store-Delivery processes in Oracle Customer Experience Cloud for the Consumer Goods mobile application.

The back-end systems include supply chain, accounting, and order management systems.

This guide describes how the integrated applications work with the consumer goods mobile and how to configure them to support the integrated business processes.

## Related Guides

See the following related guides to understand more about the integrations covered in this guide.

***Table***

| Title | Description |
|---|---|
| Oracle CX for Industries Using Consumer Goods | Describes the business flows and functionality in retail execution and trade-promotion execution areas, and how you can perform your day-to-day operations in these areas. |
| Oracle CX for Industries Getting Started with Consumer Goods Implementation | Describes how to set up and configure Oracle CX for Consumer Goods either while implementing or after implementing Oracle CX Sales and Oracle B2B Service. |
| Oracle CX for Industries Using Consumer Goods Mobile | Describes how to perform tasks related to retail execution and trade promotions management using consumer goods mobile application. |

# 2 Introduction

## Integration: Overview

DSD enables your sales team to deliver goods directly for your customers using mobile application. You can use mobile application in connected or disconnected mode for anytime, anywhere access to key information about customers, orders, route inventories, and promotions.

To use DSD processes, you must integrate them with your back end systems such as supply chain, accounting, or order management systems. When fully integrated, you can perform, track, measure, and manage your transactions at each stage of the delivery process, starting from checking your appointments to completing your visits, for better business results.

The following consumer goods DSD flows are available for integration:

- Integrate Routes: Covers the integration of route assignments in consumer goods with a route planning and optimization system.
- Integrate Route Inventory: Covers the integration of consumer goods mobile with a supply chain system to manage route inventories.
- Integrate Orders: Covers the integration of consumer goods mobile with order management system to manage order requests and order statuses.
- Integrate Shipments: Covers the integration of consumer goods mobile with a supply chain system to enable shipments.
- Integrate Invoices: Covers the integration of consumer goods mobile with Enterprise Resource Planning (ERP) accounting system for processing invoices.
- Integrate Credit Lines: Covers the integration of consumer goods mobile with an ERP accounting system to manage credit lines and payments.
- Integrate Payments: Covers the integration of consumer goods mobile with an ERP accounting system to manage payments.
- Integrate Store Credits: Covers the integration of consumer goods mobile with an ERP accounting system to manage store credits.
- Integrate Empty Containers: Covers the integration of consumer goods mobile with an ERP system to manage empty containers.

The chapters in this guide describe each of these integrations in detail. To integrate your back end systems, you must perform the configurations according to the instructions given in this guide.

# 3 Integrate Routes

## Integrate Routes

As sales representatives, you integrate the route assignments in the mobile with a route planning and optimization system.

In the mobile application, you must first check in to their assigned routes to carry out their activities for the day. While the consumer goods application manages the route assignments, to perform route-based retail execution you must integrate a route planning system to manage the account lineups and appointments for each such route assignment.

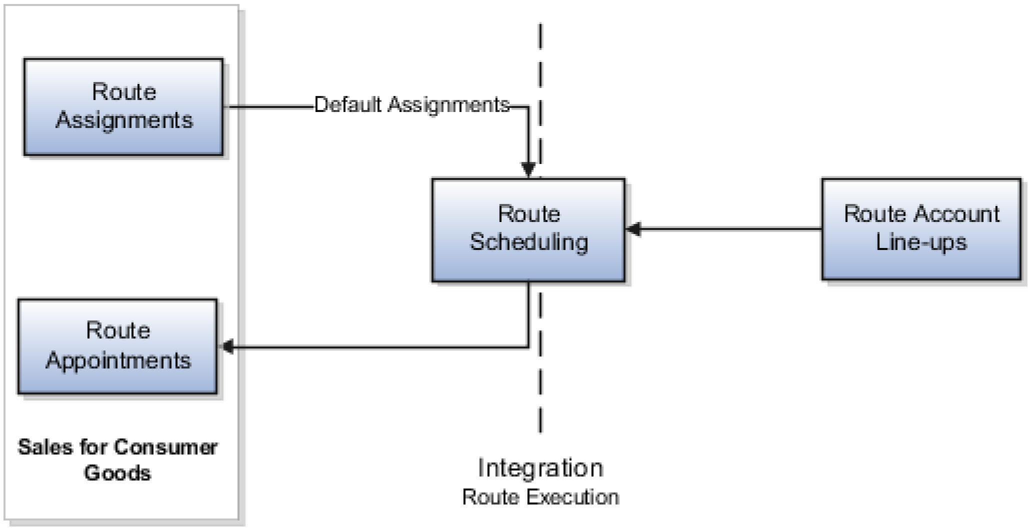
You can create or update route appointments using either consumer goods Web application or create these programmatically in your route planning system.

The integrated route execution assigns the appointments to the sales representatives based on the route account lineup and their default route assignments. For example, a sales representative who is assigned route A, is assigned to the appointments for the accounts on route A.

Here's how this integration works:

- The integrated system retrieves the default route assignments from consumer goods.
- The appointments for the route execution are created or updated based on the account lineups and the default route assignments in consumer goods.
- The appointments for a route are assigned to the same sales representative who was assigned that route.

The following graphic shows the route scheduling integration flow.



# Configure Integration Flows for Routes

You must configure the route execution process to enable the supported integration flows.

In the integrating system, configure the route-execution process to refresh or update the data, as applicable, each time it runs. For example, the process must detect the changes to route assignments and assign the sales representatives accordingly. You must run and complete the scheduling process for the next route execution before the mobile application starts synchronizing data for the next route execution.

# Tables and Key Fields

You must use or configure the tables and the key fields to support the integration.

To facilitate the data flow between multiple systems, you can integrate the following staging tables with the store route information:

- Route Allocation
- Activities

## Route Allocation

Use the following table and key fields in the Route Allocation `(__ORACO__RouteAllocation_c)` staging table for default route assignments.

| Key Fields | API Names | Description |
|---|---|---|
| Route | `__ORACO__Route_c` | Indicates the route assigned by default to the sales representatives. |
| Primary | `__ORACO__SalesRepresentative_c` | Indicates the assigned sales representative of the route. |
| Resource | `__ORACO__Primary_c` | Indicates whether the assigned resource is the primary sales representative.<br>Each route can have only one primary sales representative. The primary route resource must also be the primary sales representative for the route appointments. |

## Activities

Use the following table and the key fields in the Activity table for appointments and tasks.

| Key Fields | API Names | Description |
|---|---|---|
| Route | `__ORACO__Route_c` | The route to which the appointment is assigned. |

To import Activities, see Understanding Import and Export Management for CX Sales and B2B Service guide at https://docs.oracle.com

# 4 Manage Route Inventory

## Integrate Route Inventory

You can understand the integration of the consumer goods mobile application with supply chain system and manage route inventories.

During the route execution, the mobile application captures and maintains the route-inventory transactions. The following inventory transactions are read-only in the sales application:

- **Restock for Shipment**: Increases the route inventory with the quantity that's required for a shipment. It also provides the sellable quantity of the transaction.
- **Restock for Additional Inventory**: Increases the route inventory with the quantity that's required due to reasons other than shipment. It also provides the sellable quantity of the transaction.
- **Delivery**: Decreases the route inventory based on the quantity delivered in a shipment. It also provides the sellable quantity of the transaction.
- **Damage**: Changes the route inventory from sellable to unsellable. When the quantity of sellable inventory decreases, the quantity of the unsellable inventory is increased by the same amount.
- **Loss**: Decreases the route inventory based on the quantity of the inventory lost. It also provides the sellable quantity of the transaction.
- **Recovery**: Increases the route inventory based on the quantity of the inventory recovered. It also provides the sellable quantity of the transaction.
- **Return**: Increases the route inventory based on the quantity of the inventory returned. It also provides the sellable quantity of the transaction.
- **Unload**: This transaction decreases the route inventory based on the quantity of the inventory unloaded. It also provides the sellable quantity of the transaction.

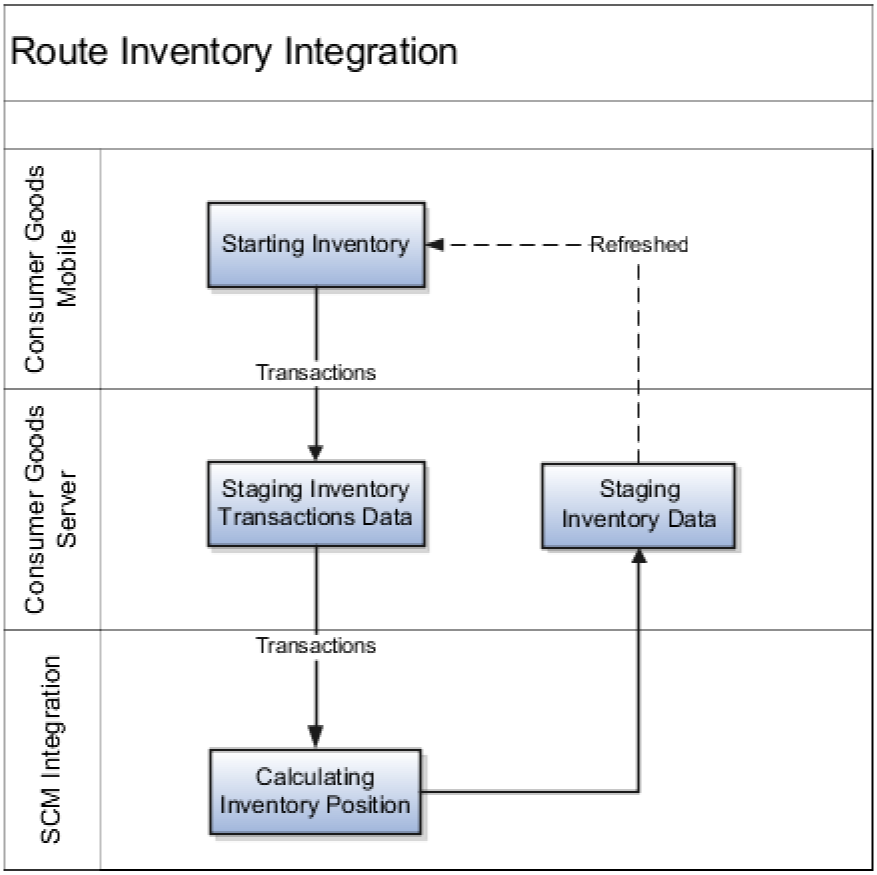
To update the inventory records periodically or after each route execution, you must use Supply Chain Management (SCM) system to process and refresh inventory position in the mobile application.

Depending on your business use case, you can first determine how you want to refresh your inventory records and then configure your system accordingly.

Here's how this integration works when using the Maintaining Static Inventory Position method of calculating the inventory level:

1. The SCM system recalculates inventory using the Maintaining Static Inventory Position method and sends the updated records to the consumer goods server.
2. The CX Retail Execution Mobile retrieves the refreshed inventory records from the consumer goods server to update the inventory position for the next route execution.
3. CX Retail Execution Mobile sends transaction records created during route execution to the consumer goods server.
4. The consumer goods server prepares the transactions records and makes them available to the integrating SCM system.

The following graphic shows the integration flow:



In this integration, the mobile application calculates the inventory position as follows:

1. Reads the route-inventory level and the **Last Processed Transaction Time Stamp** from the Route Inventory staging table.
2. Reads all transactions later than the **Last Processed Transaction Time Stamp**. See, Integration Tables and Key Fields.
3. Adds the transaction quantity of each transaction to the original product inventory level.

The inventory position thus calculated by the mobile application is temporary and isn't sent to the consumer goods server. The inventory transactions that remove items are in the negative and those that add items are in the positive.

# Configure Integration Flows for Route Inventory

You configure data transfer to enable integration flows. Here's how you can transfer data to or from Oracle CX Sales to integrated applications and use the integration methods for updating route-inventory records.

## Maintain Static Inventory Position

In the static inventory method, the mobile application uses the Route Inventory Transactions staging table to calculate the inventory position. You can use this method when the mobile device doesn't download partial inventory records.

**Note:** Before using this method, ensure that the mobile device and the integrating system aren't performing any read-write operation in the transactions table.

To maintain static inventory position, transfer all route inventory transactions from the Route Inventory Transactions table to the integrating system. The integrating system purges all route inventory and route inventory transactions and uploads the new route inventory to the Route Inventory Transactions table.

For highly dynamic inventories, you must use the Maintaining Inventory Position Using Transactions method.

## Maintain Inventory Position Using Transactions

In the transactions method, the mobile application uses the Route Inventory Transactions table to calculate the inventory position. Use this method where the inventory transactions between the mobile application and the back-office inventory system happen throughout the day.

This integration method involves a bidirectional flow of transactions between the integrating system and the mobile application.

To maintain inventory position, the integrating system stores these transactions in the Route Inventory Transaction table:

- **Restock for Shipment**: Records the transfer of inventory from the distribution center to the route for delivery to a specific customer.
- **Restock for Additional Inventory**: Records the transfer of inventory from the distribution center to the route in cases where the inventory isn't allocated to a specific customer.
- **Unload**: Records the transfer of inventory from the route to the distribution center.

**Note:** Transactions method doesn't use the Route Inventory table.

The mobile device first totals all restock transactions and then deducts from it all the unload transactions to calculate the inventory position.

## Maintain Inventory Using Static Inventory Position and Transactions

Use a mix of static inventory method and the transactions method to maintain the inventory position.

To maintain inventory using both the methods, configure the route by initializing its Route Inventory staging table and clearing its Route Inventory Transaction staging table. The inventory system records these transactions in the Route Inventory Transaction table:

- **Restock for Shipment**: Records the transfer of inventory from the distribution center to the route for delivery to a specific customer.
- **Restock for Additional Inventory**: Records the transfer of inventory from the distribution center to the route in cases where the inventory isn't allocated to a specific customer.
- **Unload**: Records the transfer of inventory from the route to the distribution center.

You can periodically reconcile the route inventory transactions into the Route Inventory staging table. You can use the Static Inventory method to perform this if the transactions can be downloaded, processed, and deleted.

**Note:** During this process, ensure that the mobile application isn't reading the information from the Route Inventory staging table.

In cases, where the mobile application is simultaneously reading the Route Inventory staging table, configure the Supply Chain Management (SCM) system to perform these tasks:

1. Read the current route inventory level for a product and note the **Last Processed Transaction Time Stamp** of those products.
2. Retrieve and process the route inventory transactions for products where the **Creation Time Stamp** is later than the **Last Processed Transaction Time Stamp**.

   You must configure the inventory system to process the retrieved transactions in the descending order of the **Creation Time Stamp**. This means that the most recent transaction is processed first.

   To prevent the inventory system from processing in-progress transaction records based on most recent **Creation Time Stamp**, do either of these in the inventory system:

   - Induce delay in processing the transaction record having the most recent **Creation Time Stamp**.
   - Process transaction records up to but not including the most recent **Creation Time Stamp**.

   Update the **Last Processed Transaction Time Stamp** with the **Creation Time Stamp** of the last processed record after the inventory level is updated. The same operation must update both the fields.

You can also configure the integrating system to reconcile the records multiple times in a day.

# Tables and Key Fields for Route Integration

You must use or configure the tables and the key fields to support the integration.

To facilitate the data flow between multiple systems, this integration uses the following staging tables to store route inventory information:

- Route Inventory
- Route Inventory Transactions

## Route Inventory

Use the following table lists the key fields in the Route Inventory `(__ORACO__RouteInventory_c)` staging table, which contains the information about the current inventory.

| Key Fields | API Name | Description |
|---|---|---|
| Route ID<br>Product | ID<br>`__ORACO__Product_c` | Indicates the user keys that uniquely identify each inventory record. |
| Total Quantity | `__ORACO__TotalQuantity_c` | Indicates the quantity of inventory in a route. |
| Sellable Quantity | `__ORACO__SellableQuantity_c` | Indicates the sellable subset of the total quantity of inventory in a vehicle. |

| Key Fields | API Name | Description |
|---|---|---|
| Unsellable Quantity | `__ORACO__UnsellableQuantity_c` | Indicates the unsellable subset of the total quantity of inventory in a vehicle. |
| Last Processed Transaction Time Stamp | `__ORACO__LastProcessedTimestamp_c` | Indicates the last creation time stamp from the Route Inventory Transaction table. This time stamp is used for calculating the inventory position for the route. See, Configuring Integration Flows on how this time stamp is used for refreshing inventory position. |

## Route Inventory Transactions

Use the following table and the key fields in the Route Inventory Transactions (`__ORACO__RouteInvTransDSD_c`) staging table, which contains the information about the inventory transactions.

| Key Fields | API Name | Description |
|---|---|---|
| Route ID | Id | A unique and incremental database sequence ID generated by application when a record is created. |
| Creating Time Stamp | `__ORACO__CreatedOn_c` | The date when the inventory transaction record was created. |
| Route ID<br>Product | `__ORACO__TransferRouteID_c`<br>`__ORACO__Product_c` | The keys that uniquely identify each inventory transaction. These correspond to a Route ID and the associated product. |
| Transaction Date<br>Transaction Type<br>Transaction Quantity | `__ORACO__TransactionDate_c`<br>`__ORACO__TransactionType_c`<br>`__ORACO__TransactionQuantity_c` | These fields describe how the route inventory has changed. |

# 5 Integrate Orders

## Integrate Orders

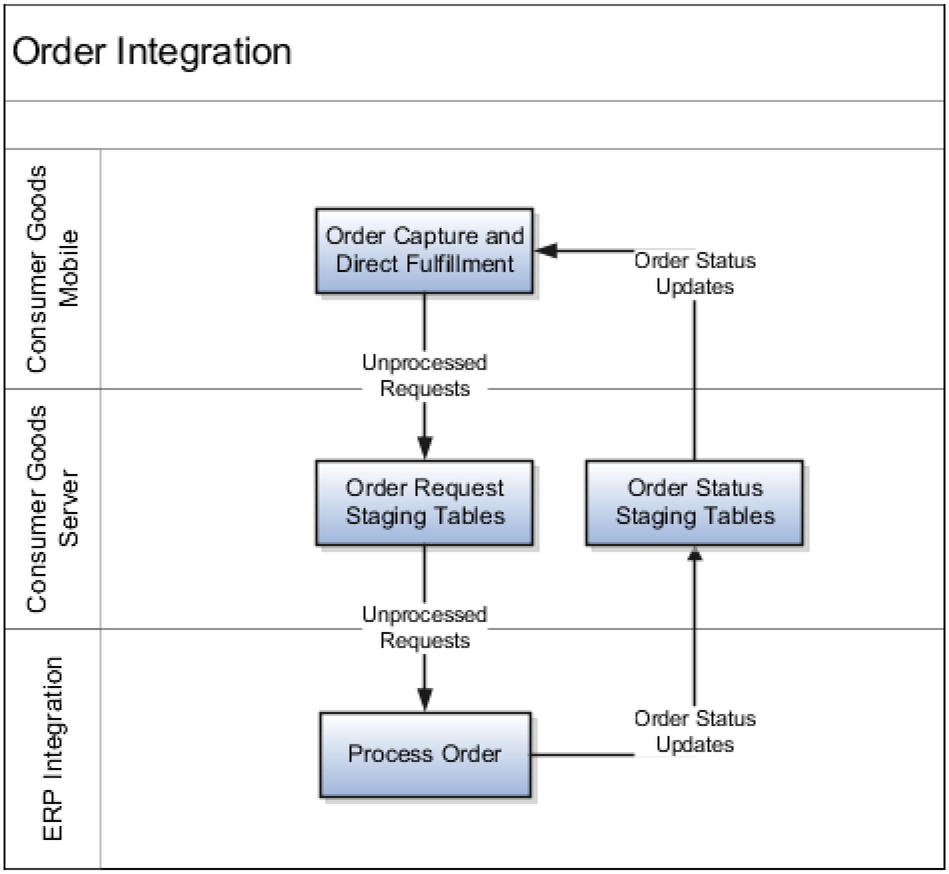
This topic helps sales representatives with the integration of your mobile application with order management system. The consumer goods application enables you to capture order requests. To process the order requests further through order to cash business processes, you must integrate an order

When fully integrated, you can capture order requests and track order fulfillment that can happen either through direct delivery or through the integrated shipment process. The integration enables you to track the status of orders based on statuses such as revisions, returns, and cancellations to address customer queries related to orders.

Here's how the order integration process works:

1. The mobile application captures order requests and transfers the unprocessed requests to the staging table.
2. The integrating system picks up the unprocessed requests and updates the status of each order based on the type of request.
3. The consumer goods mobile application picks up the order status.

The following graphic shows the order integration flow.



The order integration also uses information from other integrations such as route inventory, invoicing, and shipments. For specifics about those integrations, see the relevant chapters in this guide.

# Configure Integration Flows for Orders

As an administrator, you configure the integration system based on how the consumer goods application manages order requests and how it updates the values in staging tables.

## Types of Order Requests

The consumer goods application updates the staging tables based on the type of order request.

Here are the values you can use to identify the type of the order requests in the Order Request table:

**New Order**

The new order request has **Revision Number** of **1**.

**Update Order**

The update order request has **Revision Number** greater than **1** and the **Update Requested** set to **True**. Additionally, corresponding order request lines should have new revision number.

**Cancellation Order**

The cancellation order request has **Revision Number** greater than **1** and the **Cancellation Requested** set to **True**. Additionally, the corresponding order request lines should have new revision number.

> **Note:** Remember you can only cancel the entire order and not the individual line items.

**Return Order**

The new order request has **Revision Number** of **1** and **Order Type** of **Return**. Additionally, the products to be returned are captured as its order line requests. These products may or may not be from the same order.

**Direct Order (Auto-Sales)**

The new order request has **Revision Number** of **1**. Use the **Use Quantity Delivered at Creation** in the Order Line Request table to identify whether the sales representative has fulfilled the entire order request from the route inventory.

Sales representatives create and deliver direct orders or auto-sales orders immediately during route execution. The products delivered are captured as its order line requests using the field **Quantity Delivered at Creation** in the Order Line Request table.

The integrating system generates shipments and marks them as delivered using the field **Quantity Delivered at Creation**. In this case, the integration must handle transactions for inventory delivery such that it doesn't trigger a double decrement.

## Process Order Requests

You can configure the integrating system to identify and process the order requests.

Here's how the integration system can retrieve order requests:

1. Read the value of the **Process Status** field to distinguish between unprocessed and processed orders. For unprocessed orders, the value of the **Process Status** field is **Ready to process**.
2. Set the value of **Process Status** to **Processed** after processing the orders.

## Update Order Status Based on Requests

You configure the integrating system to update the status of an order along with other fields in the staging tables.

You must configure the integrating system to update the order status each time an action is taken on a request. Ensure Order Status and Order Line Status staging tables are populated from the corresponding Order Request and Order Line Request staging tables. Additionally, link the Order Line Status records to the parent Order Status records using the internal ID of the Order Status record.

Here's how the integration system must set the order status based on the action taken on an order:

| On This Action | Order Status is Set To | Here's How |
|---|---|---|
| Order is read by integrating system and it can be fulfilled | Booked | • In the Order Status table, creates a record with the value of **Status** and **Last Request Status** as **Booked**.<br>• In the Order Line Status table, creates a record for each order line with the value of **Status** and **Last Request Status** as **Booked**. |
| Order is read by the integrating system and it can't be fulfilled | Booking Rejected | • In the Order Status table, creates a record with the value of **Status** and **Last Request Status** as **Booking Rejected**.<br>• In the Order Line Status table, creates a record for each order line with the value of **Status** and **Last Request Status** as **Booking Rejected**. |
| Some of the order lines are delivered | Partially Delivered | • In the Order Status table, creates a record with the value of **Status** and **Last Request Status** as **Partially Delivered**.<br>• In the Order Line Status table, creates a record for each order line with the value of **Status** and **Last Request Status** as **Partially Delivered**.<br>**Note:**<br>For each order line not delivered, creates an Order Line Status record with the previous status value. |
| All order lines are delivered | Delivered | • In the Order Status table, creates a record with the value of **Status** and **Last Request Status** as **Delivered**.<br>• In the Order Line Status table, creates a record for each order line with the value |

| On This Action | Order Status is Set To | Here's How |
|---|---|---|
| | | of **Status** and **Last Request Status** as **Delivered**. |
| An update to order is requested through order revision | Booked or Partially Delivered | **Note:**<br>In an order revision, multiple order lines may have updates. The integration doesn't support partial updates or updates to some of the order lines in the order. Either accepts all the updates in an order revision or rejects all the updates.<br>If the updated order can be fulfilled:<br>• In the Order Status table, creates a record with the value of **Status** as either **Booked** or **Partially Delivered**, as applicable, and **Last Request Status** as **Updated**.<br>• In the Order Line Status table, creates a record for each order line with the value of **Status** as either **Booked** or **Partially Delivered**, as applicable, and **Last Request Status** as **Updated**.<br>If the updated order isn't fulfilled:<br>• In the Order Status table, creates a record with the value of **Status** as either **Booked** or **Partially Delivered**, as applicable, and **Last Request Status** as **Update Rejected**.<br>• In the Order Line Status table, creates a record for each order line with the value of **Status** as either **Booked** or **Partially Delivered**, as applicable, and **Last Request Status** as **Update Rejected**. |
| Order cancellation request | Canceled or Canceled Rejected | If the order can be canceled:<br>• In the Order Status table, creates a record with the value of **Status** and **Last Request Status** as **Canceled**.<br>• In the Order Line Status table, creates a record for each order line with the value of **Status** and **Last Request Status** as **Canceled**.<br>If the order isn't canceled:<br>• In the Order Status table, creates a record with the value of **Status** as either **Booked** or **Delivered**, as applicable, and **Last Request Status** as **Canceled Rejected**.<br>• In the Order Line Status table, creates a record for each order line with the value of **Status** as either **Booked** or **Delivered**, as applicable, and **Last Request Status** as **Canceled Rejected**. |

# Tables and Key Fields for Orders

As an administrator or implementor, use or configure these tables and key fields to support the integration.

The creation and fulfillment of an order requires data flow between multiple systems that may work offline or independent of each other. The integration uses these staging tables to store the order information:

- DSD Order Request
- DSD Order Line Request
- DSD Order Status
- DSD Order Line Status

## DSD Order Request

Here are the key fields in the DSD Order Request (`__ORACO__OrderDSD_c`) staging table that you use or configure to support the integration. This table captures and stores information about the order requests from telesales and the mobile application. The integrating system must read from this table to identify and process the revisions in order.

| Key Fields | API Name | Description |
|---|---|---|
| Account | `__ORACO__Account_c` | The account to which the order belongs. |
| Order Type | `__ORACO__OrderType_c` | Differentiates the types of orders. |
| Source System<br>Source System Number | `__ORACO__OrderSS_c`<br>`__ORACO__OrderSSN_c` | A unique combination for identifying orders.<br>• Source System: A unique identifier that the mobile device issues. This must be unique across different originating systems.<br>• Source System Number: A unique number within each source system.<br>**Note:** Source System and Source System Number of an order is the same across Order Request records and Order Line Request records. |
| Revision Number | `__ORACO__OrderRN_c` | Represents the revisions of an order. This must start from 1 (when creating an order) and then increment by 1 (for subsequent revisions). |

| Key Fields | API Name | Description |
|---|---|---|
| | | **Note:**<br>Revision Number is the same across the order request and its order line request for each revision. Every revision of an order includes all of its line items. |
| Process Status | `__ORACO__ProcessStatus_c` | Differentiates between new or processed orders. |
| Cancellation Requested | `__ORACO__CancelRequestFlag_c` | Captures cancellation request. |
| Update Requested | `__ORACO__UpdateRequestFlag_c` | Captures update request. |
| Creation Time | `__ORACO__CreationTime_c` | Captures the order creation time. |
| Order | `__ORACO__Order_c`<br>`(__ORACO__Order_Id_c)` | The foreign key to Order Rollup record of the Order Request table.<br>The Order Rollup object is internal to Oracle CX Sales for displaying a consolidated order record on the Web UI. |

## DSD Order Line Request

Here are the key fields in the DSD Order Line Request `(__ORACO__OrderLineDSD_c)` staging table that you use or configure to support the integration. This table stores the order lines provided by the captured order request. The integrating system must read from this table to identify and process the revisions in order.

| Key Fields | API Name | Description |
|---|---|---|
| Source System<br>Source System Number<br>Revision Number | `__ORACO__OrderLineSS_c`<br>`__ORACO__OrderLineSSN_c`<br>`__ORACO__OrderLineRN_c` | The source system, source system number, and revision number of the corresponding order request.<br>• Source System: A unique identifier that the mobile device issues. This must be unique across different originating systems.<br>• Source System Number: A unique number within each source system.<br>**Note:**<br>Source System and Source System Number of an order is the same across Order Request records and Order Line Request records.<br>• Represents the revisions of an Order. This must start from 1 (when creating an order) |

| Key Fields | API Name | Description |
|---|---|---|
| | | and then increment by 1 (for subsequent revisions).<br>**Note:**<br>Revision Number is the same across the Order Request and its Order Line Request for each revision. Every revision of an order includes all of its line items. |
| Line Number | `__ORACO__OrderLineLN_c` | Uniquely identifies an order line of an order when used together with Source System, Source System Number, and Revision Number. |
| Product | `__ORACO__Product_c`<br>`(__ORACO__Product_Id_c)` | The product of the order line. |
| Quantity | `__ORACO__Quantity_c` | The quantity of the products in the order line. |
| Quantity Delivered at Creating | `__ORACO__QuantDelAtCreate_c` | The quantity already delivered while taking the order. Used for partial deliveries. |
| Order | `__ORACO__Order_c`<br>`(__ORACO__Order_Id_c)` | The foreign key to order request of the Order Line table. |
| Order Line | `__ORACO__OrderLine_c`<br>`(__ORACO__OrderLine_Id_c)` | The foreign key to order line rollup of the Order Line table. |

The integrating system can uniquely identify orders using a combination of fields in cases where mobile devices may have generated the orders. These fields enable distributed systems to operate in situations where the Order ID is unknown.

These fields are covered in the Order Request and Order Line Request tables:

- Source System
- Source System Number
- Revision Number
- Line Number

> **Note:** These fields don't replace or invalidate the use of ID fields. Oracle CX Sales enforces that the ID fields are available, unique, and automatically populated for all the objects.

If the integrating system posts multiple Order Status revisions (with corresponding lines) for the same order, the application uses the latest revision for calculating the final status.

## DSD Order Status

Here are the key fields in the DSD Order Status (`__ORACO__OrderStatusDSD_c`) staging table that you use or configure to support the integration. This table captures order status updates from the order management system.

| Key Fields | API Name | Description |
|---|---|---|
| Order | `__ORACO__Order_c`<br>`(__ORACO__Order_Id_c)` | The corresponding order request. |
| Order - Source System | `__ORACO__OrderSS_c` | The source system of the corresponding order request. |
| Order - Source System Number | `__ORACO__OrderSSN_c` | The source system number of the corresponding order request. |
| Order - Revision Number | `__ORACO__OrderRN_c` | The revision number of the corresponding order request. |
| Revision Number | `__ORACO__RefOrderRN_c` | The revision number of the order status. This must start from 1 and then increment for each subsequent order status of an order. |
| Last Request Status | `__ORACO__LastRequestStatus_c` | The status of the corresponding order request. |
| Status | `__ORACO__Status_c` | The final status of the order. |

## DSD Order Line Status

Here are the key fields in the DSD Order Line Status (`__ORACO__OrderLStatusDSD_c`) staging table that you use or configure to support the integration. This table provides information about the order lines based on the status update of the orders.

| Key Fields | API Name | Description |
|---|---|---|
| Order Line | `__ORACO__OrderLine_c` | The corresponding order line request. |
| Order Line - Source System | `__ORACO__OrderLineSS_c` | The source system of the corresponding order line request. |
| Order Line - Source System Number | `__ORACO__OrderLineSSN_c` | The source system number of the corresponding order line request. |
| Order Line - Line Number | `__ORACO__OrderLineNumber_c` | The line number of the corresponding order line request. |

| Key Fields | API Name | Description |
|---|---|---|
| Order Line - Revision Number | `__ORACO__OrderLineRN_c` | The revision number of the corresponding order line request. |
| Order Status | `__ORACO__OrderStatus_c` | The parent order status of the order line status. |
| Revision Number | `__ORACO__OrderLineStatusRN_c` | The revision number of the order line status. |
| Last Request Status | `__ORACO__LastRequestStatus_c` | The status of the corresponding order line request. |
| Status | `__ORACO__Status_c` | The final status of the order line. |

# 6 Integrate Shipments

## Integrate Shipments

As sales administrator, you can integrate the consumer goods mobile application with a supply chain system to manage shipments.

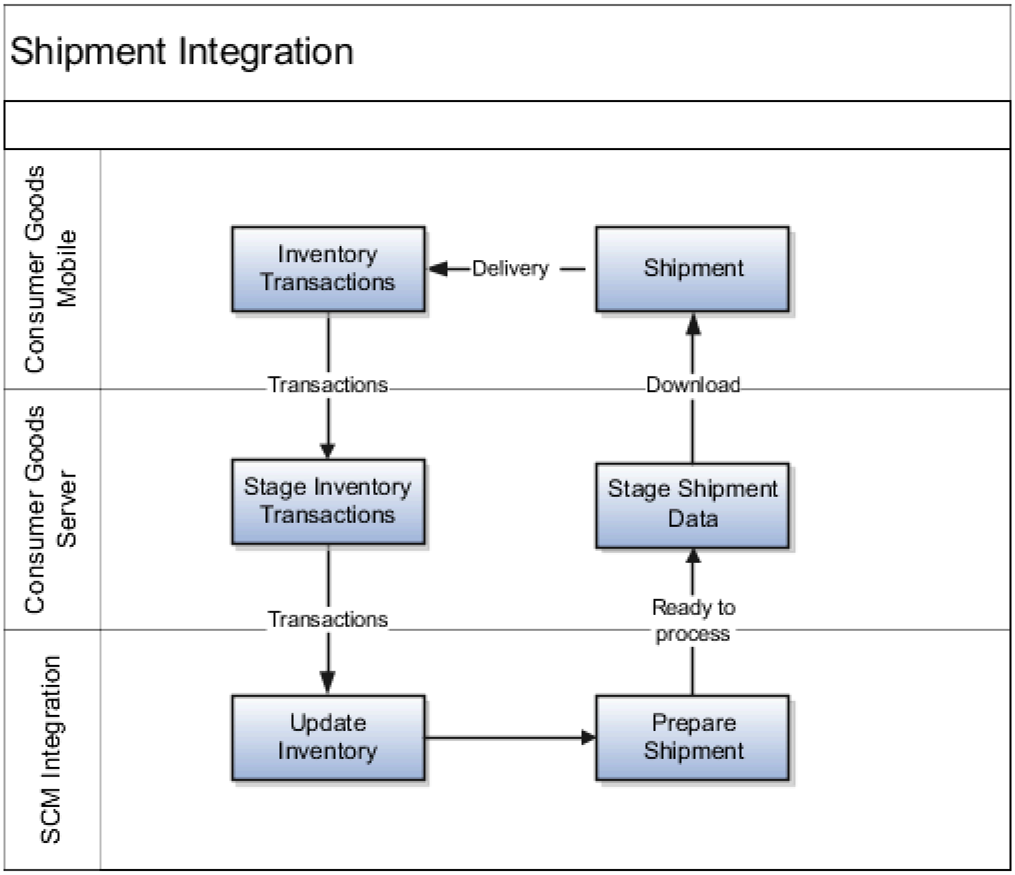
To map the consumer goods inventory transactions with supply chain. you must integrate it with a supply chain system to get the shipment details. Depending on your business requirements, you must configure the Supply Chain Management (SCM) system to process the captured order requests.

For more information about how to configure the integrating SCM application to transfer data to the consumer goods server, see Configuring Integration Flows.

Here's how this integration flow works:

1. The integration uploads the shipment data to consumer goods server and the consumer goods mobile application picks it up.
2. The consumer goods mobile application delivers the shipments and creates corresponding inventory transactions.
3. The consumer goods mobile application uploads these inventory transactions to the staging tables of the consumer goods server.
4. The integration system picks up the inventory transactions and uploads them to SCM. The following graphic shows the shipment integration flow.

The following graphic shows the shipment integration flow.



# Configure Integration Flows for Shipments

You must configure the shipments to enable the supported integration flows. You can transfer the data to or from the sales application. You can integrate the application and transfer shipment details from the SCM to the consumer goods mobile application.

## Transfer Shipment Details

The mobile application can read-only complete shipment records. To identify or capture completed shipments, the value in the **Process Status** field is used.

The **Process Status** field is a static pick list on the shipments header record in SCM. After the shipment headers and all shipment lines are created, SCM must set this **Process Status** field to **Ready to process**.

To transfer shipments, configure your SCM to do the following tasks:

1. Create a shipment header with Process Status as NULL.
2. Retrieve Shipment ID of the shipment header.
3. Create all shipment lines using the Shipment ID.

4. Update the **Process Status** in the shipment header to **Ready to process**.

> **Note:** Do not update shipment and shipment lines for shipment headers in Ready to process status.

## Set the Delivery Status of Shipment

When you deliver shipments using the consumer goods mobile application, corresponding inventory transactions are generated for each shipment. You must configure SCM to identify and track whether the shipments were delivered by using the corresponding inventory transactions.

# Tables and Key Fields for Shipments

You must use or configure the tables and the key fields to support the integration.

As a sales administrator or implementor, use this integration to facilitate the data flow between multiple systems to store shipment information:

- DSD Shipment
- DSD Shipment Line

## DSD Shipment

Use the following table and the key fields in the DSD Shipment (`__ORACO__ShipmentDSD_c)` staging table used for storing shipment header information.

| Key Fields | API Name | Description |
|---|---|---|
| Shipment Number | `RecordName` | A unique alphanumeric key in SCM that identifies a shipment. |
| Account | `__ORACO__Account_c`<br>`(__ORACO__Account_Id_c)` | The retailer account to which the shipment is delivered. You must ensure that this field is correctly set. |
| Order Number | `__ORACO__OrderRN_c` | The order for which the shipment was created. |
| Shipment Date | `__ORACO__ScheduledDate_c` | The shipment date provided by SCM. |
| Legal Entity | `__ORACO__LegalEntity_c`<br>`(__ORACO__LegalEntity_Id_c)` | The manufacturer legal entity for the shipment. |
| Source System | `__ORACO__OrderSS_c` | Uniquely identifies the source for the shipment to track shipment. |
| Source System Number | `__ORACO__OrderSSN_c` | The unique number within each source system for a shipment to track shipment. |

| Key Fields | API Name | Description |
|---|---|---|
| | | |
| Created On | `__ORACO__CreatedOn_c` | This field must be set to 0 (zero) to identify the new records from SCM. |

## DSD Shipment Line

Use the following table lists the key field in the DSD Shipment Line (__ORACO__ShipmentLineDSD_c) staging table for storing shipment line information.

| Key Fields | API Name | Description |
|---|---|---|
| Shipment Line Number | `RecordName` | A globally unique alphanumeric string provided by SCM for each shipment line. |
| Line Number | `__ORACO__ShipmentLineLN_c` | A numeric and unique line number within a shipment. |
| Shipment | `__ORACO__Shipment_c`<br>`(__ORACO__Shipment_Id_c)` | The shipment header associated with the shipment line. |
| Account | `__ORACO__Account_c`<br>`(__ORACO__Account_Id_c)` | The retailer account to which the shipment is delivered. Same as the Account field in the Shipment header. This field must be set correctly. |
| Product | `__ORACO__Product_c`<br>`(__ORACO__Product_Id_c)` | The product being shipped. |
| Quantity | `__ORACO__Quantity_c` | The quantity of the product being shipped. |
| Source System | `__ORACO__ShipmentLineSS_c` | Uniquely identifies the source for the shipment to track shipment. |
| Source System Number | `__ORACO__ShipmentLineSSN_c` | The unique number within each source system for a shipment to track shipment. |
| Line Number | `__ORACO__ShipmentLineLN_c` | The unique numeric line number within a shipment to track shipment. |
| Created On | `__ORACO__CreatedOn_c` | This must be set to 0 (zero) to identify the new records from SCM. |

# 7 Integrate Invoices

## Integrate Invoices

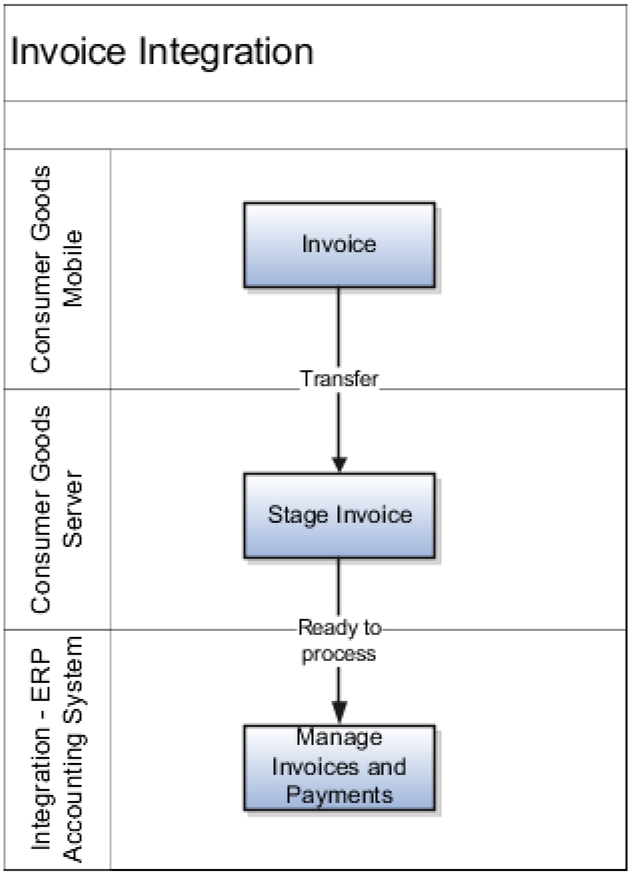
You can integrate the consumer goods mobile application with Enterprise Resource Planning (ERP) accounting system for processing invoices.

As sales representatives, you can generate and print invoices for deliveries. To further manage and process these invoices for accounting purposes, you must integrate your back-office accounting system with the mobile application.

Here's how this integration works:

1. The consumer goods mobile generates and transfers invoices to the consumer goods server.
2. The consumer goods server prepares the invoices for the integrated accounting system to pick them up.
3. The accounting system further manages and maintains the invoice data.

The following graphic shows the integration process flow.



# Configure Integration Flows for Invoices

You must configure the invoices process to enable the supported integration flows. It includes the information about how the accounting system picks up the invoices generated by the CX Retail Execution Mobile.

## Process Invoices Created by CX Retail Execution Mobile

Use the **Process Status** field in each of the generated invoices to indicate whether the integrating system picks up the invoice.

You must configure the integrating ERP system to perform the following:

1. Read all invoice records where the **Process Status** is **Ready to process**.
2. Read all invoice lines of the **Ready to process** invoices.
3. Set the **Process Status** of those invoices to **Processed**.

# Tables and Key Fields for Invoices

As an administrator or implementor, understand the tables and the key fields that you must configure to support the integration.

To facilitate the data flow between multiple systems, this integration uses these staging tables to store invoice information:

- DSD Invoice
- DSD Invoice Line

## DSD Invoice

Here are the key fields in the DSD Invoice `(_ORACO__InvoiceDSD_c)` staging table that you must configure for storing invoice header information.

| Key Fields | API Name | Description |
|---|---|---|
| Invoice Number | `RecordName` | A unique alphanumeric key that identifies the invoices generated by the mobile application. |
| Account | `__ORACO__Account_c`<br>`(__ORACO__Account_Id_c)` | Indicates the account that's associated with the invoice. |
| Shipment | `__ORACO__Shipment_c`<br>`(__ORACO__Shipment_Id_c)` | Indicates the shipment associated with the invoice. |

| Key Fields | API Name | Description |
|---|---|---|
| Process Status | `__ORACO__ProcessStatus_c` | Indicates whether the invoice is ready for processing. Accounting system must process only those invoices where this field value is **Ready to Process**. |
| Order Source System | `__ORACO__OrderSS_c` | Indicates the source application of the order associated with the invoice. The Order Source System is copied from the shipment for which the invoice was created. |
| Order Source System Number | `__ORACO__OrderSSN_c` | The source application number of the order associated with the invoice. The Order Source System Number is copied from the shipment for which the invoice was created. |
| Order Revision Number | `__ORACO__OrderRN_c` | The revision number of the order associated with the invoice. |
| Order Number | `__ORACO__OrderNumber_c` | The order number of the order associated with the invoice. This value is picked up from the shipment for which the invoice was created. |
| Legal Entity | `__ORACO__LegalEntity_c`<br>`(__ORACO__LegalEntity_Id_c)` | The manufacturer legal entity for all products in an invoice. |
| Source System<br>Source System Number | `__ORACO__InvoiceSS_c`<br>`__ORACO__InvoiceSSN_c` | These are the unique user keys for an invoice record generated by the mobile application. |

## DSD Invoice Line

Here are the key fields in the DSD Invoice Line `(_ORACO__InvoiceLineDSD_c)` staging table that you must configure for storing invoice lines in the invoice.

| Key Fields | API Name | Description |
|---|---|---|
| Invoice Line Number | `RecordName` | A unique alphanumeric key generated by the mobile application. |
| Line Number | `__ORACO__InvoiceLineLN_c` | A unique line number within an invoice and has a numeric value. |
| Invoice | `__ORACO__Invoice_c`<br>`(__ORACO__Invoice_Id_c)` | The invoice header associated with the invoice line. |
| Product | `__ORACO__Product_c` | The product for which the invoice is generated. |

| Key Fields | API Name | Description |
|---|---|---|
| | `(__ORACO__Product_Id_c)` | |
| Quantity | `__ORACO__Quantity_c` | The quantity of the product invoiced. |
| Shipment Line | `__ORACO__ShipmentLine_c`<br>`(__ORACO__ShipmentLine_Id_c)` | The shipment line associated with the invoice line. |
| Order Line Source System | `__ORACO__OrderLineSS_c` | The source application of the order line that's associated with the invoice line. The Order Line Source System is copied from the shipment line for which the invoice line was created. It must have the same value as the Order Source System field in the invoice header. |
| Order Line Source System Number | `__ORACO__OrderLineSSN_c` | The source application number of the order line associated with the invoice line. The Order Line Source System is copied from the shipment line for which the invoice line was created. It must have the same value as the Order - Source System Number field in the invoice header. |
| Order Line Revision Number | `__ORACO__OrderLineRN_c` | The revision number of the order line associated with the invoice line. |
| Order Line Number | `__ORACO__OrderLineNumber_c` | The number of the order line associated with the invoice line. The order line is copied from the shipment line for which the invoice line was created. |
| Source System<br>Source System Number | `__ORACO__InvoiceLineSS_c`<br>`__ORACO__InvoiceLineSSN_c` | The unique user keys for an invoice record generated by the mobile application. They must have the same values as those in the invoice header. |

# 8 Integrate Credit Lines

## Integrate Credit Lines

As sales administrator or implementor, you can integrate CX Retail Execution Mobile with an Enterprise Resource Planning (ERP) accounting system to manage credit lines.

In CX Retail Execution Mobile, sales representatives can view the credit lines available to the customers and can create the payment transactions against those credit lines. To further process these payment transactions and get updated credit lines after processing the payments, you must integrate an ERP system with CX Retail Execution Mobile.

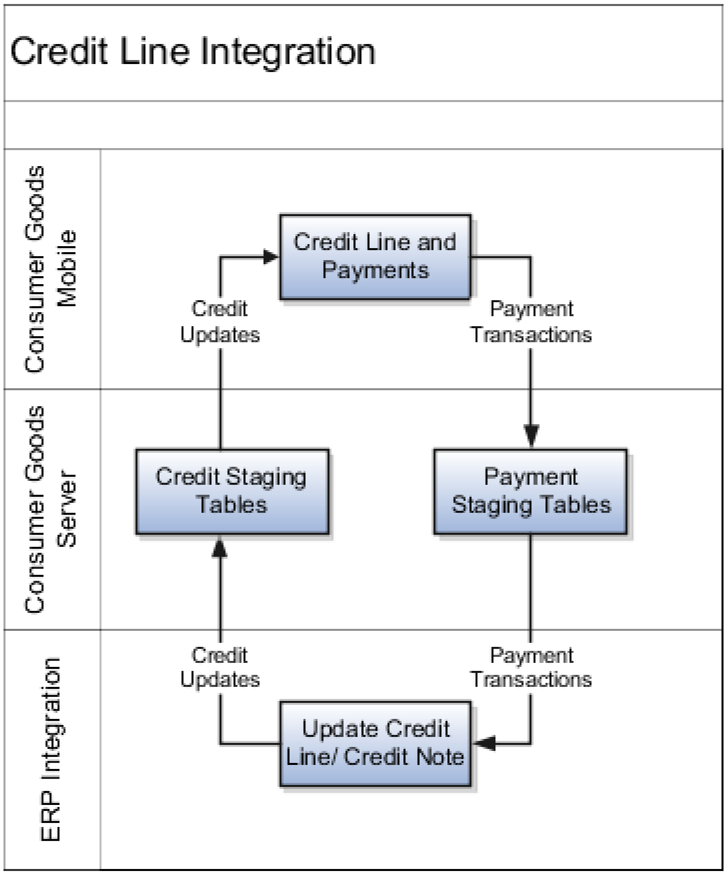
When integrated, you can use CX Retail Execution Mobile to see the usage of credit lines or credit notes available to each customer. You can capture and manage payments done using these credit lines or credit notes against the invoices.

You can use credit lines to store records of type credit line and proof of purchase, and to verify the authorization of proof of purchase. See: Integration Tables and Key Fields for the fields used for the type of record and the authorization of proof of purchase.

Here's how the integration flow works:

1. Integration system uploads the credit line information to the consumer goods server.
2. CX Retail Execution Mobile captures payment transactions against these credit lines.
3. The payment transaction records are transferred to the staging tables.
4. The integrated system picks up these transaction records and updates the credit line information based on payments.

The following graphic shows the credit line integration flow.



# Configure Integration Flows for Credit Lines

As sales administrator or implementor, you can integrate system to manage the balances in credit notes and credit lines. It also covers how you must configure the integration to refresh the credit records in the integration tables periodically.

> **Note:** If you don't use credit notes, you can skip the integration of credit note and integrate only credit line.

The credit line and credit note records that the CX Retail Execution Mobile uses are read-only. Although the credit balances are updated in the mobile application to facilitate route execution, these balances are not sent to the consumer goods server. Only the payment transaction records are sent to the consumer goods server. Instead of using credit line balances, the integrating system must use the captured payment transaction records to update the credit balances.

The integration system must do the following to maintain credit details:

- Add and update records for open credit lines and credit notes.
- Remove records for closed credit lines and credit notes.

Link a credit note to its parent credit line to create the credit note records in consumer goods, you must perform the following tasks:

- Create the parent credit line record first.

- Read the Internal ID of the credit line record and use it to create the child credit note records so that the records are properly linked.

# Tables and Key Fields for Credit Lines

As a sales administrator, you can use the tables and the key fields to support the integration.

To facilitate the data flow between multiple systems, this integration uses the following staging tables to store credit line and credit note information:

- DSD Credit Line
- DSD Credit Note

## DSD Credit Line

Use the following table and the key fields in the DSD Credit Line `(__ORACO__CreditLineDSD_c)` staging table.

| Key Fields | API Name | Description |
|---|---|---|
| Credit Line Number | `RecordName` | An external ERP data source issues the Credit Line Number. |
| Source System<br>Source System Number | `__ORACO__CreditLineSS_c`<br>`__ORACO__CreditLineSSN_c` | The user keys for a credit line. |
| Account | `__ORACO__Account_c`<br>`(__ORACO__Account_Id_c)` | The account for which the credit line was approved. |
| Credit Limit | `__ORACO__CreditLimit_c` | The maximum amount the customer can use with this credit line. |
| Number of Notes Allowed | `__ORACO__NoOfNotesAllowed_c` | The number of credit notes allowed with the credit line. |
| Amount Used | `__ORACO__AmountUsed_c` | The amount the customer has used with this credit line. |
| Notes in Use | `__ORACO__NotesInUse_c` | The number of credit notes opened with the credit line. |
| Available Credit | `__ORACO__AvailableCredit_c` | The available amount the customer can use with the credit line. |
| Balance Date | `__ORACO__BalanceDate_c` | The date on which the balance is updated. |

| Key Fields | API Name | Description |
|---|---|---|
| Amount Due | `__ORACO__AmountDue_c` | The amount due with the credit line. |
| Legal Entity | `__ORACO__LegalEntity_c`<br>`(__ORACO__LegalEntity_Id_c)` | The legal entity for the credit line. |
| Type | | Indicates the type of the credit line record. The type can be standard credit line or proof of purchase. |
| Approved From Date | | The date from which the record is valid. |
| Approved To Date | | The date up to which the record is valid. |

## DSD Credit Note

Use the following table and the key fields in the DSD Credit Note `(__ORACO__CreditNoteDSD_c)` staging table.

| Key Fields | API Name | Description |
|---|---|---|
| Note Number | `RecordName` | An external ERP data source issues the note number. |
| Account | `__ORACO__Account_c`<br>`(__ORACO__Account_Id_c)` | The account for which the credit note was created. |
| Credit Line | `__ORACO__CreditLine_c`<br>`(__ORACO__CreditLine_Id_c)` | The parent credit line. |
| Source System<br>Source System Number<br>Line Number | `__ORACO__CreditNoteSS_c`<br>`__ORACO__CreditNoteSSN_c`<br>`__ORACO__CreditNoteLN_c` | The user keys for a credit note. |
| Order Number<br>Order - Source System Number<br>Order Source System<br>Order - Revision Number | `__ORACO__OrderNumber_c`<br>`__ORACO__OrderSSN_c`<br>`__ORACO__OrderSS_c`<br>`__ORACO__OrderRN_c` | Identifies the order for which the credit note was opened. |
| Issue Date | `__ORACO__IssueDate_c` | The date when the credit note was opened. |
| Due Date | `__ORACO__DueDate_c` | The date when the credit note is due. |

| Key Fields | API Name | Description |
|---|---|---|
| | | |
| Amount | `__ORACO__Account_c`<br>`(__ORACO__Account_Id_c)` | The amount used with the credit note. |
| Balance | `__ORACO__Balance_c` | The outstanding amount from the credit note. |
| Balance Date | `__ORACO__BalanceDate_c` | The date on which the balance is updated. |
| Legal Entity | `__ORACO__LegalEntity_c`<br>`(__ORACO__LegalEntity_Id_c)` | The legal entity for the credit note. |

# 9 Integrate Payments

## Integrate Payments

You can integrate CX Retail Execution Mobile with an Enterprise Resource Planning (ERP) accounting system to manage payment information.

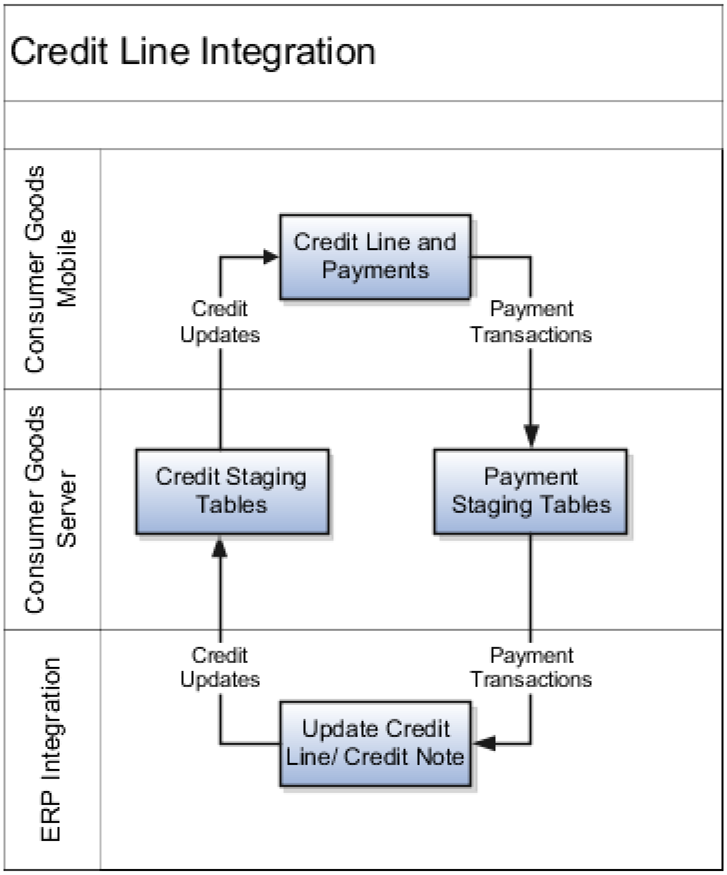
In CX Retail Execution Mobile, sales representatives can create payment transactions against credit lines. To process these payment transactions further and to update the credit lines after processing the payments, you must integrate an ERP system with CX Retail Execution Mobile.

When integrated, you can use CX Retail Execution Mobile to capture payments and view updated credit lines or credit notes based on those payments.

Here's how the integration flow works:

1. CX Retail Execution Mobile captures payment transactions against credit lines.
2. The payment transaction records are transferred to the staging tables.
3. Integration system processes the payment transactions and uploads the credit line information to the consumer goods server.
4. CX Retail Execution Mobile picks up the updated credit line information.

The following graphic shows the payments integration flow.



# Integrate Payment Types

This integration supports the following types of payments:

- Cash: Payments made using physical currency.
- Check: Payments made using checks.
- Credit line: Payments made using credit lines. ERP issues and manages the credit lines to pay for items up to a maximum credit limit for a fixed number of times. Customers must later pay for these credit lines using cash.
- Credit or Debit cards: Payments made using credit or debit cards. A credit card lets your customers pay for your goods and services by incurring a debt with a credit card provider.
- Electronic transfer: Payments made using Websites or portals.
- Proof of purchase: Payments made using proof of purchase. This proof of purchase has a unique number that must be captured in this integration.
- Coupon: Payments made using coupons. A coupon has a prepaid amount that customers can use.

You must also provide values associated with each type of payment to process each payment transaction. For example, if your customer uses checks for payments, capture the values for fields such as Bank Name, Bank Account Number, and Check Number. Similarly, specify the following fields for each payment type:

- Coupon: Coupon and Number of Coupons.

- Credit Line: Credit Line Number, Credit Line - Source System and Credit Line - Source System Number.
- Credit Card: Voucher Number.
- Electronic Transfer: Electronic Transfer Transaction Number.
- Proof of Purchase: Proof of Purchase Number and Purchase Order Number.

# Configure Integration for Payments

As an administrator, you can manage the integrating system and the balance of payments in credit notes or credit lines. Remember to configure the integration to refresh the payments in the integration tables periodically.

The credit line and credit note records that the consumer goods mobile application uses are read-only. Although the credit balances are updated in the mobile application to facilitate route execution, these balances aren't sent to the consumer goods server. Only the payment transaction records are sent to the consumer goods server. Instead of using credit line balances, the integrating system must use the captured payment transaction records to update the credit balances.

## Process Payments

The sales representatives who collect cash and checks must deliver them to the back-office system regardless of any issues (if any) in other payment methods. For example, an issue with a coupon must not block uptake of a payment in cash.

The back-office system must process payments one by one, which means that the payment lines are processed individually over iterations. The integrating ERP system must process only those payment lines where the **Process Status** is **Ready to process**. Optionally, it can also process the corresponding payment headers.

After processing a payment line, ensure that the **Process Status** field is set to **Processed** to prevent reprocessing of the payment line records.

# Tables and Key Fields for Payments

As an administrator or implementor, use or configure these tables and key fields to support the integration.

The fulfillment of a payment requires data flow between multiple systems. This integration uses the following staging tables to store payment information:

- Coupon
- DSD Payment
- DSD Payment Line

## Coupon

Here are the key fields in the Coupon (__ORACO__Coupon_c) table. This table captures and stores all the information about coupons, such as the value, currency, and validity.

| Key Fields | API Name | Description |
|---|---|---|
| Coupon Name | `RecordName` | The unique name of the coupon. |
| Start Date | `__ORACO__StartDate_c` | The date when the validity period of a coupon begins. |
| End Date | `__ORACO__EndDate_c` | The date when the validity period of a coupon ends. |
| Coupon Value | `__ORACO__CouponValue_c` | The value of the coupon. |
| Currency Code | `__ORACO__Currency_c` | The currency of the coupon. |

## DSD Payment

Here are the key fields in the DSD Payment (`__ORACO__PaymentDSD_c`) table. This table captures and stores information about payments.

| Key Fields | API Name | Description |
|---|---|---|
| Payment Number | `RecordName` | The unique payment reference number. |
| Account | `__ORACO__Account_c` | The corresponding retailer account which has made the payment. |
| Payment Date | `__ORACO__PaymentDate_c` | The date of payment. |
| Amount | `__ORACO__Amount_c` | The amount for each payment. |
| Source System<br>Source System Number | `__ORACO__PaymentSS_c`<br>`__ORACO__PaymentSSN_c` | A unique combination for identifying payments.<br>• Source System: A unique identifier that the mobile device issues. This must be unique across different originating systems.<br>• Source System Number: A unique number within each source system. |

## DSD Line Payment

Here are the key fields in the DSD Payment Line (`__ORACO__PaymentLineDSD_c`) table. This table captures and stores the information about payment lines.

| Key Fields | API Name | Description |
|---|---|---|
| Payment Line Number | `RecordName` | The globally unique alphanumeric string generated in mobile for each payment line. |
| Account | `__ORACO__Account_c` | The corresponding retailer account which has made the payment. |
| Payment | `__ORACO__Payment_c` | The payment header information. |
| Amount | `__ORACO__Amount_c` | The amount for each payment line. |
| Pay For | `__ORACO__PayFor_c` | The payment type against which the payment is made. |
| Pay For Credit Line | `__ORACO__PayToCreditLine_c` | The value used when the payment is made using a credit line. You must enter the Credit Line Number, Credit Line - Source System, and Credit Line - Source System Number. |
| Pay For Credit Note | `__ORACO__PayToCreditNote_c` | The value used when the payment is made using a credit note. You must enter the Credit Note Number, Credit Note - Source System, Credit Note - Source System Number, and Credit Note - Line Number. |
| Pay For Invoice | `__ORACO__PayToInvoice_c` | The value used when the payment is made using an invoice. You must enter the Invoice Number, Invoice - Source System, and Invoice - Source System Number. |
| Type | `__ORACO__Type_c` | The type of payment method for the payment line. |
| Source System<br>Source System Number<br>Line Number | `__ORACO__PaymentSS_c`<br>`__ORACO__PaymentSSN_c`<br>`__ORACO__PaymentLineLN_c` | The user keys for a payment. |

# 10 Integrate Store Credits

## Integrate Store Credits

As sales administrators or implementors, you can integrate consumer goods with your Enterprise Resource Planning (ERP) accounting system to manage store credit information.

As salesperson, use the mobile to capture the product returns when they visit retail stores. After accepting the returns, the retail store can get the credit for these returns and they can use it for paying the outstanding order invoices.

To refresh the store credit data based on the return and other sales transactions, you must integrate the mobile application with an external the ERP.

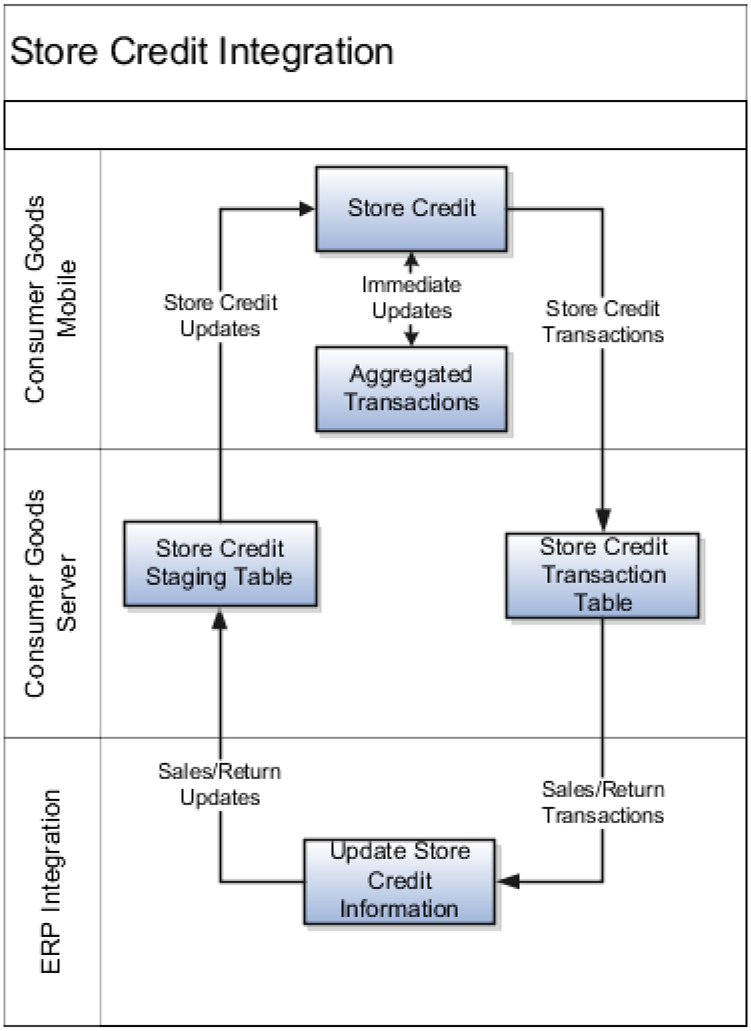
The store credit transactions are read-only in consumer goods server and must be available to the integrating the ERP for further processing. You can define the processing logic by setting the profile option.

When integrated, you can use the mobile application to manage store credits based on sales and return orders.

Here's how the integration flow works:

1. The mobile application captures store credit transactions for each account. You can integrate these transactions with your ERP for further processing based on your processing logic.
2. The Integrating system processes the store credit transactions and uploads the information to the consumer goods server.
3. The mobile application selects the updated or refreshed store credit information. The following graphic shows the store credit integration flow.

The following graphic shows the store credit integration flow.



# Configure Integration Flows for Store Credits

Understand how you must configure the integration to refresh the store credit in the integration tables periodically.

## Set Up Profile Option

The integration supports two strategies for the mobile application to manage the store credits based on the profile option enabled.

| Profile Option | Description |
|---|---|
| Immediate | Makes the store credits allocated to an account immediately available in the mobile application for redemption when a product is returned.<br>Mobile application aggregates the transactions to create a local copy of the store credits. Enterprise Resource Planning (ERP) doesn't validate the transactions. |

| Profile Option | Description |
|---|---|
| | |
| Back Office | Prevents the store credits allotted to an account from being redeemed immediately using the mobile when a product is returned.<br>ERP validates the transactions along with the payment and order details before generating the store credits for the account. These credits are a part of the download package and can be used by the account after they're synced to mobile. |

## Process Store Credits

Here's how the mobile application processes the store credit information for an account:

1. Reads the accounts base store-credit and the **Last Processed Transaction Time Stamp** from the Store Credit staging table.
2. Reads all transactions that are greater than the **Last Processed Transaction Time Stamp**.
3. Adds or reduces the transaction quantity for each transaction from the original store credit value, as applicable.
4. Generates a new record in store credit staging table with the final value. This new record has its **Last Processed Transaction Time** set to the creation time of the most recent transaction for the account.

# Integrate Tables and Key Fields for Store Credits

As an administrator or implementor, you use or configure these tables and the key fields to support the integration.

To facilitate the data flow between multiple systems, this integration uses the following staging tables for store credit information:

- Store Credits
- Store Credits Transaction Table

## Store Credits

Here are the key fields in the store credits `(__ORACO__StoreCredit_c)` table.

| Key Fields | API Name | Description |
|---|---|---|
| Store Credit Number | `RecordName` | The unique name for store credits. |
| Amount | `__ORACO__Amount_c` | The amount available as credit for the account. |
| Account | `__ORACO__Account_c` | The account to which the store credit is allotted. This must be set by the integrating ERP system. |
| Date | `__ORACO__Date_c` | The date on which the store credit is calculated and allotted. This must be set by the integrating ERP system. |

| Key Fields | API Name | Description |
|---|---|---|
| | | |
| Legal Entity | `__ORACO_LegalEntity_c` | The legal entity for which the store credit is allotted. |

## Store Credit Transactions Table

Here are the key fields in the store credit transactions (`__ORACO__StoreCreditTrans_c`) table.

| Key Fields | API Name | Description |
|---|---|---|
| Store Credit Transaction Number | `RecordName` | The unique identifier for the store credit transaction. |
| Transaction Date | `__ORACO__TransactionDate_c` | The date on which the transaction was recorded. |
| Transaction Amount | `__ORACO__TransactionAmount_c` | The amount involved in the transaction. |
| Transaction Type | `__ORACO__TransactionType_c` | The type of the transaction. This can be debit or credit. |
| Description | `__ORACO__Description_c` | The description of the transaction. |
| Store Credit | `__ORACO__StoreCredit_c` | The ID of the store credit related to the transaction. |
| Account | `__ORACO__Account_c` | The account where the credit transaction has happened. |
| Order | `__ORACO__Order_c` | The return order or sales order which created the transaction. |
| Source System<br>Source System Number | `__ORACO__StoreCreditTransSS_c`<br>`__ORACO__StoreCreditTransSSN_c` | The unique user keys for store credits.<br>• Source System is the route number.<br>• Source System Number is the unique number generated in the mobile application for each route. |
| Order Source System | `__ORACO__OrderSS_c` | The source system of the order. |
| Order Source System Number | `__ORACO__OrderSSN_c` | The source system number of the order. |
| Order Revision Number | `__ORACO__OrderRN_c` | The revision number of the order. |

| Key Fields | API Name | Description |
|---|---|---|
| Order Number | `__ORACO__OrderNumber_c` | The number of the order. |
| Legal Entity | `__ORACO__LegalEntity_c` | The legal entity of the order for which the transaction was performed. |

# 11 Integrate Empty Containers

## Integrate Empty Containers

As a sales administrator, you can integrate CX Consumer Goods with an ERP system to manage empty containers.

As sales representatives, you can manage empty containers in both offline and online mode. The empty containers refer to the returnable containers or material such as bottles or cartons that the retail stores must return after the delivery of the products.

The mobile application captures the delivery or return of containers as transactions, and uses the integration with the ERP to refresh or recalculate the balance of these containers before each route execution.

The transactions data is read-only and determines the charge on the empty containers which are not returned.

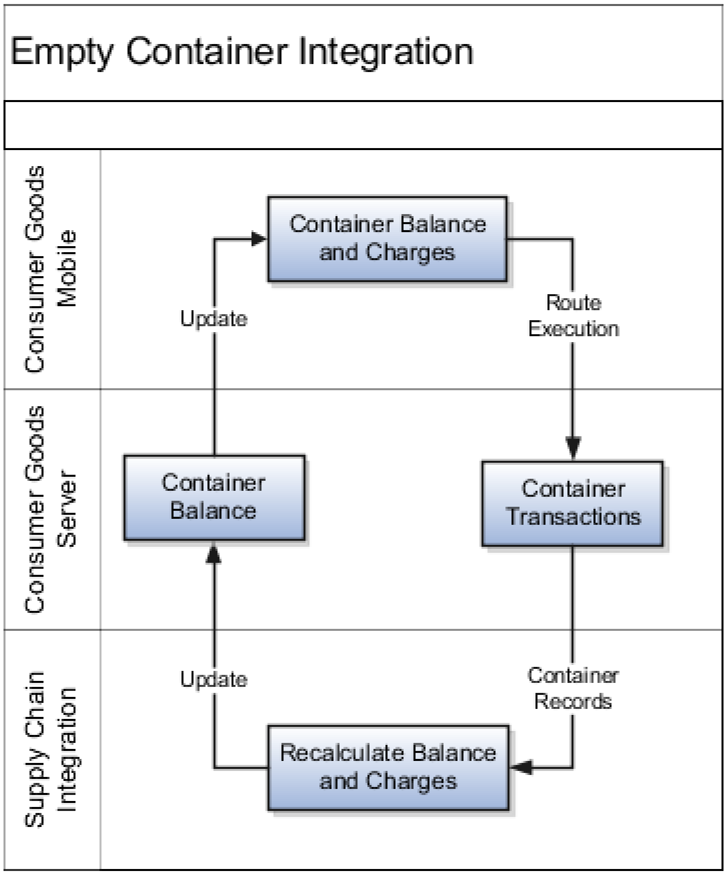
The transactions are of the following types:

1. Product Delivery: Increases the number of containers loaned to the customer based on the quantity delivered in a shipment to the customer. This does not include any containers purchased by the customer.
2. Return of Empty Containers: Decreases the number of containers loaned to a customer based on the number of the empty containers returned by the customer.
3. Return of Product: Decreases the number of containers with a customer based on the quantity of the products returned by the customer.

Here's how the integration flow works:

1. The ERP system recalculates the balance and sends the new balance records to the consumer goods server. See, Calculating Balance Using Transactions Method.
2. The CX Retail Execution Mobile retrieves the refreshed balance records from the consumer goods server.
3. The CX Retail Execution Mobile sends transaction records created during route execution to the consumer goods server.
4. The consumer goods server prepares the transactions records and makes them available to the integrating ERP system.

This graphic shows the empty container integration flow:



# Configure Integration Flows for Empty Containers

Manage and configure the empty containers transactions to transfer data to or from the sales application to your integrated application. You can learn about the methods or strategies to update the container-class balance records.

The mobile application uses only the container-class transactions that are periodically purged using the table designed to support this operation.

> **CAUTION:** If the transactions table is not used, you must configure the integration in a way that it does not purge the transactions. This can affect the recalculation of the inventory level in the device.

## Calculate Balance Using Transactions Method

Here's how the integration calculates the balance using the mobile transactions:

1. Reads the current balance for an account and container class combination and notes the **Last Transaction Processed Time Stamp** of that container class.

2. Retrieves and processes the container class transactions, where the **Creation Time Stamp** is later than the **Last Transaction Processed Time Stamp**.

   Configure the integration to process these retrieved transactions by descending order of the **Creation Time Stamp** so that the most recent transaction is processed first. To prevent the integration from processing in-progress transactions based on the most recent **Creation Time Stamp**, you can do one of the following configurations:

   - Induce delay in processing the transaction record having the most recent **Creation Time Stamp**.
   - Process transaction records up to but not including the most recent **Creation Time Stamp**.
3. After the container-class balance is updated, the **Last Transaction Processed Time Stamp** is replaced with the **Creation Time Stamp** of the last processed record. You must ensure that the same operation updates both these fields.

You can configure the integrating system to reconcile the records multiple times in a day.

# Tables and Key Fields for Empty Containers

As an administrator or implementor, you use or configure these tables and key fields to support the integration.

To facilitate the data flow between multiple systems, this integration uses the following staging tables to store container information:

- Container Class Balance
- Container Class Transactions

## Container Class Balance

Here are the key fields in the Container Class Balance (`__ORACO__ContainerClassBal_c`) staging table. This table contains information about the empty containers that have not been returned by the customer at the start of the day or as configured.

| Key Fields | API Name | Description |
|---|---|---|
| Account<br>Container Class | `__ORACO__Account_c`<br>`__ORACO__ContainerClass_c` | The fields that correspond to an account and container class combination. One container class transaction corresponds to a combination of account and container class. |
| Current Balance | `__ORACO__CurrentBalance_c` | The balance or the current count of the empty containers that have not been returned. |
| Balance Counted On | `__ORACO__BalanceCountedOn_c` | The date and time when the container balance was last counted. |
| Last Transaction Processed Time Stamp | `__ORACO__LastTxnProcessedTS_c` | The last creation Time Stamp from the Container Class Transactions table. |

## Container Class Transactions

Here are the key fields in the Container Class Transaction (__ORACO__ContainerClassTxn_c) table. This table captures and stores the transaction records of the containers.

| Key Fields | API Name | Descriptions |
|---|---|---|
| Record ID | Id | A unique and incremental database sequence identifier generated by consumer goods server when a record is created. |
| Creation Date | CreationDate | The date when the transaction occurs. |
| Account<br>Container Class | __ORACO__Account_c<br>__ORACO__ContainerClass_c | The fields that correspond to an account and container class combination. One container class transaction corresponds to a combination of account and container class. |
| Transaction Date<br>Transaction Type<br>Quantity | __ORACO__TransactionDate_c<br>__ORACO__TransactionType_c<br>__ORACO__Quantity_c | A combination of fields that represent how the container class balance has changed. |