Accelerators

Generated on: 2020-12-08 14:44:06 GMT+0000

SAP Commerce | 1905

PUBLIC

Original content: https://help.sap.com/viewer/4c33bf189ab9409e84e589295c36d96e/1905/en-US

Commerce B2B Accelerator

Commerce B2B Accelerator is a ready-to-use Web framework that enables you to jump-start your B2B implementation and easily build and maintain a feature-rich, omni-channel commerce solution.

B2B Accelerator Module

B2B Accelerator includes features that manage your inventory, security, AddOns, and user accounts.

B2B Accelerator AddOns Module

B2B Accelerator AddOns allow you to manage your accounts, order forms, login security, and punchout procurements.

B2B Commerce Module

B2B Commerce Module adds business-to-business functionality to SAP Commerce. It enables you to integrate multiple channels, business models, and markets on a single platform.

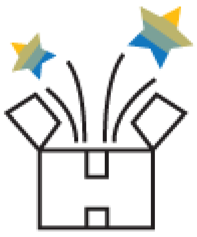
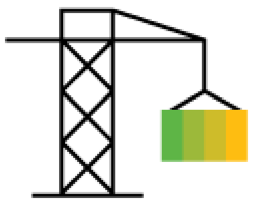
B2B Commerce Module

B2B Commerce Module adds business-to-business functionality to SAP Commerce. It enables you to integrate multiple channels, business models, and markets on a single platform.

B2B Commerce Module offers a retail-like shopping experience to your business customers, featuring integrated order process handling. Various permission settings enable you to efficiently and securely process your sales. You can integrate various products, multiple supplier catalogs, inventory, contracts, customer information, content, and orders onto a single solution.

Note

A SAP Commerce module may include or enable functionality that is not covered by your individual license. Make sure to limit your implementation to features as defined in your license contract. In case of doubt, please contact yourSAP Sales representative.

Features	Architecture
	
B2B Accelerator and the Backoffice Administration Cockpit	b2bapprovalprocess Extension
B2B Checkout and Order Process	b2bapprovalprocessfacades Extension
Backoffice Integration	b2bcommerce Extension
Catalogs and Pricing	b2bcommercefacades Extension
Online Ordering	b2bcommercebackoffice
Configuring Sales Organization in Backoffice	

B2B Commerce Features

SAP Commerce B2B Commerce Module provides B2B organizations with a flexible multi-channel solution that can easily be customized to match specific requirements of the B2B purchasing cycle. Multiple business models, channels, and markets can be managed on a single platform.

By combining standard SAP Commerce functionality and B2B-specific features, a sophisticated retail-like shopping experience can be created in service of business customers. The SAP Commerce B2B Commerce Module offers additional administration tools, customer self-service tools, and functionality for the storefront and backend. All sales administration tasks can be automated, reducing operational costs and eliminating manual, low-value processes. Salespeople can then focus on value selling.

The basic building block of B2B functionality in SAP Commerce is support for customer organizations, known as units. A unit represents a division, department, office, factory, or any other grouping that may exist for a customer's company. Typically one primary parent unit is created for a company, and then other child units are assigned to this parent unit.

Challenges Faced by B2B Customers

B2B organizations manage a complex ecosystem of suppliers, distributors, and partners with different terms and contracts. Customized pricing and catalogs are required. It is challenging to accurately and effectively represent complex and highly specialized B2B products across a wide range of channels, all with different display capabilities. Customer relationships and transactions can also be complex.

In addition, there are different types of users with different needs. Sales and service resources are required for taking orders and handling order problems. Salespeople must perform manual tasks like managing customer information, ordering using phone or fax, and following up on order approval processes. Ordering processes are complex because orders must pass through approval processes. Account managers face similar administrative overhead.

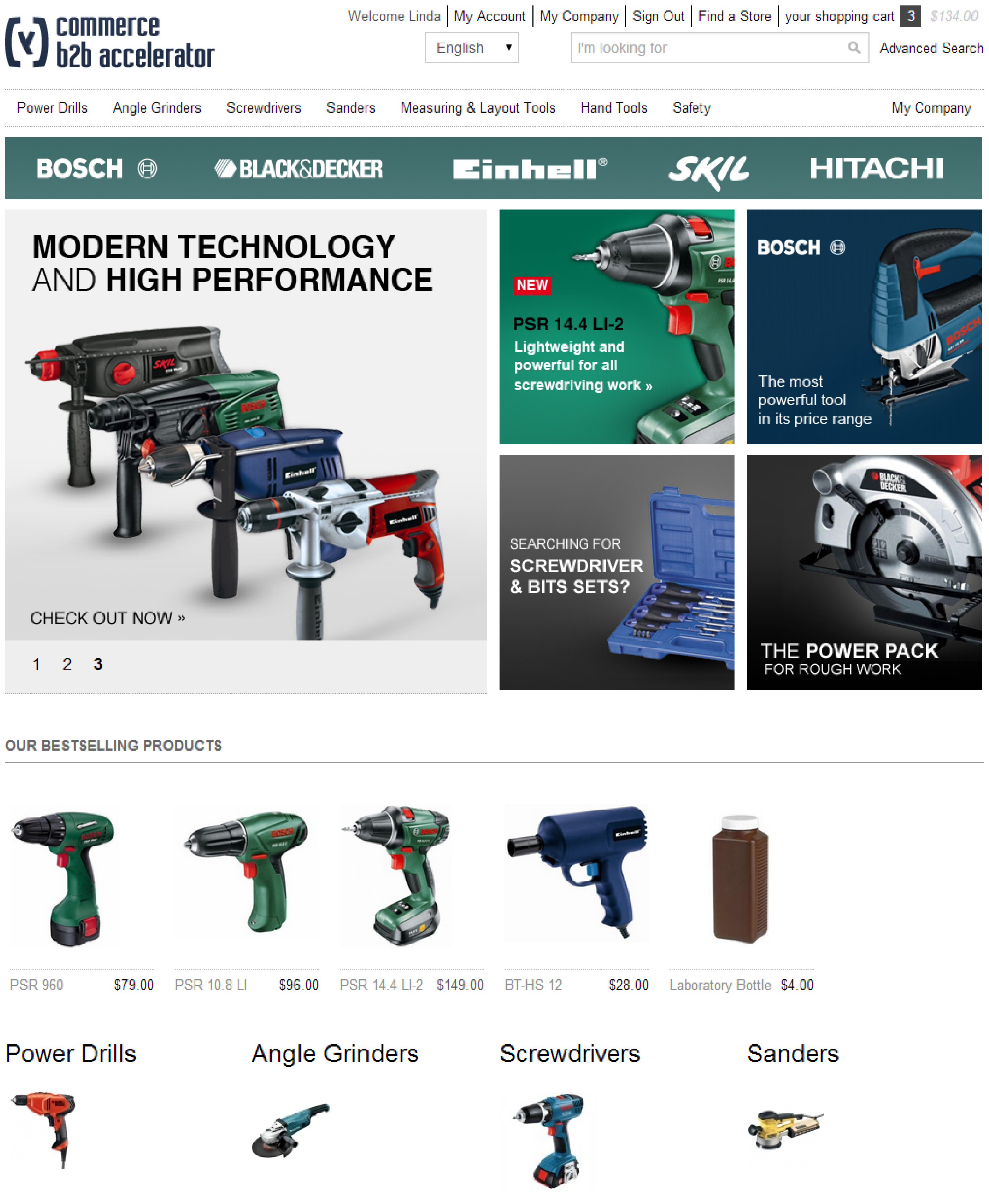
Key Features

The B2B Commerce Module extends the current SAP Commerce B2C functionality with the extra functionality required in a B2B context.

All features are manageable using the SAP Commerce Administration Cockpit. Customers use the site itself to manage their organizations.

Powertools Storefront Example

The SAP Commerce includes a fully functional B2B storefront example, named Powertools, that is used to help jump-start the customer implementation and easily build and maintain a feature-rich multi-channel B2B commerce solution.



[B2B Accelerator and the Backoffice Administration Cockpit]

Learn about the Backoffice Administration Cockpit as it relates to Commerce B2B Accelerator.

[B2B Checkout and Order Process]

The B2B checkout and order process allows a registered customer to make a one-time purchase, schedule a replenishment order, or request a quote.

[Backoffice Integration]

Integrating the Backoffice features results in a seamless customer experience by bridging the gap between your online e-commerce system with your accounts and database systems.

Catalogs and Pricing

You can link the pricing and product catalogs to specific units based on the logged-in purchaser.

Online Ordering

B2B-specific checkout features offere more than standard features like being able to create one-time orders or re-ordering from existing orders.

Self-service Account Management

Each customer manages their own organization through the B2B site.

Sales Organization

The Sales Organization feature allows you to manage sales organizations. You can use it to create and modify sales organizations, and it also allows you to assign sales representatives to customers.

Configuring Sales Organization in Backoffice

In Backoffice, the Sales Organization perspective allows you to create and modify sales units, add sales employees, modify employee details, and manage the relationships between sales units and customers.

B2B Accelerator and the Backoffice Administration Cockpit

Learn about the Backoffice Administration Cockpit as it relates to Commerce B2B Accelerator.

Use this URL to access the Backoffice Administration Cockpit: `https://powertools.local:9002/backoffice/`

Note

For these links to work, install SAP Commerce locally using a B2B recipe, and set `powertools.local` to `localhost` in your computer's `hosts` file.

Backoffice is a flexible framework designed to facilitate the creation of business tools and user interfaces in an easy and consistent way. It can be used to configure SAP Commerce

Finding Members of a B2B User Group

To display the members of a user group in B2B Backoffice Administration Cockpit:

1. Click **B2B Commerce B2B User Group**. The list of user groups appears.
2. Click a user group. The details for that user group appear.
3. Click the **Administration** tab. The users are shown in the **Members** list box.

You can also use this list box to search for members.

Related Information

Backoffice & Backoffice Product Content Management
Backoffice Administration Cockpit - User Guide

B2B Checkout and Order Process

The B2B checkout and order process allows a registered customer to make a one-time purchase, schedule a replenishment order, or request a quote.

Introduction to the Checkout and Order Process

On the Final Review step of the B2B checkout flow, customers indicate how the items in their cart should be processed. They can do one of the following:

- Place an order now (one-time purchase)

- Schedule automatic replenishment of what is in the cart

i Note

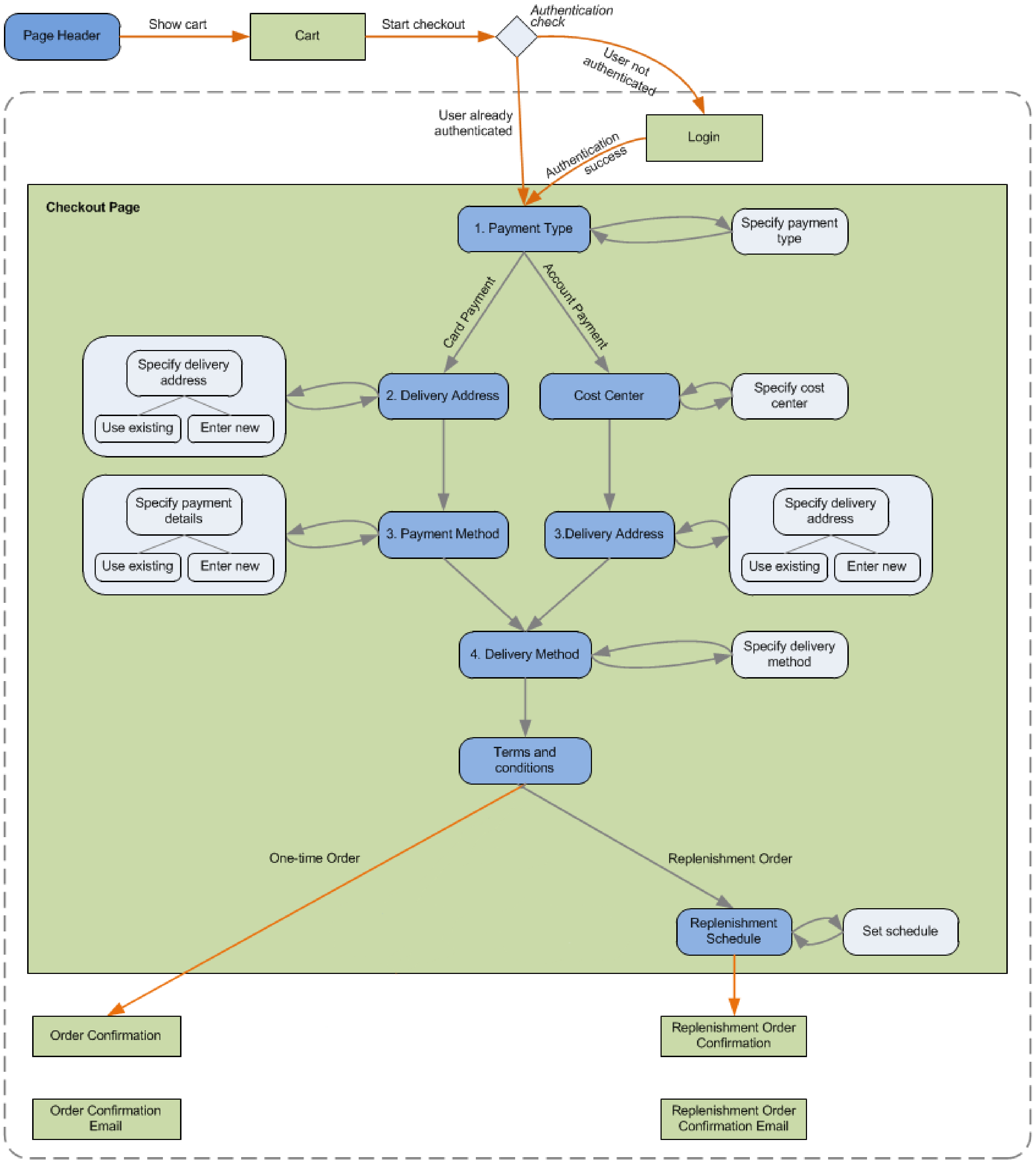
Customers can also request quotes from their cart. For more information, see Commerce Quotes.

Process Overview

To make a one-time purchase or schedule a replenishment order, customers do the following:

1. Display the cart and click **Checkout**, logging in if required.
2. Provide **Payment Type** information.
3. Provide **Shipping Address** information.
4. Provide **Shipping Method** information.
5. Select the **Terms and Conditions** check box.
6. Perform one of the following actions, depending on the type of order:
    - One-time order: Click **Place Order**.
    - Replenishment order: Click **Schedule Replenishment**, provide scheduling information, and click **Place Replenishment Order**.

The following diagram illustrates the page flow for this process.



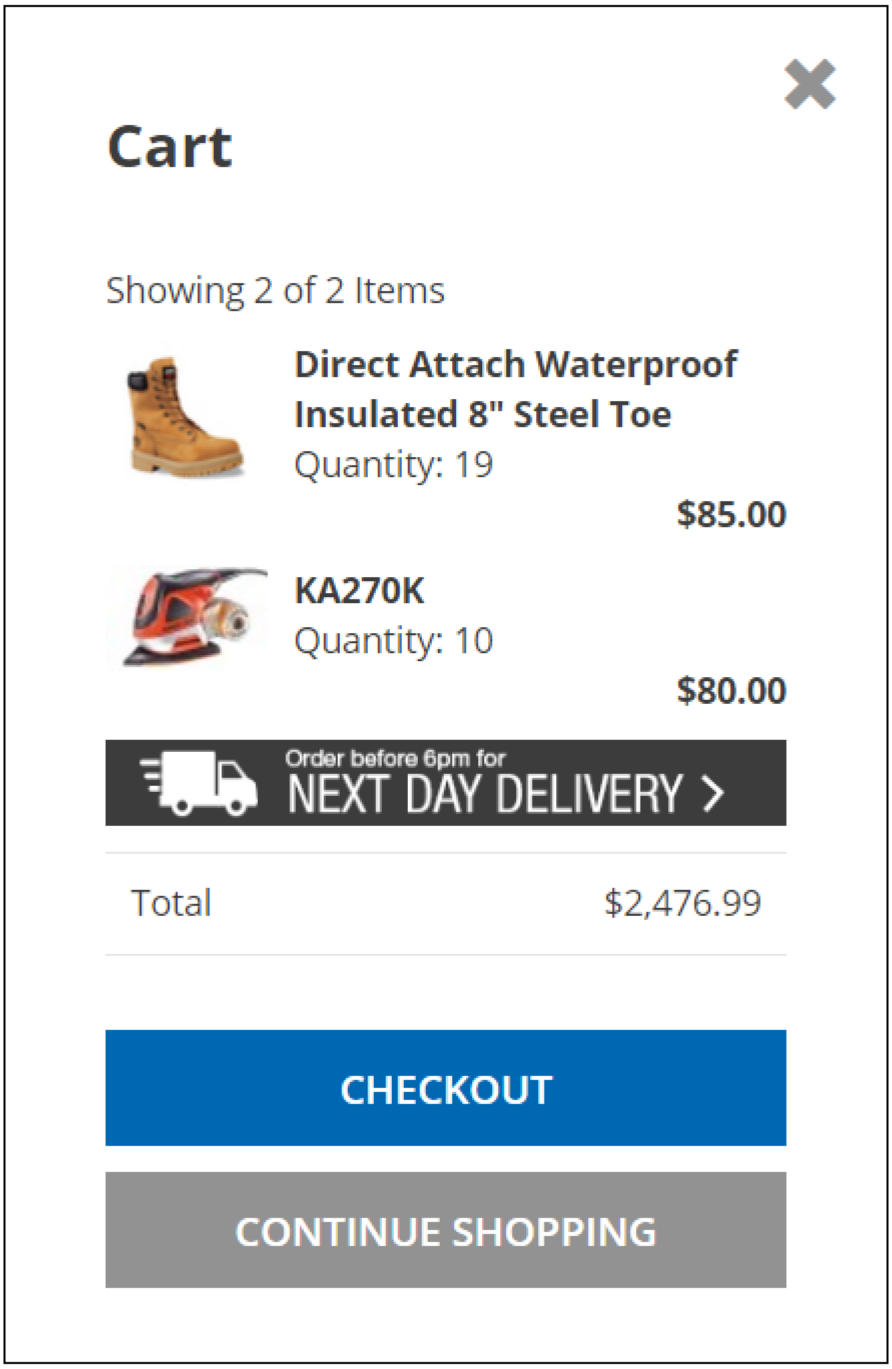
Starting the Checkout and Order Process

The B2B Checkout and Order Process is started when customers click the **Checkout** button from the **Cart** page, or if they are directed to initiate the checkout process after re-ordering from your order history.

- If customers are already logged in, the **Payment Type** page is displayed.
- If customers must log in, the **Return Customer** page is displayed, which allows them to log in. Customers must be registered with the site to make a purchase.

Mini Cart

Customers can display the mini cart by clicking the cart icon in the top-right corner of the storefront page. The mini cart displays a summary of all the items in the cart and allows customers to either proceed to checkout or continue their shopping.



Cart

Showing 2 of 2 Items

Direct Attach Waterproof Insulated 8" Steel Toe
Quantity: 19
$85.00

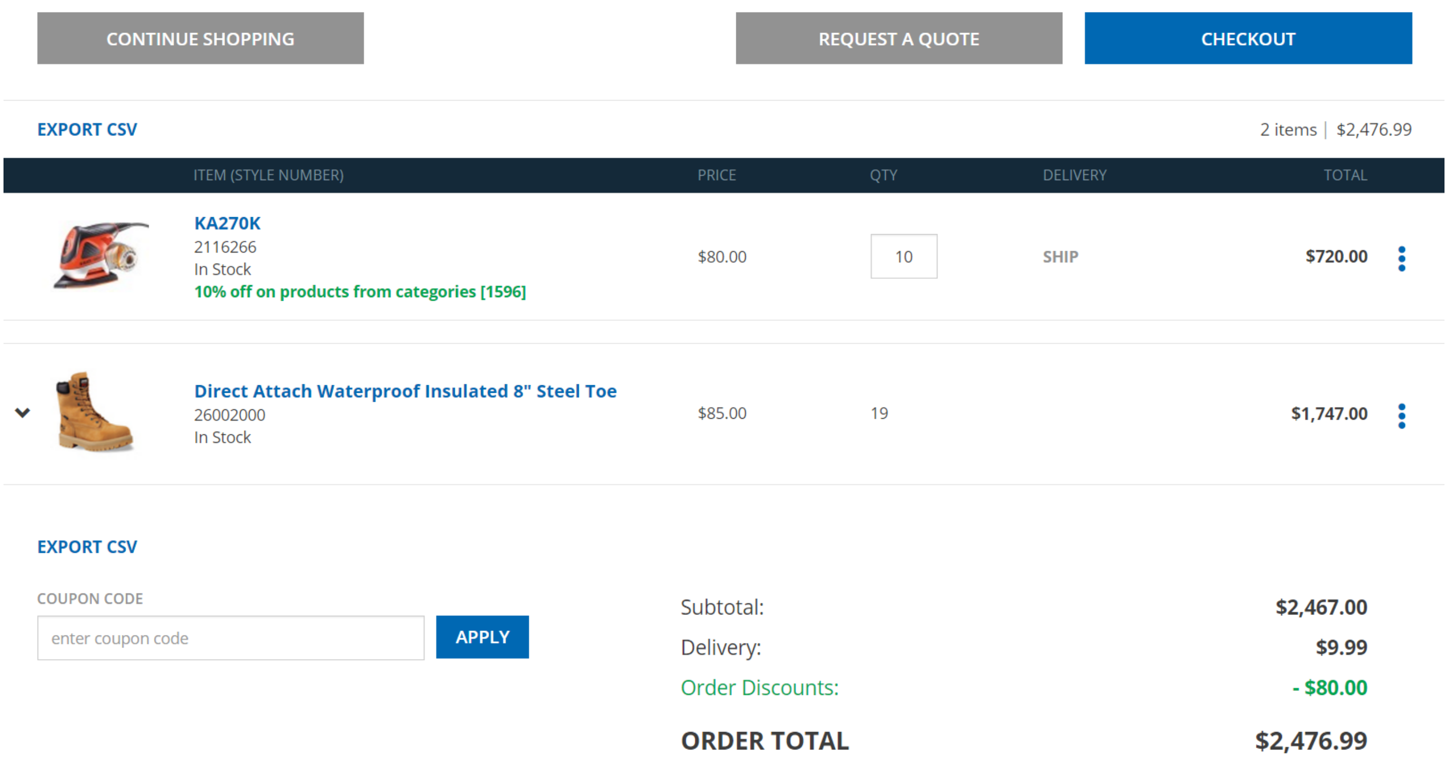
KA270K
Quantity: 10
$80.00

Total $2,476.99

Cart Pages

To display the cart, customers click **Checkout** in the mini cart.

In the following cart example, the boot is a multi-dimensional product. Multi-dimensional products are grouped into one line in the cart. For example, three pairs of size 9 boots, five pairs of size 10.5 boots, and 11 pairs of size 12 boots, all variants of the same base product, would appear as 19 boots in one line, as shown in the example below.



CONTINUE SHOPPING | REQUEST A QUOTE | CHECKOUT

EXPORT CSV — 2 items | $2,476.99

	ITEM (STYLE NUMBER)	PRICE	QTY	DELIVERY	TOTAL
	KA270K 2116266 In Stock **10% off on products from categories [1596]**	$80.00	10	SHIP	**$720.00**
	Direct Attach Waterproof Insulated 8" Steel Toe 26002000 In Stock	$85.00	19		**$1,747.00**

EXPORT CSV

COUPON CODE

enter coupon code — APPLY

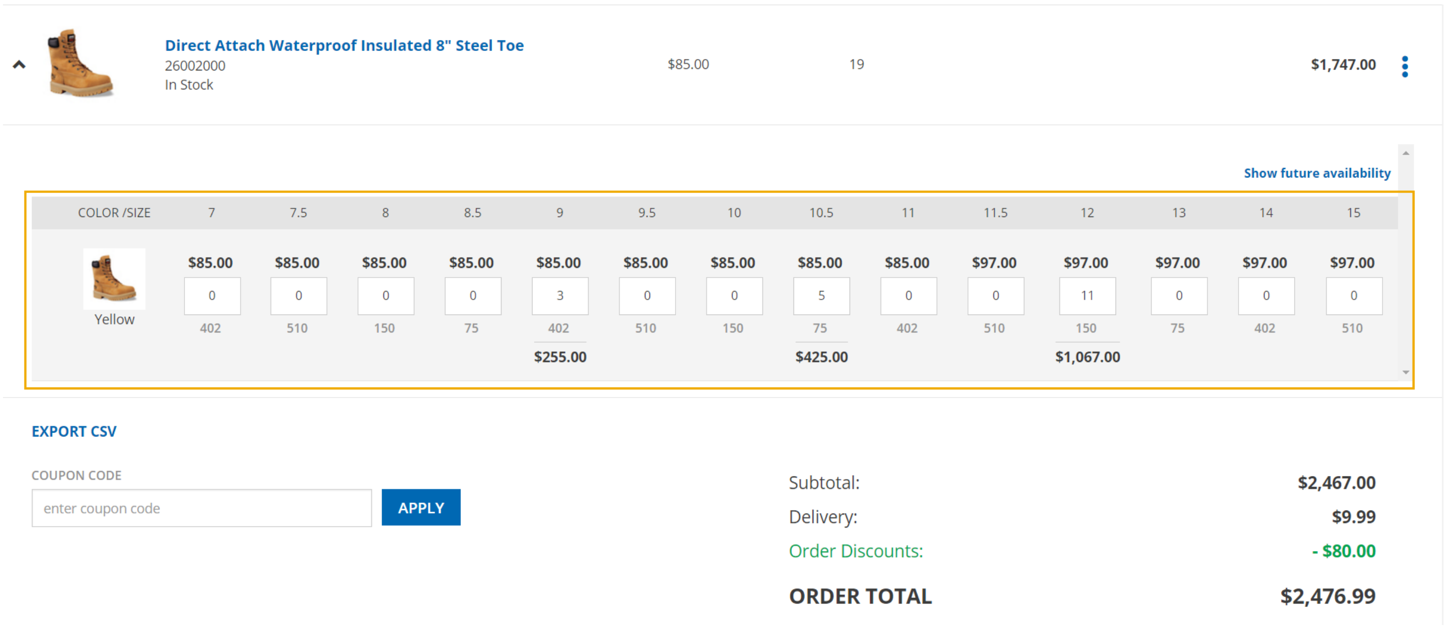
Subtotal:	**$2,467.00**
Delivery:	**$9.99**
Order Discounts:	**- $80.00**
ORDER TOTAL	**$2,476.99**

Changing Quantities

The Cart page can be used to update quantities or remove items from the cart.

To update standard products, customers can change the value directly in the Quantity field. This value is updated automatically when the field loses focus.

For multi-dimensional products, customers can view the quantities of each variant by clicking the chevron to the left of the product image. They can then modify the quantities for each variant, as shown in this screenshot:



	ITEM	PRICE	QTY	TOTAL
	Direct Attach Waterproof Insulated 8" Steel Toe 26002000 In Stock	$85.00	19	**$1,747.00**

Show future availability

COLOR /SIZE	7	7.5	8	8.5	9	9.5	10	10.5	11	11.5	12	13	14	15
Yellow	**$85.00**	**$85.00**	**$85.00**	**$85.00**	**$85.00**	**$85.00**	**$85.00**	**$85.00**	**$85.00**	**$97.00**	**$97.00**	**$97.00**	**$97.00**	**$97.00**
	0	0	0	0	3	0	0	5	0	0	11	0	0	0
	402	510	150	75	402	510	150	75	402	510	150	75	402	510
					$255.00			**$425.00**			**$1,067.00**			

EXPORT CSV

COUPON CODE

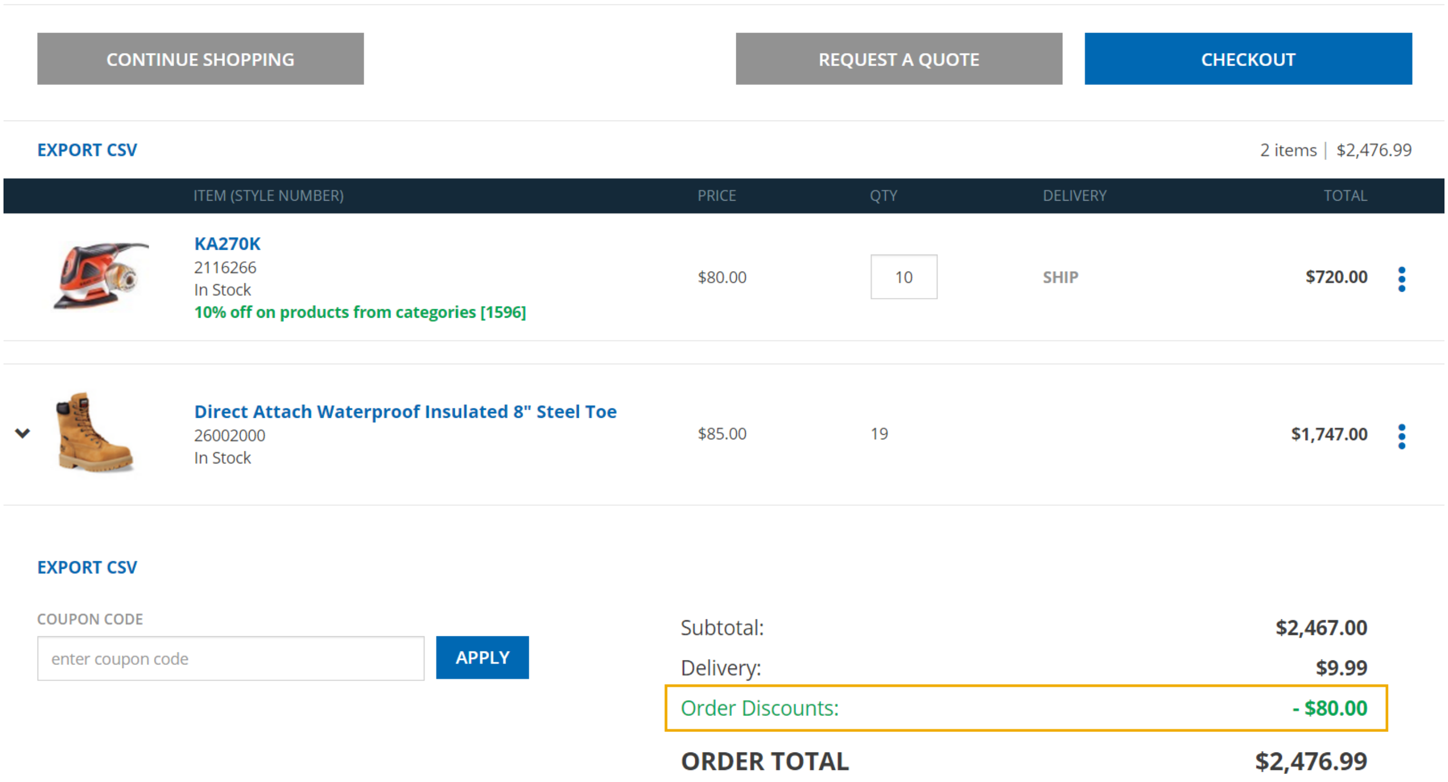
enter coupon code — APPLY

Subtotal:	**$2,467.00**
Delivery:	**$9.99**
Order Discounts:	**- $80.00**
ORDER TOTAL	**$2,476.99**

The grid displays the quantities ordered for each variant. Customers can update the quantities of each variant in the grid, and all cart data is dynamically updated and refreshed when the number in the Quantity field is changed. For example, if the quantity of the size 9 boot is changed from 3 to 0, the quantity, subtotal, average price per unit, and order totals are all automatically updated in the cart.

Received Promotions

If promotions are being applied to the purchase, they appear as Order Discounts in the total calculation section of the cart.



CONTINUE SHOPPING | REQUEST A QUOTE | CHECKOUT

EXPORT CSV — 2 items | $2,476.99

	ITEM (STYLE NUMBER)	PRICE	QTY	DELIVERY	TOTAL
	KA270K 2116266 In Stock 10% off on products from categories [1596]	$80.00	10	SHIP	$720.00
	Direct Attach Waterproof Insulated 8" Steel Toe 26002000 In Stock	$85.00	19		$1,747.00

EXPORT CSV

COUPON CODE
enter coupon code — APPLY

Subtotal:	$2,467.00
Delivery:	$9.99
Order Discounts:	- $80.00
ORDER TOTAL	$2,476.99

Providing Payment and Shipping Information

This section describes the process of supplying payment and shipping information during checkout.

Process Overview

To make a one-time purchase or schedule a replenishment order, customers must first supply payment and shipping information.

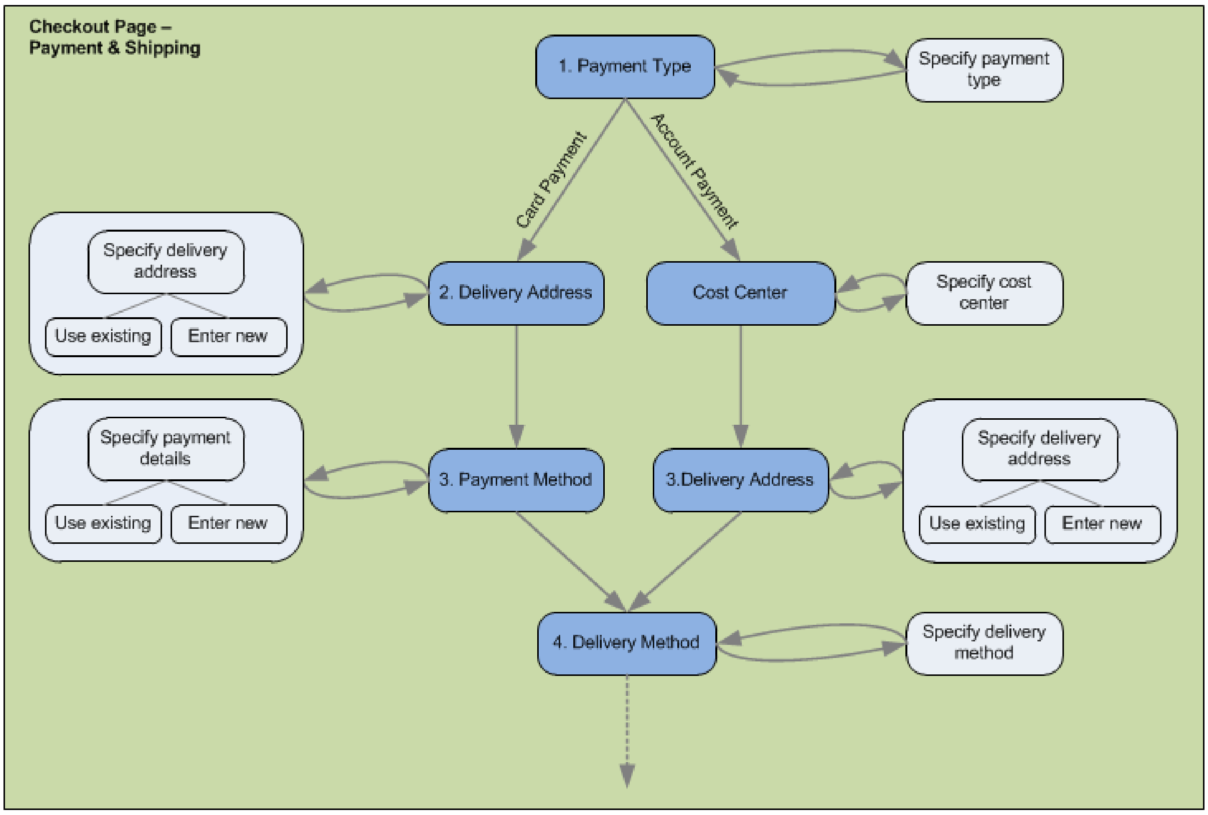
If the payment type is **Account Payment**, the following information is specified in the order presented:

1. Payment type by selecting the Account Payment radio button, cost center, and P.O. number in the Payment Type step.
2. Shipping address in the Shipping Address step.
3. Shipping method (standard, express, etc.) in the Shipping Method step.

If the payment type is Card Payment, the following information is specified in the order presented:

1. Payment type by selecting the Card Payment radio button, and P.O. number in the Payment Type step.
2. Shipping address in the Shipping Address step.
3. Shipping method (standard, express, etc.) in the Shipping Method step.
4. Credit card and billing address information on the Payment and Billing Address step.

The following diagram illustrates this process for both payment types.



Placing a One-Time Order

A one-time order is an order that customers submit manually, as opposed to a scheduled replenishment, which is submitted automatically.

Process Overview

To place a one-time order, customers do the following:

1. Click **Checkout** from the mini cart.
2. Provide payment and shipping information (as described in the Providing Payment and Shipping Information section).
3. Select the Terms and Conditions check box on the **Final Review** step.
4. Click **Place Order**.

After placing the order, an order confirmation page is displayed containing the details of the order. The system also sends the customer an order confirmation email containing the order information.

Depending on the total order value, the order may require approval by the customer's own approvers, or by the merchant.

Viewing the Order History

Customers can view the order history through the My Account page.

To view order history:

1. Click the **My Account** link in the header of any storefront page.
2. Click **Order History** on the My Account page.
3. Click **View** under the **Actions** heading for the order you want to view.

Reordering

Customers can reorder from an existing order.

To reorder:

1. Display the completed order from the Order History page.
2. Click **Reorder**. If the order was made using Account Payment as the payment type, the Final Review page appears with all of the order details filled in. If the order was made using Card Payment as the payment type, the Payment & Billing Address page appears. Once customers have re-entered the credit card payment information, they are directed to the Final Review page, with all of the order details filled in.

Order Approval

Depending on the configuration of the B2B store, orders may require approval from the customer's organization, from the merchant, or both.

Customer Organization Approval

B2B customers are assigned one of the following ordering permissions:

- The **Threshold per order** permission limits how much can be spent per order.
- The **Threshold per timespan** permission limits how much can be spent per day, week, month, quarter, or year.
- The **Budget exceeded permission** allows customers to exceed an assigned budget.

When an order is placed:

- If the order is within the defined threshold for that customer, the order is automatically approved.
- If the order exceeds the defined threshold, the order is set to pending. One of the order approvers for that user must approve or reject the order.

The necessary permissions, thresholds, and approver assignments on the customer side are managed by the client's administrators in the My Company page of the storefront. To view, approve, or reject a pending order, the approver uses the Order Approval Dashboard in the storefront's My Account page.

The Order Approval Dashboard displays orders awaiting approval. This page is only visible to users who are assigned the role of B2B Approver.

To display the order with all of the order details already filled out, the B2B approver clicks **My Account**, then clicks **Order Approval Dashboard**.

The approver can add comments and click either **Approve** or **Reject**.

Merchant Approval

Merchant credit limits are intended to limit the merchant's credit exposure to the customer organization. Credit Limits can be assigned to any unit, for any timespan. Merchants can also create alerts when credit reaches a certain amount, but which do not require merchant approval.

Credit limits are managed by the merchant using the Backoffice, and not by the customer organization's B2B Administrator.

If an order exceeds the B2B unit's credit limit on the merchant's side, the assigned account manager is notified. The account manager must then approve or reject the order using the Backoffice.

Approval Status and History

Approvers can view customer and merchant approval history. Approval/rejection history appears at the bottom of an order and includes other entries like fraud check results.

Scheduling a Replenishment Order

A replenishment order is an order that is automatically placed daily, weekly, or monthly. Replenishment schedules can be set to the following:

- Every x number of days.
- Every x number of weeks. You can also specify the day of the week the order is processed.
- Every month. You can also specify which day of the month the order is processed.

This section describes the process of creating a B2B replenishment order.

Process Overview

To schedule a replenishment order:

1. Click **Checkout** from the mini cart or the regular Cart page.
2. Provide payment and shipping information (as described in the Providing Payment and Shipping Information section).
3. Click **Schedule Replenishment** in the Final Review step.
4. Specify the following replenishment details:
   - The date the auto-replenishment should start
   - When and how frequently the auto-replenishment should occur
5. Click **Schedule Replenishment**.

After the replenishment order is submitted, the Replenishment Order Confirmation page is displayed. The system also sends the customer an email containing the replenishment order information.

Viewing Replenishment Orders

Customers can view replenishment orders through the My Account page.

To view replenishment orders:

1. Click the **My Account** link in the header of any storefront page.
2. Click **My Replenishment Orders** on the My Account page.
3. Under the **Replenishment No** heading, click the number of the replenishment order you want to view.

Ending Automatic Replenishment of Orders

Customers can end the automatic creation of replenishment orders through the My Account page.

To cancel a replenishment order:

1. Click **My Replenishment Orders** on the My Account page.
2. Click **X** in the **Cancel** column for the replenishment order you want to cancel. The Confirm Removal of Replenishment Schedule page appears.
3. Click **Yes**.

Customers can also click **Cancel Replenishment** while viewing a replenishment order.

Triggering of Replenishment Orders

When a replenishment order is triggered, two emails are sent to the customer.

- The first email states that the replenishment order was placed on the customer's behalf automatically.
- The second email is the standard order confirmation email stating that the order was processed.

Summary of Emails Sent to Customers

The following is a list of emails that are sent to customers, depending on the type of order that has been submitted, or the type of action that has been performed:

- One-time orders
  - Confirmation of placement of order
- Replenishment orders
  - Confirmation of creation of a replenishment order
  - Confirmation of placement of an order triggered by an existing replenishment order
- Internal approval (purchaser's own company)
  - Approval pending
  - Approval rejected

No email stating "approved" (regular one-time order approval sent out).

- Overage of credit limit
  - Credit limit overage rejected by merchant

The following events also trigger an email stating confirmation of the placement of an order.

- An order has been placed due to an automatic replenishment order.
- The merchant has approved an order that had surpassed purchase limits (in this case, no special "merchant approval" email is sent).

Other emails can also be generated depending on the approval flow. For example, an order scheduled to be placed today, due to an automatic replenishment, may require internal approval, which would trigger an "order pending approval" email.

Related Information

Customizing B2B Order Approvals
Commerce Quotes

Customizing B2B Order Approvals

The `b2bapprovalextension` extension implements order approvals for B2B scenarios.

The order approvals are highly customizable and are meant as a starting point. Some extension points to be considered within your implementation are described on this page.

B2B Customer Order Approval Management

Commerce B2B Accelerator allows customers to manage the kind of orders their employees can make. By using the `b2bapprovalprocess` extension, all submitted orders pass through an approval process where three order permissions are evaluated.

An example permission is the Order Threshold Permission that grants users the ability to place orders up to a certain threshold amount, for example 1000 EUR.

If the user satisfies all permissions, the order is placed, otherwise it is placed on hold to await further approval. One approval process provided finds approvers assigned to the individual, their unit or its parent units until an approver is found that possesses a missing permission to place the order. The approver is notified to grant or reject approval for the missing permission. Once all permissions are satisfied, the order is placed.

For more information, see:

- b2bapprovalprocess Extension
- B2B Permissions

Custom Permissions

The `b2bapprovalprocess` extension is highly customizable. Any custom permission may be created and added to the approval process to suit particular business needs. To learn how, refer to B2B Permissions.

Adding your Own Approval Process

The approval processes defined in the `b2bapprovalprocess` extension are described in b2bapprovalprocess Extension.

You modify or add you own approval processes as described in b2bapprovalprocess Extension.

Backoffice Integration

Integrating the Backoffice features results in a seamless customer experience by bridging the gap between your online e-commerce system with your accounts and database systems.

Site administrators and merchants use the Backoffice instead of the B2B storefront to control the merchant approvals and most merchant settings.

A customer's B2B site administrator can:

- Define which employees can make purchases
- Define cost centers against which purchases are to be made
- Limit how much an employee can spend, per order or per timespan (day, week, month, quarter, or year)
- Limit how much can be spent against a cost center, by assigning budgets to cost centers and the timespan during which budgets are active
- Define who can approve orders that surpass order limits
- Define standard shipping addresses

Merchants can:

- Define a minimum order limit for quotation requests
- Define separate purchasing limits
- Define credit limits and alerts
- Require early login; when this feature is enabled, customers must log on before being able to view the site's products

Managing Customers

All B2B customers are assigned to a B2B Unit. They must be members of at least one B2B group. Visibility and actions of B2B customers are limited to the B2B organization and B2B branch to which they belong.

Managing Customer User Groups in Backoffice

B2B customers can be organized into user groups using the Backoffice Administration Cockpit.

Managing Order Permissions in Backoffice

A company's purchasers (B2B Customers) are assigned permissions that allow them to make purchases up to a certain amount per order, or a total amount per time span.

B2B Commerce Sequence

To set up a B2B Commerce, you need to define the Merchant and Customer roles, responsibilities, and structure.

Working with B2B Units

You can represent complex organizations through hierarchical structures of B2B units.

Working with Budgets

Budgets help companies to monitor their spending. Budgets are assigned to cost centers, which are used for accounting purposes. Orders or line items of an order are charged against a cost center. One budget may be assigned to multiple cost centers.

Working with Cost Centers

Companies may choose to classify business units as cost centers. Cost centers are usually established in large organizations for multiple purposes such as identifying responsibility, controlling costs, and for accounting purposes.

Managing Credit Limits

In some cases, you may want to assign a credit limit to a customer. The account manager role sets up customer credit limits. Learn how to define B2B account managers in the Backoffice Administration Cockpit and how they manage the credit limits in their B2B units.

Managing Customers

All B2B customers are assigned to a B2B Unit. They must be members of at least one B2B group. Visibility and actions of B2B customers are limited to the B2B organization and B2B branch to which they belong.

B2B customers can be active or inactive. If a B2B customer is inactive, the customer cannot log in and therefore cannot perform any administration activities.

You can move B2B customers from one B2B unit to another within the B2B organization by updating the parent B2B unit. The budgets and cost centers of the updated parent B2B unit area automatically applied to the B2B customers. However, B2B permission groups can also be transferred, but they may need to be modified. An order history is maintained for both the B2B customer and the B2B unit at the time the order was placed, so historical data is maintained.

You can manage B2B customers using the Backoffice Administration Cockpit.

Default Customer Groups

There are four default user groups defined within the system.

User Group	Description
b2bcustomergroup	Members of this group can: • View cost centers • Create and view their own orders

User Group	Description
b2bapprovergroup	Members of this group can: • View cost centers • Create and view their own orders • Approve and reject orders according to the order approval workflow
b2bmanagergroup	Members of this group can: • View cost centers • Create and view their own orders • Approve and reject orders according to the order approval workflow • Manage all items with their branch of the organization tree
b2badmingroup	Members of this group can: • View cost centers • Create and view their own orders • Approve and reject orders according to the order approval workflow • Manage all items with their branch of the organization tree

Creating a B2B Customer

1. Log into the Backoffice Administration Cockpit (`localhost/backoffice`).
2. Go to B2B Commerce B2B Customer . The list of customers appears.
3. Click + B2B Customer.
4. Type the customer's ID, name, and email address in the appropriate fields. Note that a customer ID must be unique within the system.
5. Click Next.
6. Optional: Select the standard language and currency for the customer.
7. Click Next.
8. Select the default B2B unit for the customer.
9. Optional: Select the groups the customer will belong to. The customer is a member of the **b2bcustomergroup** by default.
10. Click Done. The customer is created.

Displaying a Customer

1. Log into the Backoffice Administration Cockpit (`localhost/backoffice`).
2. Go to B2B Commerce B2B Customer . The list of customers appears.
3. Type the name or ID of the customer in the Search field, and then click Search.
4. Click the customer you want to display.

Updating a Customer

1. Display the customer in the Backoffice Administration Cockpit.
2. Make the required changes to the customer.
3. Click **Save**.

Changing a Customer's Parent Unit

1. Display the customer in the Backoffice Administration Cockpit.
2. Click the **General** tab.
3. In the Groups list, point at the existing parent unit. A small **X** appears to the right of the entry. Click the **X** to remove the parent unit.
4. In search field of **Group** list, type part of the name of the new parent B2B Unit.
5. Select the unit name.
6. Click **Save**.

Disabling a Customer

A disabled B2B customer cannot log in.

1. Display the customer in the Backoffice Administration Cockpit.
2. Click the **Administration** tab.
3. In the **Unbound** section, under **Active**, select **False**.
4. Click **Save**.

Managing Customer User Groups in Backoffice

B2B customers can be organized into user groups using the Backoffice Administration Cockpit.

Creating B2B User Groups

To create a B2B user group:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Commerce B2B User Group** . The list of user groups appears.
3. Click **+ B2B User Group**.
4. In the **ID** field, type the user group's ID. Note that a customer ID must be unique within the system.
5. From **Parent B2B Unit**, select the user group's parent unit.
6. Enter other information as required.
7. Click **Next** and select languages and catalog versions as required.
8. Click **Done**.

Updating a B2B User Group

To update a B2B user group:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Commerce B2B User Group** . The list of user groups appears.

3. Select the user group you want to modify.
4. Make the required changes.
5. Click **Save**.

Deleting a B2B User Group

To delete a B2B user group:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Commerce** **B2B User Group** . The list of user groups appears.
3. Select the user group you want to delete.
4. Click **Delete** in the toolbar, and then click **Yes** to confirm.

Displaying Members of a B2B User Group

To display the members of a B2B user group:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Commerce** **B2B User Group** . The list of user groups appears.
3. Click a user group. The details for that user group appears.
4. Click the **Administration** tab. The users are shown in the **Members** list box.

You can also use this list box to search for members.

Managing Order Permissions in Backoffice

A company's purchasers (B2B Customers) are assigned permissions that allow them to make purchases up to a certain amount per order, or a total amount per time span.

If an order is within the scope of permissions when it is placed, the order is automatically approved. If the order exceeds the permission, the order is placed in a pending approval state. The company's order approvers (B2B Approvers) are notified and must approve or reject the order.

For example, Mark, a purchaser, is allowed to purchase up to $500 per order, but not more than $1200 per month.

- Mark's first order of the month, for $400, is automatically approved.
- Mark makes a second order of $600 a few days later; it is submitted for approval. Assuming that this order is approved, then Mark has no now spent $1000
  - Assuming Mark's previous order was approved, he has now spent $1000.
- Mark makes another purchase for $300. This order is also submitted for approval, since he is now surpassed the $1200 limit.

Approvers also have limited permissions. If a purchase surpasses an approver's limit, the approval is escalated to the next person in the approval chain, ending with a B2B Administrator.

Displaying a B2B Permission

To display B2B permissions:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Approval Process** **B2B Permission** . The list of permissions appears.

3. Type the name or ID of the permission in the **Search** field, and then click **Search**.
4. Click a permission to display it.

Creating a B2B Permission

To create a B2B permission:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Approval Process** **B2B Permission**. The list of permissions appears.
3. Click the down arrow next to **+ B2B Permission**, and then click the arrow to the left of **+ B2B Permission**, and then click the type of permission you want to create.

   i Note

   Clicking **+ B2B Permission** directly gives an error.

4. Fill in the required fields, and then click **Done**.

Updating a B2B Permission

To update a B2B permission:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Approval Process** **B2B Permission**. The list of permissions appears.
3. Type the name or ID of the permission in the **Search** field, and then click **Search**.
4. Click the permission you want to modify.
5. Make your changes, and then click **Save**.

Deleting a B2B Permission

i Note

Deleting a permission does not affect orders that were already placed, approved or assigned to be approved. It applies only to future actions.

To delete a B2B permission:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **B2B Approval Process** **B2B Permission**. The list of permissions appears.
3. Type the name or ID of the permission in the **Search** field, and then click **Search**.
4. Click the permission you want to delete.
5. Click **Delete** in the toolbar, and then click **Yes** to confirm.

Managing B2B Permission Groups

B2B Permission Group is a user group that you assigned permissions to. For more information on how to manage B2B User groups, see Managing Customer User Groups in Backoffice.

Related Information

B2B Orders

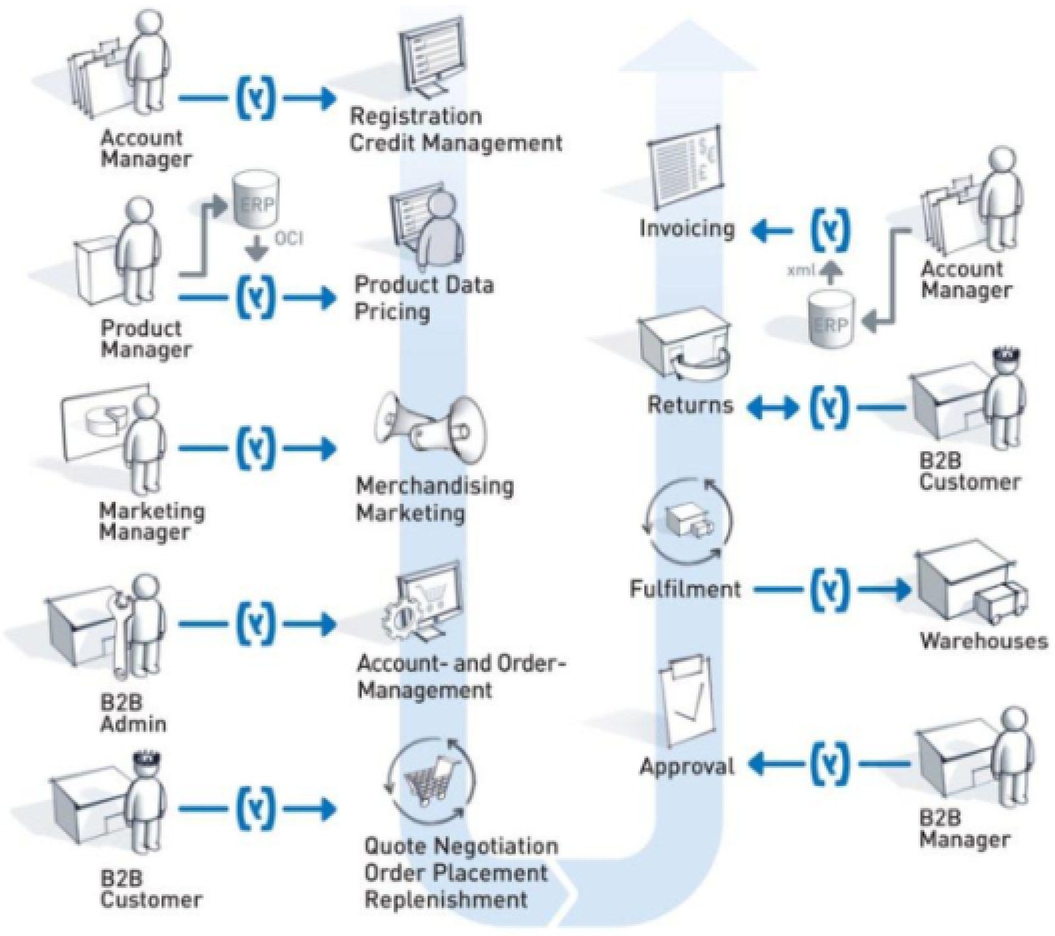
b2bapprovalprocess Extension

B2B Permissions

B2B Commerce Sequence

To set up a B2B Commerce, you need to define the Merchant and Customer roles, responsibilities, and structure.

When a company is set up to make purchases through a B2B storefront, the following diagram illustrates the SAP Commerce B2B Commerce sequence functionality.



Merchant Sequence

- The manager for the new account registers a representative of the new company as the B2B Administrator, who will be responsible for setting up the new company's organization within the B2B site.
- The account manager also sets up credit limits and alerts. If credit limits are surpassed, the purchase requires the merchant's approval, which is separate from the customer's own approval rules based on employee and cost center purchasing limits.
- The product manager sets up product catalogs and pricing specific to the B2B customer. This process can be managed through SAP Commerce cockpits or through an ERP system.
- The marketing manager configures promotions and other marketing features.

Customer Sequence

- The customer's B2B Administrator defines an organization by adding units (which can represent companies, divisions, locations, or any other entity), users, buying permissions, approval permissions, cost centers, and budgets.

- Employees from the customer company that are registered on the B2B site can make one-time purchases and schedule automatic replenishments. Orders are subject to approval by the purchaser's company or by the merchant depending on purchasing and credit limits.

Working with B2B Units

You can represent complex organizations through hierarchical structures of B2B units.

B2B units are the foundation nodes of the B2B organization tree. They represent a division, department, office, factory, or any other group within the organization. You can apply B2B employees, budgets, cost centers, and so on, to created B2B units.

Displaying a B2B Unit

1. Log into the Backoffice Administration Cockpit (`localhost/backoffice`).
2. Click B2B Commerce B2B Unit . The list of units appears.
3. Type the name or ID of the unit in the Search field, and then click Search.
4. Click a unit.

Creating a B2B Unit

1. Log into the Backoffice Administration Cockpit (`localhost/backoffice`).
2. Click B2B Commerce B2B Unit . The list of units appears.
3. Click + B2B Unit.
4. Fill the fields with appropriate data. The only required field is ID.
5. Click Done.
6. Display the unit you just created.
7. In the Organization tab, select the parent unit by adding a unit to the Group field. If you create a root company, you don't need to select any parent unit.
8. In the Organization tab, select child units by adding units to the Members list.
9. Click Save.

Editing a Unit

1. Log into the Backoffice Administration Cockpit (`localhost/backoffice`).
2. Click B2B Commerce B2B Unit . The list of units appears.
3. Click a unit. The system displays the unit editor.
4. Browse the tabs to add or edit the information you need. For example:
   - Add an order approver group:
     a. Click the Approvers tab.
     b. Select or create approver groups from the Approver Groups list.
     c. Click Save.
   - Add an order approver:
     a. Click the Approvers tab.
     b. Select or create approvers from the Approver list.
     c. Click Save.
   - Add a cost center or budget:

a. Click the Cost Center tab.

b. Select or create budgets from the B2B Budgets list.

c. Select or create cost centers from the B2B Cost Centers list.

d. Click Save.

Disabling a Unit

i Note

When you disable a unit, all its descendant B2B units are also disabled. All nodes linked to this unit and all their descendant units will also be disabled. These nodes might be such things as employees, cost centers and budgets. To revert, you need to manually enable each unit.

1. Display the unit you want to disable.
2. Click Disable B2B Unit from the toolbar. You are asked to confirm.
3. Click Yes.

Enabling a Unit

1. Display the unit you want to enable.
2. Click Enable B2B Unit from the toolbar. The unit is enabled.

Related Information

B2B Units and Organization Tree

Working with Budgets

Budgets help companies to monitor their spending. Budgets are assigned to cost centers, which are used for accounting purposes. Orders or line items of an order are charged against a cost center. One budget may be assigned to multiple cost centers.

Cost centers and budgets are managed using the Backoffice Administration Cockpit.

Displaying a Budget Using

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Click B2B Commerce > B2B Budget. The list of budgets appears.
3. Type the name or ID of the budget in the Search field, and then click Search.
4. Click the budget you want to display.

Creating a Budget Using the Backoffice Administration Cockpit

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Click B2B Commerce > B2B Budget. The list of budgets appears.
3. Click + B2B Budget.
4. Type the budget's ID in the appropriate field. Note that the ID must be unique within the system.
5. Specify the budget's currency, date range, parent B2B unit, budget amount, in the appropriate fields.
6. Click Done.

Updating a Budget Using the Backoffice Administration Cockpit

1. Display the budget you want to modify.
2. Make the required changes to the budget.
3. Click **Save**.

Disabling a Budget

To disable a budget, go to the **General** tab and set **Active** to **False**. Disabling a budget means that you are not able to check out any new orders against this budget.

Related Information

[B2B Budgets and Cost Centers]()

Working with Cost Centers

Companies may choose to classify business units as cost centers. Cost centers are usually established in large organizations for multiple purposes such as identifying responsibility, controlling costs, and for accounting purposes.

Cost centers and budgets are managed using the Backoffice Administration Cockpit.

Displaying a Cost Center Using

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Click **B2B Commerce** **B2B Cost Center** . The list of cost enters appears.
3. Type the name or ID of the cost center in the **Search** field, and then click **Search**.
4. Click the cost center you want to display.

Creating a Cost Center Using

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Click **B2B Commerce** **B2B Cost Center** . The list of cost centers appears.
3. Click **+ B2B Cost Center**.
4. Type the cost center's ID in the appropriate field. Note that the ID must be unique within the system.
5. Specify the cost center's currency and parent B2B unit in the appropriate fields.
6. Click **Done**.

Updating a Cost Center

1. Display the cost center you want to modify.
2. Make the required changes to the cost center.
3. Click **Save**.

Disabling a Cost Center

To disable a cost center, go to the**General** tab and set **Active** to **False**. Disabling a cost center means that you are not able to check out any new orders against the cost center.

Related Information

B2B Budgets and Cost Centers

Managing Credit Limits

In some cases, you may want to assign a credit limit to a customer. The account manager role sets up customer credit limits. Learn how to define B2B account managers in the Backoffice Administration Cockpit and how they manage the credit limits in their B2B units.

A credit limit defines the maximum amount of credit that an organization or unit can have. You can define multiple credit limits for each unit. A credit limit alert defines when an account manager should be informed that the customer is approaching the credit limit.

A credit limit alert is triggered when the amount of used credit exceeds the specified value.

To start using a credit limit for any unit, you have to assign an account manager. This is the role that is responsible for group of customers and manages a credit limit parameters.

i Note

When a B2B credit limit is assigned to more than one B2B unit, the credit limit is not shared between the B2B units. For example, if you assign a $100,000 credit limit to unit A and unit B this means that the total value of the credit limits assigned is $200,000. Unit A has $100,000 and unit B has $100,000.

Defining Account Managers for Units

To define unit account managers:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to B2B Commerce B2B Unit . The list of units appears.
3. Type the name or ID of the unit in the Search field, and then click Search.
4. Click the unit you want to modify.
5. Click the Merchant Check tab, which displays the account managers or account manager groups already associated with the unit.
6. Select an account manager from the Account Manager field. To change the existing entry, first remove the entry by pointing at the entry and clicking the X that appears to the right.
7. Select account manager groups from the Account Manager Groups list.
8. Click Save.

Displaying Credit Limits

To display credit limits:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to B2B Commerce B2B Unit . The list of units appears.
3. Type the ID of the credit limit in the Search field, and then click Search.
4. Click the credit limit you want to display.

Creating a Credit Limit

To create a credit limit:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to B2B Commerce B2B Unit . The list of units appears.
3. Click the down arrow next to + B2B Merchant Check, click the arrow to the left of B2B Merchant Check, and then click B2B Credit Limit.
4. Specify the following required information.

Field	Description
ID	The ID of a credit limit. Note that a credit limit ID must be unique within the system.
Currency	The currency of a credit.
Amount	The value of a credit to be assigned.

5. Click Done.

Creating a Credit Limit Alert

To create a credit limit:

1. Display a credit limit in the Backoffice Administration Cockpit.
2. In the General tab, in the Properties section, choose a type of alert from Alert Rate Type drop-down list. You can choose from either Currency or Percentage options. The Currency type means that the alert is triggered after a certain value is achieved for Percentage type, the alarm is triggered after certain percent value is achieved.
3. Type a value into the Alert Threshold field.
4. Click Save.

Assigning a Credit Limit to a Unit

To assign a credit limit to a B2B unit:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to B2B Commerce B2B Unit . The list of units appears.
3. Type the name or ID of the unit in the Search field, and then click Search.
4. Click the unit you want to modify.
5. Click the Merchant Check tab, which displays the currently assigned credit limit in the Credit Limit field.
6. Select a credit limit from the Credit Limit field. To change the existing credit limit, first remove the entry by pointing at the entry and clicking the X that appears to the right.
7. Click Save.

Editing a Credit Limit or Credit Limit Alert

To edit a credit limit or alert:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to B2B Commerce B2B Merchant Check . The list of credit limits appears.
3. Type the ID of the credit limit in the Search field, and then click Search.
4. Click the credit limit you want to modify.

5. Modify the credit limit information.
6. Click **Save**.

Catalogs and Pricing

You can link the pricing and product catalogs to specific units based on the logged-in purchaser.

Setting Product Visibility

You can restrict the visibility of certain product groups to a user group. This is useful if you want to create multiple product catalogs for various B2B Customers.

Setting Customized Prices

B2B customers may have individually negotiated price lists that can be managed either in SAP Commerce or in the ERP system. Customer-specific prices can be negotiated on different levels, either on the B2B organization, B2B company, or B2B department.

B2B Multi-Dimensional Products

Learn about multi-dimensional products and their creation using the Backoffice Administration Cockpit.

Setting Product Visibility

You can restrict the visibility of certain product groups to a user group. This is useful if you want to create multiple product catalogs for various B2B Customers.

Find more information on product catalog in Product Content and Catalogs.

Restricting Visibility of a Product Group

To restrict visibility of a product group:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Click **Catalog Categories**. The list of categories appears.
3. Type the name or ID of the category in the search field, and then click **Search**.
4. Click the category you want to modify.
5. In the **General Category Visibility** section, in the **Visible to** list, add or remove customers, groups and units.
6. Click **Save**.

Related Information

B2B Price, Product and Catalog Management
Product Content and Catalogs

Setting Customized Prices

B2B customers may have individually negotiated price lists that can be managed either in SAP Commerce or in the ERP system. Customer-specific prices can be negotiated on different levels, either on the B2B organization, B2B company, or B2B department.

In many cases, individual price lists are negotiated per article or article group. Many of such price lists may exist. A base price list is available for all customers who do not have an individual price list.

You can assign the customized prices to a single B2B customer, to a B2B customer group, or to a whole B2B unit.

Setting up Customized Prices

To set up customized prices:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Do one of the following:
    - To set up customized prices for a B2B customer, go to **B2B Commerce B2B Customer** , and then select a customer.
    - To set up customized prices for a B2B group, go to **User User Groups** , and then select a user group.
    - To set up customized prices for a B2B unit, go to **B2B Commerce B2B Unit** , and then select a B2B unit.
3. In the **Prices** tab, set the **Price Group**, **Tax Group**, and **Discount Group**.
4. Click **Save**.

Related Information

B2B Price, Product and Catalog Management
Price, Tax, and Discount Calculation

B2B Multi-Dimensional Products

Learn about multi-dimensional products and their creation using the Backoffice Administration Cockpit.

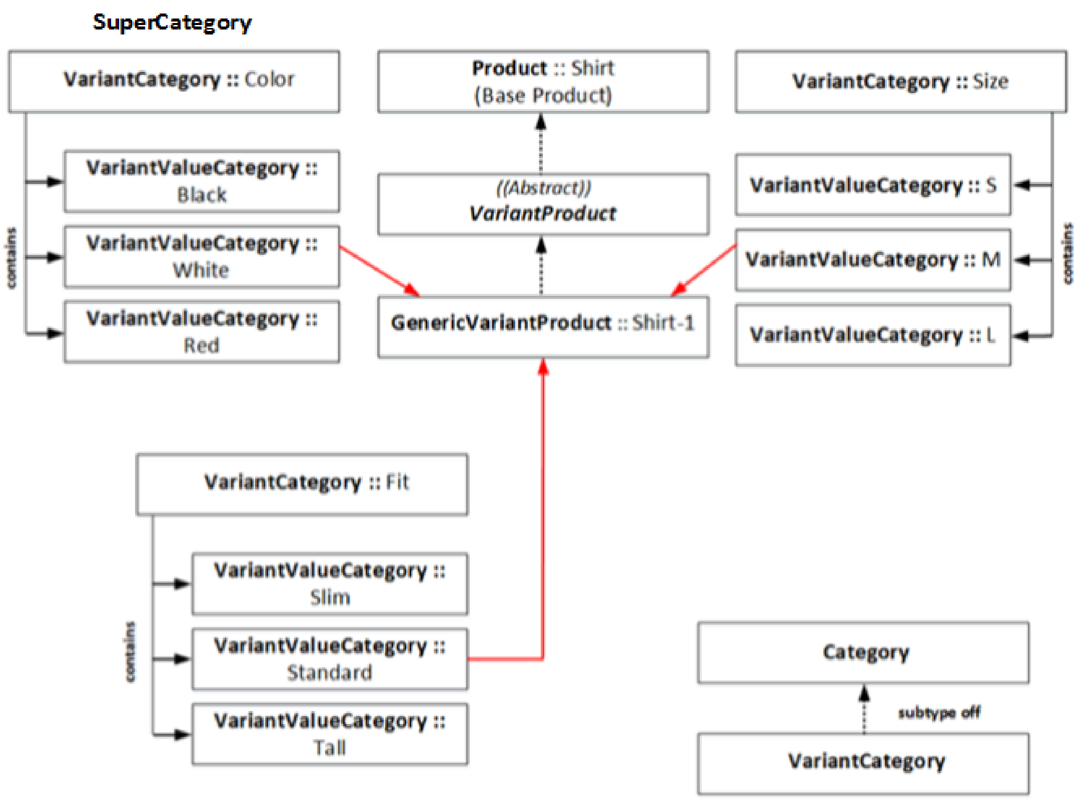
Introduction

Multi-dimensional products are described using multiple attributes. You can define the dimensions and attributes needed to describe the product. Out of the box, the B2B Commerce Accelerator includes three dimensions. You could use one, two, or three dimensions to describe your product under this model. For example, a shirt has the following dimensions and attributes:

- Color: black
- Size: medium
- Fit: tall

Order forms displaying the product attributes are used to add multiple products to the cart at the same time.

This graphic below shows how the `VariantCategory` dimensions color, size, and fit are used to describe the different characteristics of a product. You can also see how the `VariantValueCategory` attributes describe the color of the product, such as black, white, and red.

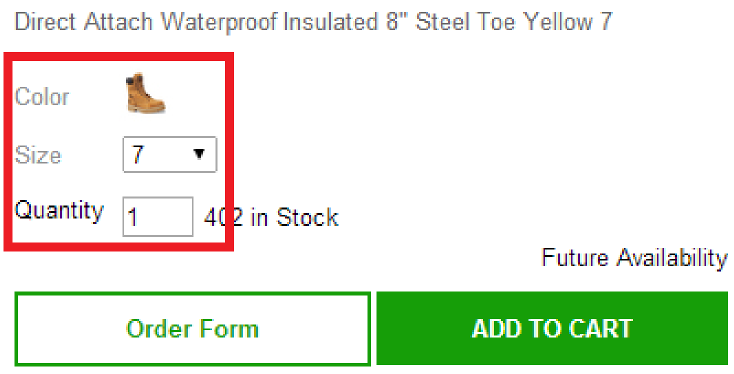

SuperCategory
VariantCategory :: Color
Product :: Shirt (Base Product)
VariantCategory :: Size
VariantValueCategory :: Black
VariantValueCategory :: White
VariantValueCategory :: Red
((Abstract)) VariantProduct
GenericVariantProduct :: Shirt-1
VariantValueCategory :: S
VariantValueCategory :: M
VariantValueCategory :: L
contains
contains
VariantCategory :: Fit
VariantValueCategory :: Slim
VariantValueCategory :: Standard
VariantValueCategory :: Tall
contains
Category
subtype off
VariantCategory

This table contains descriptions for the elements included in the graphic.

Name	Description
`Product`	A Shirt is the base product. Nothing describes the shirt; not size, color, or fit.
`GenericVariantProduct`	A `GenericVariantProduct` describes a product with at least one `VariantValue` attribute.
`SuperCategory`	A `SuperCategory` is a root category, which has subordinate categorys. These subordinate categorys share something in common with each other regarding the supercategory.
`VariantCategory`	A `VariantCategory` is equivalent to one of the dimensions, such as size, color, or fit.
`VariantValueCategory`	A `VariantValueCategory` is an attribute of a dimension, such as red, green, large, small, narrow, or wide.

Adding Multi-Dimensional Products to the Cart

Because a multi-dimensional product actually corresponds to many similar products, users must specify the product's variants when adding to the cart, as shown in the following example from B2B Accelerator.



The B2B Accelerator also provides an example implementation of how a customer could order multiple variants at once. For more information, see B2B Saved Order Forms.

Building a Variant Product through Categorization

To create multi-dimensional products, it is important to first create the variant category, then the required variant value categories, and then finally create the products. Refer to these topics for more details:

1. Creating Variant Categories
2. Creating Variant Value Categories
3. Creating Multi-Dimensional Products

Creating Variant Categories

A variant category describes a dimension of a product, such as color or size. A maximum of three variant categories are available. By default, these are set to color, size, and fit. You can modify the variant categories to represent other values, such as weight, length, finish, experience, and so on.

Creating Variant Value Categories

Product variant categories define the product variants, such as size or color. Once a variant category is defined, you must create the values for each category (for example, large or blue).

Creating Multi-Dimensional Products

Multi-dimensional products represent variations of a base product and are offered in multiple unique combinations of these dimensions. A multi-dimensional stock keeping unit is composed of a defined variant category and variant value category, and is created in the Backoffice Administration Cockpit.

Related Information

Modeling Product Variants

Styles and Size Variants in the SAP Commerce Product Cockpit

Catalog Guide

Creating Variant Categories

A variant category describes a dimension of a product, such as color or size. A maximum of three variant categories are available. By default, these are set to color, size, and fit. You can modify the variant categories to represent other values, such as weight, length, finish, experience, and so on.

Creating a Variant Category

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Go to **Catalog** **Categories** . The list of categories appears.
3. Click the down arrow next to the **+** icon, click the arrow to the left of **Category**, and then click **Variant Category**.

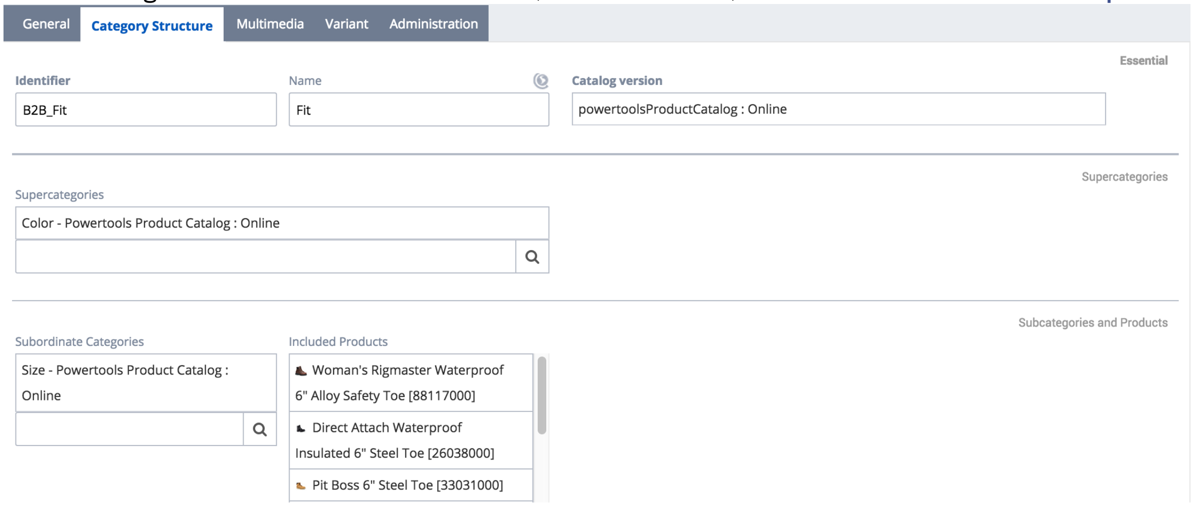
4. Specify a unique ID in the Identifier field.
5. From the Catalog Version drop-down menu, select the Catalog Version.
6. Click Done.

Modifying the Variant Category you created

1. Display the VariantCategory you just created.
2. Click the General tab.
3. Type a name to the variant category. Adding a name helps when searching for the variant category later.
4. In the Category Visibility section, add the users who can view the variant category.
5. In the Description section, type a description for the VariantCategory.
6. Click Save.

Assigning a Super Category

1. Define a second VariantCategory.
2. Display one of the VariantCategory items you created.
3. Click the Category Structure tab.
4. In the Supercategories area, add a supercategory. This chains the two together. In the example below, you can see a chain of three categories. Color is the first in the chain, Fit is the second, and Size is the third. Fit is the SuperCategory for Size.



Identifier: B2B_Fit
Name: Fit
Catalog version: powertoolsProductCatalog : Online

Supercategories: Color - Powertools Product Catalog : Online

Subordinate Categories: Size - Powertools Product Catalog : Online

Included Products:
- Woman's Rigmaster Waterproof 6" Alloy Safety Toe [88117000]
- Direct Attach Waterproof Insulated 6" Steel Toe [26038000]
- Pit Boss 6" Steel Toe [33031000]

5. Click Save.

Related Information

Modeling Product Variants
Styles and Size Variants in the SAP Commerce Product Cockpit

Creating Variant Value Categories

Product variant categories define the product variants, such as size or color. Once a variant category is defined, you must create the values for each category (for example, large or blue).

A `VariantValueCategory` describes an attribute of a `VariantCategory`. If the `VariantCategory` covers the product attribute color, the product can be offered in many different colors. Each `VariantValueCategory` is equivalent to one color of the product. Values for `VariantValueCategory` can be defined in the Backoffice Administration Cockpit.

To create a `VariantValueCategory`:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. Navigate to **Catalog Categories** . The list of categories appears.
3. Click the down arrow next to **+ Category**, click the arrow to the left of **Category**, and then click **Variant Value Category**.
4. From the **Catalog Version** drop-down menu, select the Catalog Version.
5. Specify a unique ID in the **Identifier** field.
6. Click **Done**.
7. Display the **VariantValueCategory** you just created.
8. Click the **General** tab. In the **Category Visibility** section, add the users who can view the variant value category. In the **Description** section, type a description for the **VariantValueCategory**.
9. Click the **Category Structure** tab. In the **Supercategories** section, add a supercategory for the VariantValueCategory you are creating (**color** for example).
10. Click the **Variant** tab. In the **Attributes** area, type a value in the **Sequence** field. Valid values are greater than or equal to zero. Each **VariantValueCategory** has a unique **Sequence** attribute, and this attribute value stays the same through the life of the product. The **Sequence** attribute describes the order in which the **VariantCategory** colors are defined.
11. Click **Save**.

Related Information

Modeling Product Variants
Styles and Size Variants in the SAP Commerce Product Cockpit

Creating Multi-Dimensional Products

Multi-dimensional products represent variations of a base product and are offered in multiple unique combinations of these dimensions. A multi-dimensional stock keeping unit is composed of a defined variant category and variant value category, and is created in the Backoffice Administration Cockpit.

Multi-dimensional products are defined by one or more `VariantCategory`, and each variant category has one or more `VariantValueCategory`. For example, shoes can have variant categories for size, color, and width, and have variant value categories for the different available colors. The required `VariantCategory` and `VariantValueCategory` must be defined before creating a multi-dimensional product. For more information, see Creating Variant Categories and Creating Variant Value Categories.

There are three main steps for creating multi-dimensional products:

- Create a base product
- Allow variants on the base product
- Add variant products to the base product

Creating a Multi-Dimensional Base Product

The first step in creating a multi-dimensional product is to create a base product. The base product defines in which category its variants are available in the storefront, as well as the variant category used by its variants.

To create a base product:

1. Log into the Backoffice Administration Cockpit (`https://localhost:9002/backoffice`).
2. In the navigation bar, click **Catalog Products** .
3. Click the **+** icon in the tool bar to open the Create New Product wizard.
4. In the **Essentials** tab, enter a unique **Article Number**, **Approval Type** (approved), **Catalog Version**, and then click **Next**

5. In the **Description** tab, enter the product's **Identifier** and **Description**, and then click **Next**.
6. In the **Categorization** tab, select a super category. Note that two different super categories are set in this field:
    - The catalog category where the product is found in the storefront, such as a brand category or product category
    - The variant category for multi-dimensional products, such as color or size

    Color, **Fit**, and **Size** are available out-of-the-box. If adding these super categories, ensure that they match the catalog version of the product.
7. Click **Done**.

Allow Variants on Base Product

The next step is to configure your base product to allow for variant products:

1. In the Backoffice Administration Cockpit navigation bar, navigate to **Catalog** **Products** , search for the base product you just created, and then click that product.
2. In the **Variants** tab, select **Generic Variant Product** in the **Product variants type** field.
3. In the **Price** tab, click **Create new Price Row**, and then add a **Unit**, a **Price**, and a **Currency**. Click **Done** when you are finished.
4. In the **Multimedia** tab, add an **Image**.
5. In the **Extended Attributes** tab, add an **EAN**.
6. Click **Save**.

Adding Variant Products to the Multi-Dimensional Base Product

The last step is to add the variant products to the base product. The variant products are the stock keeping units that customers can buy in the storefront.

To create the variant products:

1. In the Backoffice Administration Cockpit navigation bar, navigate to **Catalog** **Products** .
2. Create a new **Generic Product Variant**:
    - Click the down arrow next to the **+** icon in the tool bar.
    - Click the arrow to the left of **Product**.
    - Click the arrow to the left of **Variant Product**.
    - Click **Generic Product Variant**.
3. In the **Essentials** tab, enter the unique article number, approval type, catalog version, and base product, and then click **Next**. Ensure that:
    - The **Base Product** field to the multi-dimensional base product you just created.
    - The same catalog version is used.
4. In the **Description** tab, enter the product's identifier and description, and then click **Next**.
5. In the **Categorization** tab, select the variant value category. This defines the actual value of the variant for the product, such as brown.
6. Click **Done**.
7. Repeat the procedure for any other variants of the base product.

Related Information

Modeling Product Variants

[Styles and Size Variants in the SAP Commerce Product Cockpit]

Online Ordering

B2B-specific checkout features offere more than standard features like being able to create one-time orders or re-ordering from existing orders.

These features include

- Schedule automatic replenishment orders
- Request a quotation from the merchant, if the current order has reached a configurable minimum
- Specify account payment (with purchase order number) as well as credit card
- Specify the cost center the purchase should be made against
- Choose from standard delivery addresses associated with the unit or cost center, or create new ones
- Orders are automatically approved or sent for approval depending on purchasing rights, cost centers, and budget settings
- Approvers can easily view all orders pending their approval
- Purchasers can view pending and fulfilled orders, pending and responses to quotes, and scheduled replenishments
- Quotation requests and the back-and-forth communication during the negotiation process is handled automatically (the merchant provides his or her response through the B2B Admin Cockpit)
- Emails are sent automatically to confirm orders or responses to interactions

Purchasers can also manage their credit card information and their own additional shipping addresses through the B2B site.

Working with Orders
The B2B Commerce Module lets buyers place a one-time order or create a replenishment order.

Working with Orders

The B2B Commerce Module lets buyers place a one-time order or create a replenishment order.

Order-related functionality delivered with B2B Commerce Module includes:

- Support for complex purchasing workflows.
- Empower end-users to buy within spend limits by using manual order approval, automatic order approval when an order does not exceed the limit, notification for the approver of pending orders for review.
- Enable efficient global fulfillment through partial delivery and multi-warehouse shipping.
- Ensure just-in-time availability with automatic replenishment.

i Note
To make purchases as a B2B customer, the user must belong to the b2bcustomergroup.

Placing a One-Time Order

1. Browse to the B2B Accelerator site.
2. Click **Login**. The Returning Customer page appears.
3. Type your username and password, and then click **Login**.
4. Add some products to the cart.
5. Show the cart, and the click **Checkout**.

6. Provide payment and delivery information.
7. Select the **Terms and Conditions** check box.
8. Click **Place Order**.

You can view the state of your orders by clicking **My Account**.

Creating a Replenishment Order

1. Browse to the B2B Accelerator site.
2. Click **Login**. The Returning Customer page appears.
3. Type your username and password, and then click **Login**.
4. Add some products to the cart.
5. Show the cart, and the click **Checkout**.
6. Provide payment and delivery information.
7. Select the **Terms and Conditions** check box.
8. Click **Schedule Replenishment**.
9. Specify the start date and type of replenishment.
10. Click **Place Replenishment Order**.

Related Information

B2B Orders
Managing Order Permissions in Backoffice

Self-service Account Management

Each customer manages their own organization through the B2B site.

A customer's B2B site administrator can:

- Define which employees can make purchases
- Define cost centers against which purchases are to be made
- Limit how much an employee can spend, per order or per timespan (day, week, month, quarter, or year)
- Limit how much can be spent against a cost center, by assigning budgets to cost centers and the timespan during which budgets are active
- Define who can approve orders that surpass order limits
- Define standard shipping addresses

Sales Organization

The Sales Organization feature allows you to manage sales organizations. You can use it to create and modify sales organizations, and it also allows you to assign sales representatives to customers.

The Sales Organization feature can be used with any responsive Accelerator, but the main use-case is focused on B2B.

To explore a complete sample use case of the Sales Organization feature, you can install the **commerceorgsamplesaddon** AddOn. For more information, see commerceorgsamplesaddon AddOn.

Organization Unit Service

The organization unit service interface and class implementation provide a way to perform operations in Backoffice using domain-specific APIs for all actions.

The organization unit service interface, `OrgUnitService.java`, is located in `de.hybris.platform.commerceservices.organization.services`.

The organization unit service class implementation, `DefaultOrgUnitService.java`, is located in `de.hybris.platform.commerceservices.organization.services.impl`

The `DefaultOrgUnitService.java` bean is defined in `commerceservices-spring.xml` as follows:

```
<alias name="defaultOrgUnitService" alias="orgUnitService"/>
        <bean id="defaultOrgUnitService" class="de.hybris.platform.commerceservices.organization.servi
```

Organization Unit APIs for Backoffice

Instead of the generic persistence API, the custom organization unit Backoffice widgets use domain-specific APIs for all actions, especially for CRUD operations. This ensures the widgets are extensible and that additional logic is applied.

OrgUnitService

The following table describes Backoffice functionality that is available through the API.

Use Case	Description
Get org unit	Returns an `optional` to enforce consumers to verify if the unit is actually present. No exception or null value handling is necessary.
Create new org unit	Creates a new organization unit.
Update org unit	Uses the `orgUnit` attribute in `OrgUnitParameter` for updates.
Enable org unit	Activates the organization unit passed in the `OrgUnitParameter.orgUnit` property.
Disable org unit	Deactivates the organization unit passed in the `OrgUnitParameter.orgUnit` property as well as its child units.
Add employee or customer to sales unit	Multiple employees or customers can be added at once. They must be passed using the `OrgUnitMemberParameter.members` property.
Remove employee or customer from sales unit	Multiple employees or customers can be removed at once. They must be passed using the `OrgUnitMemberParameter.member` property.
Get employees or customers of sales unit	Returns a paged search result for members of the given organization unit.
Get parent	Returns an `optional` to enforce consumers to verify if the unit's parent is actually present. No exception or null value handling is necessary.

Parameter Beans

The following extensible parameter beans are defined in `commerceservices-beans.xml`. You can use these beans instead of using multiple method parameters, unless the API method has only one parameter and is unlikely to change:

- `OrgUnitParameter` is used for creating and updating an organizational unit.
- `OrgUnitMemberParameter<T>` is used for operations related to members of an organization unit.

Authorization Strategy and Configuration Properties

User authorization for organization-related operations in the service layer can be configured in the `commerceservices/project.properties` file, and can be overwritten in your extension's `project.properties` file. The following table describes the access rights for organization related operations represented by configuration properties:

Property	Description
`commerceservices.organization.rights.create.groups`	User groups authorized to create **OrgUnits**
`commerceservices.organization.rights.edit.groups`	User groups authorized to edit **OrgUnits**
`commerceservices.organization.rights.edit.parent.groups`	User groups authorized to change the parent of **OrgUnits**
`commerceservices.organization.rights.view.groups`	User groups authorized to view **OrgUnits**

These access rights can be granted on a usergroup level by assigning a comma separated list of usergroups to the respective property. The properties are used by `DefaultOrgUnitAuthorizationStrategy.java`, which is injected in `DefaultOrgUnitService.java` to validate the user access rights for all of the service methods. The following example shows how the properties are configured in `commerceservices/project.properties`:

```
#########
# Access rights for organization related CRUD operations. The properties below each take a comma separ
# list of usergroups. By default members of orgemployeegroup can perform all CRUD operations. For a fi
# grained access control create your own usergroups and overwrite the properties in your extension.
commerceservices.organization.rights.create.groups=orgemployeegroup
commerceservices.organization.rights.edit.groups=orgemployeegroup
commerceservices.organization.rights.edit.parent.groups=orgemployeegroup
commerceservices.organization.rights.view.groups=orgemployeegroup
```

For finer-grained access control, you can create your own usergroups and overwrite the properties in your extension. For example, `salesadmingroup`, `salesmanagergroup`, and `salesemployeegroup` are created in the `commerceorgsamplesaddon` extension. The following configuration is an example from `commerceorgsamplesaddon/project.properties`:

```
commerceservices.organization.rights.create.groups=salesadmingroup
commerceservices.organization.rights.edit.groups=salesadmingroup,salesmanagergroup
commerceservices.organization.rights.edit.parent.groups=salesadmingroup
commerceservices.organization.rights.view.groups=salesemployeegroup
```

Related Information

[Sample Data, User Rights, and Search Restrictions]

Sample Data, User Rights, and Search Restrictions

The `commerceorgsamplesaddon` AddOn includes sample data to give you an idea of what the Sales Organization feature looks like and how it works. The user rights and search restrictions that are configured in the sample data are a good starting point for custom implementations.

Sample Data

Sales Organization

The following table indicates the structure of the Sales Organization that is included in the Sales Organization sample data:

ID	Name	Parent Unit	Description	Line of Business
Africa	Africa sales region	None	Africa sales region	Trade
Nigeria	Nigeria sales region	Africa	Nigeria sales region	Trade
South Africa	South Africa sales region	Africa	South Africa sales region	Trade
Egypt	Egypt sales region	Africa	Egypt sales region	Trade
Americas	Americas sales region	None	Americas sales region	Trade
North America	North America sales region	Americas	North America sales region	Trade
Canada	Canada sales region	North America	Canada sales region	Trade
Quebec	Quebec sales region	Canada	Quebec sales region	Trade
Montreal North	Montreal North sales region	Quebec	Montreal North sales region	Trade
Montreal South	Montreal South sales region	Quebec	Montreal South sales region	Trade
USA	USA sales region	North America	USA sales region	Trade
Mexico	Mexico sales region	North America	Mexico sales region	Trade
South America	South America sales region	Americas	South America sales region	Trade
Asia	Asia sales region	None	Asia sales region	Trade
India	India sales region	Asia	India sales region	Trade
China	China sales region	Asia	China sales region	Trade
Japan	Japan sales region	Asia	Japan sales region	Trade
Europe	Europe sales region	None	Europe sales region	Trade
Europe West	Europe West sales region	Europe	Europe West sales region	Trade
Germany	Germany sales region	Europe West	Germany sales region	Trade
France	France sales region	Europe West	France sales region	Trade
UK	UK sales region	Europe West	UK sales region	Trade
Europe East	Europe East sales region	Europe	Europe East sales region	Trade
Poland	Poland sales region	Europe East	Poland sales region	Trade
Russia	Russia sales region	Europe East	Russia sales region	Trade

The following bulleted lists and sub-lists indicate the hierarchy in the structure of the sample data:

- Africa

    - Nigeria
    - South Africa
    - Egypt
- Americas
    - North America
        - Canada
            - Quebec
                - Montreal North
                - Montreal South
        - USA
        - Mexico
    - South America
- Asia
    - India
    - China
    - Japan
- Europe
    - Europe West
        - Germany
        - France
        - UK
    - Europe East
        - Poland
        - Russia

The **Line of Business** sample data includes the following entries:

- Trade
- Bank
- Industry
- Building
- Government
- Service

Sales Organization Roles

The following table describes the Sales Organization roles:

ID	Name	Description
employeegroup	employeegroup	This is the parent group for all other user groups in the sales organization.
salesemployeegroup	salesemployeegroup	Members of **salesemployeegroup** also belong to **employeegroup**. All members of the sales organization belong to **salesemployeegroup**.
salesadmingroup	salesadmingroup	Members of **salesadmingroup** have added admin privileges, and also belong to **salesemployeegroup**.
salesmanagergroup	salesmanagergroup	Members of **salesmanagergroup** have manager privileges, and also belong to **salesemployeegroup**.
salesapprovergroup	salesapprovergroup	Members of **salesapprovergroup** have approval permissions, and also belong to **salesemployeegroup**.

The following bulleted list indicates the hierarchy of the sales organization roles in the sample data:

- employeegroup
    - salesemployeegroup
        - salesadmingroup
        - salesmanagergroup
        - salesapprovergroup

Employees

You can log in to the Sales Organization perspective of Backoffice using any of the employee data included in the Sales Organization sample data. The username is `firstname.lastname@acme.com`.

The following table lists all the employee data that is included in the Sales Organization sample data:

First Name	Last Name	Role (Group)	Sales Unit	Assigned To
Brandon	Leclair	salesemployeegroup	Nigeria	Pronto Services
Janetta	Estepp	salesemployeegroup	Nigeria	Pronto Services
Yoshie	Dority	salesmanagergroup	Nigeria	Pronto Services
Glen	Hofer	salesapprovergroup	Nigeria	Pronto Services
Temeka	Meekins	salesemployeegroup	Canada	Rustic
Dionne	Siguenza	salesemployeegroup	Canada	Rustic
Hermelinda	Cusick	salesemployeegroup	Canada	Rustic
Karleen	Holub	salesemployeegroup	Canada	Rustic
Abraham	Mclane	salesmanagergroup	Canada	Rustic
Marvel	Fargo	salesapprovergroup	Canada	Rustic

First Name	Last Name	Role (Group)	Sales Unit	Assigned To
Darrin	Hesser	salesemployeegroup	Quebec	Services East, Services West
Jules	Hasson	salesemployeegroup	Quebec	Services East, Services West
Ula	Barragan	salesemployeegroup	Quebec	Services East, Services West
Juliane	Tickle	salesmanagergroup	Quebec	Services East, Services West
Lavone	Dupler	salesapprovergroup	Quebec	Services East, Services West
Chelsie	Steck	salesemployeegroup	Montreal North	Rustic Retail
Daniele	Sorber	salesemployeegroup	Montreal North	Rustic Retail
Yan	Shehorn	salesmanagergroup	Montreal North	Rustic Retail
Fern	Henline	salesapprovergroup	Montreal North	Rustic Retail
Debera	Spiller	salesemployeegroup	Montreal South	Rustic Services
Albert	Decastro	salesemployeegroup	Montreal South	Rustic Services
Piedad	Holdren	salesmanagergroup	Montreal South	Rustic Services
Bernardo	Coelho	salesapprovergroup	Montreal South	Rustic Services
Sade	Mcdougall	salesemployeegroup	USA	None
Latisha	Latimer	salesemployeegroup	USA	None
Tom	Ziebarth	salesemployeegroup	USA	None
Mara	Martino	salesemployeegroup	USA	None
Elizabeth	Juhlin	salesemployeegroup	USA	None
Sheilah	Duffin	salesmanagergroup	USA	None
Eusebio	Scharff	salesapprovergroup	USA	None
Alda	Kamaka	salesemployeegroup	Europe West	None
Anamaria	Coots	salesemployeegroup	Europe West	None
Lael	Garibay	salesmanagergroup	Europe West	None
Jamey	Sowa	salesapprovergroup	Europe West	None
Marie	Kempner	salesemployeegroup	Germany	None
Dietrich	Brand	salesemployeegroup	Germany	None
Dagmar	Fischer	salesemployeegroup	Germany	None
Dot	Cohan	salesemployeegroup	Germany	None
Dorthy	Geoghegan	salesemployeegroup	Germany	None
Angelyn	Lobaugh	salesmanagergroup	Germany	None
Tilda	Prisbrey	salesapprovergroup	Germany	None

First Name	Last Name	Role (Group)	Sales Unit	Assigned To
James	Davis	salesemployeegroup	UK	None
Amelia	Hill	salesemployeegroup	UK	None
Oliver	Baker	salesemployeegroup	UK	None
Emily	Bennett	salesemployeegroup	UK	None
Noah	Jenkins	salesemployeegroup	UK	None
Isabella	Jackson	salesmanagergroup	UK	None
Richard	Martin	salesapprovergroup	UK	None
Hanna	Andresen	salesemployeegroup	France	None
Ossie	Dilks	salesemployeegroup	France	None
Annabel	Golder	salesmanagergroup	France	None
Francie	Wildman	salesapprovergroup	France	None
Yuka	Kobayashi	salesemployeegroup	Japan	None
Shun	Watanabe	salesemployeegroup	Japan	None
Natsumi	Takahashi	salesemployeegroup	Japan	None
Keita	Tanaka	salesemployeegroup	Japan	None
Ayumi	Nakamura	salesemployeegroup	Japan	None
Yu	Yamamoto	salesmanagergroup	Japan	None
Mai	Matsumoto	salesapprovergroup	Japan	None
Aarav	Devi	salesemployeegroup	India	None
Arjun	Sewant	salesemployeegroup	India	None
Chandni	Devaraju	salesmanagergroup	India	None
Sandesh	Patwary	salesapprovergroup	India	None
Zhang	Wei	salesemployeegroup	China	None
Wang	Lei	salesemployeegroup	China	None
Liu	Yang	salesmanagergroup	China	None
Chen	Gao	salesapprovergroup	China	None
Vada	Rahm	salesadmingroup	Sales Organization	None

Customers

The following table lists the customer data that is included in the Sales Organization sample data:

ID	Name	Assigne To Sales Unit
Pronto	Pronto	Africa
Pronto Services	Pronto Services	Nigeria

ID	Name	Assigne To Sales Unit
Pronto Goods	Pronto Goods	South Africa
Rustic	Rustic	Canada
Rustic Retail	Rustic Retail	Montreal North
Rustic Services	Rustic Services	Montreal South
Services East	Services East	Quebec
Services West	Services West	Quebec

User Rights

There can be many use cases for the Backoffice Sales Organization perspective other than sales. For this reason, the `essentialdata` file in `commerceservices` (that is, `commerceservices/resources/impex/essentialdata_usergroups.impex`) only contains an **orgemployeegroup** that can access the Backoffice perspective.

The `commerceorgsamplesaddon` AddOn is specifically focused on the sales scenario and contains **salesemployeegroup**, **salesadmingroup**, **salesmanagergroup** and **salesapprovergroup**. The admin, manager, and approver groups are members of the **salesemployeegroup**. The **salesemployeegroup** extends **orgemployeegroup**, which has create, read, and write permissions on **OrgUnits** and **Employees**. The relationship between groups can be seen in the following impex example:

```
INSERT_UPDATE PrincipalGroupRelation;source(uid)[unique=true];target(uid)[unique=true]
;salesemployeegroup;orgemployeegroup;
;salesadmingroup;salesemployeegroup;
;salesmanagergroup;salesemployeegroup;
;salesapprovergroup;salesemployeegroup;
```

The sample data, including sales organization groups and user rights, are configured in `commerceorgsamplesaddon/import/common/user-groups.impex` as follows:

- A Sales Admin (**salesadmingroup** user) can create, view and edit Sales Units, but cannot delete.
- A Sales Manager (**salesmanagergroup** user) can edit Sales Unit, but cannot create or delete.
- A Sales Employee (**salesemployeegroup** user) can only view a Sales Unit, and cannot edit, create, or delete.
- A Sales Employee can only access the Backoffice Sales Organization perspective.

In terms of visibility, all users can only see their root unit and child units.

In terms of role restrictions:

- administrators can edit any role
- managers can edit other managers, approvers, and employees
- approvers and employees can display their units as read-only

Sales employees can only see the units they are assigned to. For example, the manager for North America might only see North America, Canada, United States, and Mexico sales units, but the root administrator will see North America, Central America, South America, Europe, etc. These restrictions also apply when editing an employee and assigning units.

Administrators are assigned to a root unit called **Acme**, instead of being given default access to all units. This way, different administrators can manage different unit hierarchies separately.

Managers are prevented from editing or removing employees who are administrators. Managers can create employees but cannot assign new employees to the administrator role. Managers also cannot remove themselves from a unit.

User Rights of salesadmingroup

The following table lists the user rights of the **salesadmingroup**:

	Create	Read	Update	Delete
OrgUnit	✓	✓	✓	⛔
Address	✓	✓	✓	⛔
Employee	✓	✓	✓	⛔
StoreEmployeeGroup	✓	✓	✓	⛔
B2BUnit	⛔	✓	⛔	⛔

User Rights of salesmanagergroup

The following table lists the user rights of the **salesmanagergroup**:

	Create	Read	Update	Delete
OrgUnit	⛔	✓	✓	⛔
OrgUnit.groups	⛔	✓	⛔	⛔
Address	⛔	✓	✓	⛔
Employee	⛔	✓	✓	⛔
StoreEmployeeGroup	⛔	✓	✓	⛔
B2BUnit	⛔	✓	⛔	⛔

User Rights of salesemployeegroup

The following table lists the user rights of the **salesemployeegroup**:

	Create	Read	Update	Delete
OrgUnit	⛔	✓	⛔	⛔
Address	⛔	✓	⛔	⛔
Employee	⛔	✓	⛔	⛔
StoreEmployeeGroup	⛔	✓	⛔	⛔
B2BUnit	⛔	✓	⛔	⛔

Search Restrictions

A member of **salesemployeegroup** can only see the employees from **salesemployeegroup**. This is accomplished with the search restriction `salesemployeegroupEmployeesVisibility`, which is configured in `commerceorgsamplesaddon/import/common/user-groups.impex` as follows:

```
#
# Add restriction on salesemployeegroup. salesemployeegroup shall only see the employees from salesemp
#
INSERT_UPDATE SearchRestriction;code[unique=true];name[lang=en];query;principal(UID);restrictedType(co
;salesemployeegroupEmployeesVisibility;A member of salesemployeegroup shall only see the employees fro
```

A sales employee can only see the latest version of quotes that are owned by a user linked to the sales employee's sales units. This is accomplished with the search restrictions `salesemployeegroupQuotesOrgUnitRestriction` and `salesemployeegroupQuotesMaxVersionRestriction`, which are configured in `commerceorgsamplesaddon/import/common/user-groups.impex` as follows:

```
#
# Restrict quotes so that sales employees can only see the latest versions of quotes that are linked t
#
INSERT_UPDATE SearchRestriction;code[unique=true];name[lang=en];query;principal(UID);restrictedType(co
;salesemployeegroupQuotesOrgUnitRestriction;A sales employee can only see quotes linked to their sales
;salesemployeegroupQuotesMaxVersionRestriction;A sales employee can only see the latest version of a q
```

For B2B scenarios, `salesemployeegroupQuotesOrgUnitRestriction` is overwritten in `commerceorgsamplesaddon/import/stores/powertools/user-groups.impex` to take business units on the buyer side into account:

```
#
# Overwriting the SearchRestriction of the the same name in /commerceorgsamplesaddon/import/common/use
#
INSERT_UPDATE SearchRestriction;code[unique=true];name[lang=en];query;principal(UID);restrictedType(co
;salesemployeegroupQuotesOrgUnitRestriction;A sales employee can only see quotes linked to their sales
```

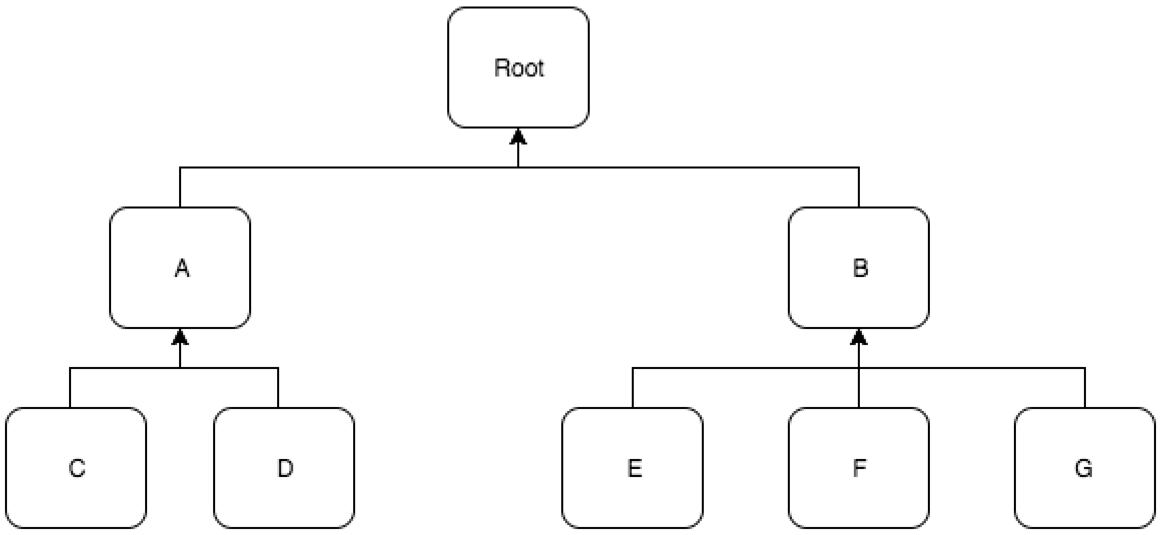
Organization Hierarchy Traversal

Sales Organization unit assignments include child units automatically for both sales units and buyer units. This is achieved using search restrictions.

Overview

For members of the **salesemployeegroup**, we want to limit the visibility of the **OrgUnit** type to the unit the employee is assigned to, as well as to all **OrgUnits** that are descendants of that unit. Globally limiting the visibility of a certain item type for a certain user group is achieved using search restrictions.

To find all **OrgUnits** in one branch of a hierarchy, every unit needs to know its place in the hierarchy, relative to the root of the organization. To illustrate, we'll use the simple hierarchy in the following diagram as an example:



For every unit in the tree, we can materialize the path of traversal from the root to the unit as follows:

Unit	Path
Root	/Root
A	/Root/A
B	/Root/B
C	/Root/A/C
D	/Root/A/D
E	/Root/B/E
F	/Root/B/F
G	/Root/B/G

The path for each unit starts with the path of the parent unit. This makes it possible to write a simple **Flexible Search Query** that returns all units in a branch, including the start node, as follows:

```
SELECT {unit:pk} FROM {OrgUnit AS unit} WHERE {unit:path} LIKE CONCAT(?startUnitPath, '%')
```

Using the path of, for example, unit B (that is, `/Root/B`) as the `startUnitPath` parameter in the query returns all units whose path starts with `/Root/B` – in other words, B, E, F, and G. This simple query doesn't need any JOINs and still works for arbitrary depths. And conveniently, using the path of the Root unit will return all units in the hierarchy.

Traversal Path Generation

Three events trigger the traversal path generation for `OrgUnitModel` and `B2BUnitModel`:

- Initializing or updating the system
- Running the `generateOrgUnitPathsJob` cron job for `OrgUnitModel` or the `generateB2BUnitPathsCronJob` cron job for `B2BUnitModel`
- Changing the parent unit

By default, the `OrgUnitModel` and `B2BUnitModel` item path generation is turned on. You can turn it off by changing the following property to **false** in the `project.properties` file of the `commerceservices` extension:

```
commerceservices.org.unit.path.generation.enabled=false
```

→ Tip

The traversal path generation may affect performance when initializing or updating the system. It is recommended to turn off the path generation before you initialize or update, and then turn it back on once the initialization or update is completed.

Implementation

The implementation for this feature affects the `commerceservices` and `b2bcommerce` extensions, as well as the `commerceorgsamplesaddon`, as described in the following sections.

commerceservices

The following sections describe the data model and service layer of the implementation in the `commerceservices` extension.

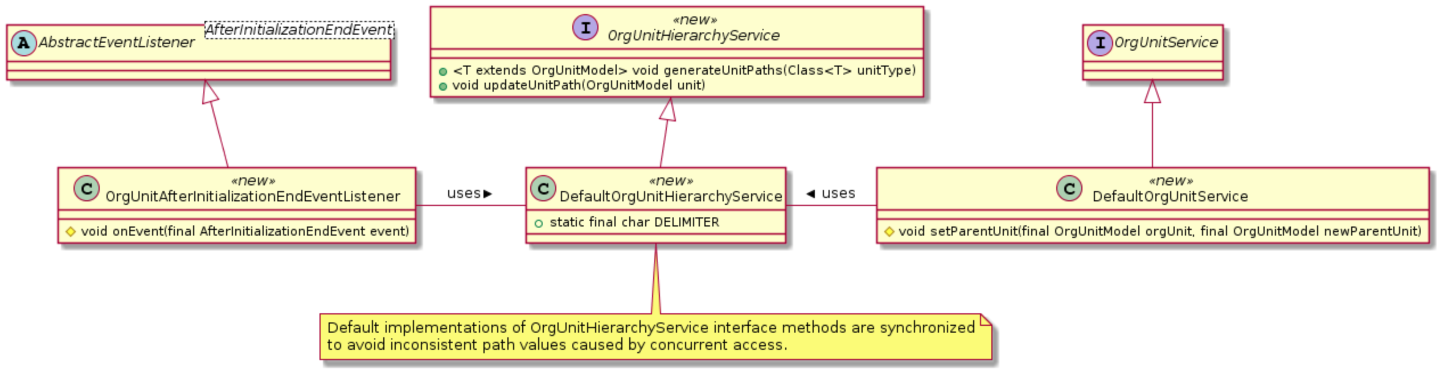
Data Model

To store the materialized path of a unit, an attribute has been added to the **OrgUnit** type, and an index has also been created for it, as illustrated in the following example from `/commerceservices/resources/commerceservices-items.xml`:

```
<itemtype code="OrgUnit" jaloclass="de.hybris.platform.commerceservices.jalo.OrgUnit" extends="Company
    <attributes>
        <attribute qualifier="active" type="java.lang.Boolean">
            <modifiers read="true" write="true" search="true" optional="false" />
            <defaultvalue>java.lang.Boolean.TRUE</defaultvalue>
            <persistence type="property" />
        </attribute>
        <attribute qualifier="path" type="java.lang.String">
            <description>Flat representation of the path of traversal to reach the OrgUnit from the rc
            <custom-properties>
                <property name="hiddenForUI">
                    <value>Boolean.TRUE</value>
                </property>
            </custom-properties>
            <persistence type="property" />
        </attribute>
    </attributes>
    <indexes>
        <index name="pathIndex">
            <key attribute="path"/>
        </index>
    </indexes>
</itemtype>
```

Service Layer

The following diagram illustrates the implementation of the service layer in the `commerceservices` extension.


AbstractEventListener
AfterInitializationEndEvent
«new» OrgUnitHierarchyService
<T extends OrgUnitModel> void generateUnitPaths(Class<T> unitType)
void updateUnitPath(OrgUnitModel unit)
OrgUnitService
«new» OrgUnitAfterInitializationEndEventListener
void onEvent(final AfterInitializationEndEvent event)
uses ▶
«new» DefaultOrgUnitHierarchyService
static final char DELIMITER
◀ uses
«new» DefaultOrgUnitService
void setParentUnit(final OrgUnitModel orgUnit, final OrgUnitModel newParentUnit)
Default implementations of OrgUnitHierarchyService interface methods are synchronized to avoid inconsistent path values caused by concurrent access.

The following table describes the classes that were implemented in the service layer to enable the organization hierarchy traversal feature.

Class	Description
`DefaultOrgUnitHierarchyService`	The `DefaultOrgUnitHierarchyService` is the default implementation of the `OrgUnitHierarchyService` interface, providing thread-safe method implementations for the generation of **OrgUnit** path values, and the update of path values for all **OrgUnits** that belong to one branch of an organization.
`OrgUnitAfterInitializationEndEventListener`	After initialization, the `OrgUnitAfterInitializationEndEventListener` triggers the generation of path values for all **OrgUnits** by calling the `OrgUnitHierarchyService`.
`DefaultOrgUnitService`	The `DefaultOrgUnitService` is the default implementation of the `OrgUnitService` interface, which calls `OrgUnitHierarchyService` to make sure that the path values for all **OrgUnits** that belong to the branch of a given **OrgUnit** are updated when the parent unit of this **OrgUnit** changes.

i **Note**

As long as `OrgUnitService` is used to manipulate **OrgUnit** items, the path values of all **OrgUnits** in the system remain consistent. It is important to note that this does not cover cases such as a system administrator changing the parent of an **OrgUnit** from Backoffice. For such cases, `generateOrgUnitPathsJob` has been configured and can be triggered from Backoffice at any time to regenerate path values for all **OrgUnits**.

b2bcommerce

The following changes have been made to the `b2bcommerce` extension to implement the organization hierarchy traversal feature:

- A B2B-specific version of `OrgUnitAfterInitializationEndEventListener` has been configured to trigger the generation of path values for all **B2BUnit** items in the system after initialization.
- `DefaultB2BUnitFacade` has been updated to call `OrgUnitHierarchyService` to keep path values for **B2BUnit** items consistent.
- `generateB2BUnitPathsJob` can be triggered from Backoffice at any time to regenerate path values for all **B2BUnits**.

commerceorgsamplesaddon

Based on the enhanced data model, we are able to create a search restriction that limits the visibility of **OrgUnits** for a sales employee to the ones in their organizational branch. The following is an example from `/commerceorgsamplesaddon/resources/commerceorgsamplesaddon/import/common/user-groups.impex`:

```
INSERT_UPDATE SearchRestriction;code[unique=true];name[lang=en];query;principal(UID);restrictedType(co
;salesemployeegroupOrgUnitVisibility;A member of salesemployeegroup shall only see sales units in thei
```

The following is an example of the query in pretty-print:

```
NOT EXISTS
(
    {{
        SELECT {unit:pk} FROM {OrgUnit! as unit}
        WHERE {unit:pk}={item:pk}
    }}
)
OR
EXISTS
(
    {{
        SELECT {unit:pk} FROM
        {
            Employee as empl JOIN PrincipalGroupRelation as rel
            ON {empl:pk} = {rel:source}
            JOIN OrgUnit as unit
            ON {unit:PK} = {rel:target}
        }
        WHERE {empl:pk} = (?session.user) AND {item:path} LIKE CONCAT({unit:path},'%')
    }}
)
```

The `NOT EXISTS` clause makes sure that this restriction is only applied to items of type `OrgUnit`, but not for items of a type extending `OrgUnit`. The second part of the query checks whether an `OrgUnit` exists that the current session user (such as the sales employee) is assigned to, and is an ancestor of the item in the unit hierarchy. This way, only `OrgUnits` in the sales employee's branch are returned.

ASM Integration

Sales representatives can interact with customers using the Assisted Service Module (ASM). The ASM customer list framework is also made use of in the sales organization module.

There are two ways that the sales organization feature integrates with the Assisted Service Module (ASM). First, the ASM feature itself is made available to sales employees. Therefore, all sales employees can authenticate in ASM and support customers with their purchases. Second, the sales organization module leverages the customer list framework from ASM. For complete documentation about the customer list feature, see the Related Links section, below.

The customer list feature allows an ASM user to invoke lists of customers that are used to find customers to support. The sales organization module leverages the customer list feature as follows: a sales representative that authenticates in ASM can invoke the customer list feature and use the **B2B Customer List**. The **B2B Customer List** provides the sales representative with a list of customers that are assigned to him in the sales organization module. This way, the sales representative has a convenient way to access his or her customers and start an ASM session.

Search Strategy Logic

At the heart of the ASM integration implementation is the customer list search strategy. This is the logic that determines which customers end up in this specific list. In the case of the **B2B Customer List**, the search logic starts with the sales representative currently using ASM. From there, the logic looks up the sales organization that the sales representative belongs to. From the sales organization, we look up the customer B2B units that are associated with the sales organization. Finally, the individual customers that belong to the customer units are returned to populate the customer list. This logic is implemented in `de.hybris.platform.b2b.strategies.impl.B2BCustomerListSearchStrategy`.

Search Strategy Configuration

The search strategy is defined as a Spring bean and is then registered in the list of search strategies handled by the ASM Customer List feature. These two steps are done through Spring configurations. Lastly, to allow sales representatives to use this customer list, it needs to be available in the ASM user interface. This last part is achieved using an impex statement. The `B2BCustomerListSearchStrategy` is defined in `b2bcommerce-spring.xml` as follows:

```
<bean id="b2bCustomerListSearchStrategy" class="de.hybris.platform.b2b.strategies.impl.B2BCustomerList
                <property name="userService" ref="userService" />
                <property name="b2bCommerceUserService" ref="b2bCommerceUserService" />
                <property name="orgUnitDao" ref="orgUnitDao"/>
        </bean>
```

The following snippet, also from `b2bcommerce-spring.xml` , shows how the Spring bean is registered in the customer list feature:

```
<bean id="b2bCustomerListSearchStrategyMergeDirective" depends-on="customerListSearchStrategyMap" pare
                <property name="key" value="B2B"/>
                <property name="value" ref="b2bCustomerListSearchStrategy" />
        </bean>
```

The above configuration links the list key `B2B` to the bean id `b2bCustomerListSearchStrategy`.

The following impex statement enables the B2B customer list:

```
# Associate members of salesemployeegroup to the B2B Customer List search strategy.
INSERT_UPDATE CustomerList;uid[unique=true];locname[lang=en];implementationType;priority;members(uid);
;b2bCustomerList;B2B Customer List;B2B;0;salesemployeegroup
```

This impex statement links the the sales representatives (that is, members of **salesemployeegroup**) to the B2B customer list configuration key `B2B` that is defined above.

Note

If the default behaviour of the `B2BCustomerListSearchStrategy` does not match your requirements, the customer list feature is extensible, so you can implement your own customer list. For more information, refer to the following documentation:

- commercefacades Extension
- commerceservices Extension

Rules and Limitations

The following sections describe the rules and limitations with regards to which customers a sales representative can interact with when using the sales organization feature.

Sales Representatives and Customer Association

Sales representatives are assigned to customers through a sales organization. A sales representative is assigned to the customers who belong to the same sales organization as the sales representative. If we take the Quebec sales region as an example (as provided in the sample data that is configured in the **commerceorgsamplesaddon** AddOn), we see that there is one tab for employees and one tab for customers. The customers that belong to the units listed in the customer tab are assigned to the sales representatives in the employee tab. This is the rule by which the B2B Customer List Search Strategy selects the customers for the **B2B Customer List**. The Sales Organization module allows the sales administrator to organize sales organizations into a tree structure. Every sales organization can potentially have a parent sales organization. The B2B Customer List Search Strategy does not assume an implicit relationship between sales representatives from a sales organization and the customers of its child sales organizations. In other words, the position of a sales organization in the tree structure does not change the results that a sales representative will see in the B2B Customer list.

Customer Visibility

The B2B Customer List is a convenience feature for the sales representative to easily access the customers he or she is assigned to. However, there are no restrictions as to which customers a sales representative can support in ASM. The sales representative still has access to all the customers through the ASM customer search.

Adding Columns to the Customer List

The following procedure describes how to add columns to the customer list. For information on the generic mechanism that allows you to add columns to the customer list, see commercefacades Extension.

1. Configure the columns map in the relevant Spring file. The keys of the map are the `CustomerList.additionalColumnsKeys` of the added columns, and the values of the map are the EL syntax of the `CustomerData` DTO. The following is an example from `b2bcommercefacades-spring.xml`:

```
<bean id="b2bCustomerListAdditionalColumnsMergeDirective" depends-on="customerListAdditionalColu
        <property name="sourceMap">
                <map>
                        <entry key="UNIT" value="unit.name" />  <!-- what happens behind the sce
                        <entry key="CURRENCY" value="currency.isocode" /> <!-- what happens behi
                </map>
        </property>
</bean>
```

2. Import the following impex:

```
INSERT_UPDATE CustomerList;uid[unique=true];locname[lang=en];implementationType;priority;members
;b2bCustomerList;My Organization Customers;B2B;0;salesemployeegroup;UNIT,CURRENCY
```

   You can import the impex using Administration Console or any other method for importing impex into the system. The above example is taken from `commerceorgsamplesaddon/import/stores/powertools/user-groups.impex`.

3. Configure properties for the added column keys, as follows:

```
text.customerList.header.${columnKey}
```

   The following is an example from `commerceorgsamplesaddon/acceleratoraddon/web/webroot/WEB-INF/messages/base_en.properties`:

```
text.customerList.header.unit.name=Account
text.customerList.header.currency.isocode=Currency
```

   In this example, the column keys are `unit.name` and `currency.isocode`.

Related Information

assistedservicestorefront Extension

assistedservicefacades Extension

commercefacades Extension

commerceservices Extension

Interceptor Validation in the Organization Unit

During the process of creating and updating an `OrgUnit`, `OrgUnitModelValidateInterceptor.java` checks certain prerequisites, and throws an exception if the following prerequisites are not fulfilled:

- An `OrgUnit` cannot be a member of more than one `OrgUnit`. In other words, an `OrgUnit` can only have one parent.

  i Note

  This validation checks the exact type instead of performing an `instanceof` check. This means it is possible for an `OrgUnit` to be a member of more than one group extending `OrgUnit`.

- An `OrgUnit` whose parent is disabled cannot be enabled.

The default Spring configuration for the interceptor validation is defined in `commerceservices-spring.xml` as follows:

```
<bean id="orgUnitModelValidateInterceptor" class="de.hybris.platform.commerceservices.organization.int
        <property name="orgUnitService" ref="orgUnitService"/>
        <property name="userService" ref="userService"/>
        <property name="l10NService" ref="l10nService"/>
        <property name="modelService" ref="modelService"/>
</bean>

<bean id="orgUnitModelValidateInterceptorMapping" class="de.hybris.platform.servicelayer.interceptor.i
        <property name="interceptor" ref="orgUnitModelValidateInterceptor"/>
        <property name="typeCode" value="OrgUnit"/>
</bean>
```

You can adjust the code directly, you can write your own implementation, or you can extend the existing implementation and configure it to be used instead of the default implementation. This gives the option to have multiple interceptors mapped to the same type.

Related Information

Interceptors

Configuring Sales Organization in Backoffice

In Backoffice, the Sales Organization perspective allows you to create and modify sales units, add sales employees, modify employee details, and manage the relationships between sales units and customers.

Sales units contain sales employees. In the sample data that is included with the `commerceorgsamplesaddon` AddOn, each sales unit includes sales employees with one of the following roles: sales employee, sales administrator, sales manager, or sales approver.

In Backoffice, you can create new sales units, and you can modify sales units by adding or removing employees. You can also edit employee details.

Logging in to Backoffice Sales Organization Perspective

Logging in to the Sales Organization perspective in Backoffice allows you to create and modify sales units, add sales employees, modify employee details, and manage the relationships between sales units and customers.

Locating and Viewing a Sales Unit

Sales units contain information about the sales unit itself, as well as information about which employees and customers are associated with it.

Editing a Sales Unit

Learn how to edit a sales unit in Backoffice.

Disabling a Sales Unit

Learn how to disable a sales unit in Backoffice.

Creating a New Sales Unit

Learn how to add a new sales unit in Backoffice.

Creating a New Employee

Learn how to add a new employee in Backoffice.

Adding an Employee to a Sales Unit

Learn how to add an employee to a sales unit in Backoffice.

Editing an Employee

Learn how to edit an employee in Backoffice.

[Removing an Employee from a Sales Unit]
Learn how to remove an employee from a sales unit in Backoffice.
[Adding an Account to a Sales Unit]
Learn how to add an account to a sales unit in Backoffice.
[Editing an Account]
Accounts cannot be edited from the Sales Organization perspective in Backoffice.
[Removing an Account from a Sales Unit]
Learn how to remove an account from a sales unit in Backoffice.

Logging in to Backoffice Sales Organization Perspective

Logging in to the Sales Organization perspective in Backoffice allows you to create and modify sales units, add sales employees, modify employee details, and manage the relationships between sales units and customers.

Procedure

1. Open the Backoffice login page in your web browser.

   If you are accessing Backoffice from a local installation, you can open the Backoffice login page by navigating to this URL: `https://localhost:9002/backoffice/login.zul`.

2. Enter login credentials in the **Username** and **Password** fields.

   In this case, log on as `vada.rahm@acme.com`. This user has administration rights in the Sales Organization perspective. These credentials are included with the `commerceorgsamplesaddon` AddOn.

3. Click **Login**.

   The Sales Organization perspective appears.

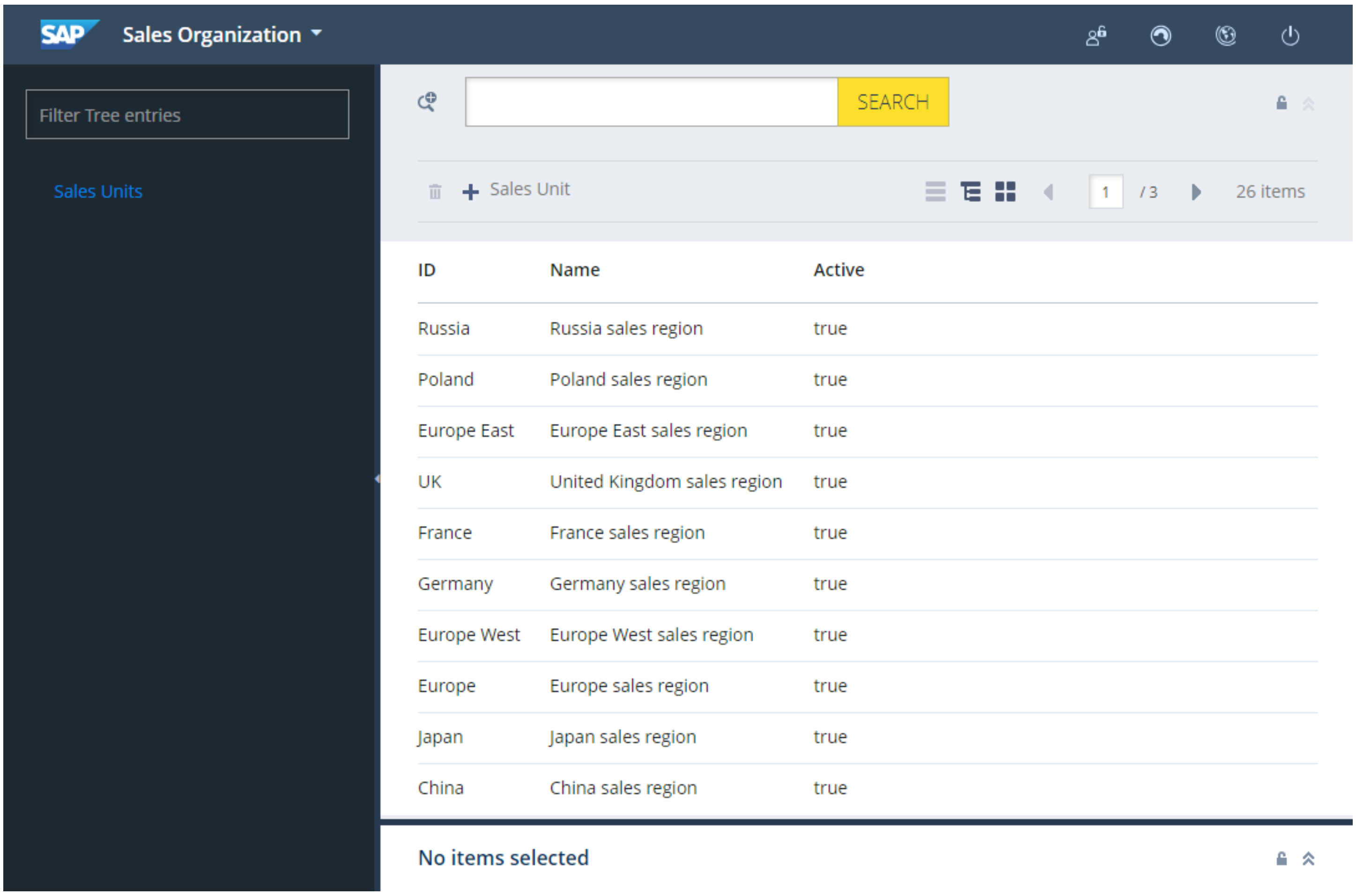
4. In the left navigation pane, click **Organization Units**, then click **Sales**.

   The Sales Units appear in the center pane of Backoffice.

Locating and Viewing a Sales Unit

Sales units contain information about the sales unit itself, as well as information about which employees and customers are associated with it.

You can locate a sales unit by searching for it in the **Search** bar at the top of the Sales Organization perspective, or you can scan through the list of sales units that are listed in the center pane when **Sales** is selected in the left navigation panel, as shown in the following image:



ID	Name	Active
Russia	Russia sales region	true
Poland	Poland sales region	true
Europe East	Europe East sales region	true
UK	United Kingdom sales region	true
France	France sales region	true
Germany	Germany sales region	true
Europe West	Europe West sales region	true
Europe	Europe sales region	true
Japan	Japan sales region	true
China	China sales region	true

No items selected

You can view a sales unit by clicking on its name.

When the **List View** button ☰ is selected, you can order the views by **ID**, **Name**, or state (**Active**) by clicking on the respective column.

When the **Tree View** button is selected, you can click the arrow next to each sales unit to see additional hierarchical information about the sales unit. When you expand the **All Groups** category, you see all the parent groups that the selected sales unit belongs to. When you expand the **Groups** category, you see the immediate parent that the selected sales unit belongs to. When you expand the **Members** category, you see all the sales units, customers, and sales employees that are the immediate children of the selected sales unit. In other words, you can see one level down in the hierarchy when you expand the **Members** category.

Editing a Sales Unit

Learn how to edit a sales unit in Backoffice.

Context

Users that belong to **salesadmingroup** or **salesmanagergroup**can edit sales units.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see Logging in to Backoffice Sales Organization Perspective.

2. Locate and select the sales unit you want to edit.

   For more information, see Locating and Viewing a Sales Unit.

3. Edit any field in the General, Employees or Customers tabs.

   As soon as you make a change to any of the fields, the Save button becomes active.

4. Click Save.

Disabling a Sales Unit

Learn how to disable a sales unit in Backoffice.

Context

Users that belong to **salesadmingroup** or **salesmanagergroup**can disable sales units.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see [Logging in to Backoffice Sales Organization Perspective]().

2. Locate and select the sales unit you want to edit.

   For more information, see [Locating and Viewing a Sales Unit]().

3. Click the Disable Unit button Disable Unit located above the General tab.

   A warning appears, stating that disabling the sales unit will disable all sub units, and that undoing this must be done manually on every descendant.

4. Click Yes in the warning dialog box.

   The sales unit is disabled. You may need to click Refresh to see that the sales unit is disabled.

Creating a New Sales Unit

Learn how to add a new sales unit in Backoffice.

Context

Users that belong to **salesadmingroup** can create new sales units. No user has the permissions to delete a sales unit (also know as OrgUnit) in Backoffice.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see [Logging in to Backoffice Sales Organization Perspective]().

2. Click the Sales Unit button + Sales Unit .

   The Create New Sales Unit wizard appears, starting with the Essential Sales Unit Information page.

3. Enter information in the ID and Name fields.

   For example, to remain consistent with the sample data included in the `commerceorgsamplesaddon` AddOn, you could enter `Antarctica` as the ID, and `Antarctica sales region` as the Name.

4. Click Next.

   The Optional Sales Unit Information page appears.

5. Enter information in the **Description**, **Line of business**, and **Parent Unit** fields.

   The **Description** can be anything you like. To continue with our example, you could enter `Antarctica sales region`. The **Line of business** is a drop-down list offering a range of business domains. You can also select **n/a** (not applicable), or simply leave it blank. The **Parent Unit** field allows you to make the new sales unit a child of another, existing sales unit. If you leave this field blank, the new sales unit is created at the same level as any other top-level parent sales units that exist in the sales organziation.

6. Click **Done**.

Creating a New Employee

Learn how to add a new employee in Backoffice.

Context

Users that belong to **salesadmingroup** can create new employees. No user has the permissions to delete a sales unit (also know as OrgUnit) in Backoffice.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see Logging in to Backoffice Sales Organization Perspective.

2. Locate and select the sales unit that you want to add a new employee to.

   For more information, see Locating and Viewing a Sales Unit.

3. Click the **Employees** tab.

   If the selected sales unit already contains employees, a search bar appears below the list of employees. Otherwise, a search bar appears as the only element in the **Employees** tab.

4. Click in the search bar.

   The option to **Create new Employee** appears above the search bar.

5. Click **Create new Employee**.

   The **Create New Employee** wizard appears.

6. Enter information in the **ID** and **Name** fields.

   For example, to remain consistent with the sample data included in the `commerceorgsamplesaddon` AddOn, you could enter an email address in the **ID** field, and first name and last name in the **Name** field.

7. Click in the **Roles** field to assign a role to the employee.

   When you click in the **Roles** field, you can select from the available roles that appear in a drop-down list below the field.

   After assigning a role to the employee, another field appears below the initial **Roles** field, allowing you to assign additional roles to the employee, if needed.

8. Assign a sales unit to the employee.

   The **Sales Unit** field lists the current sales unit by default. You can assign the employee to additional sales units by clicking in the blank field below the **Sales Unit** field and then searching for the desired sales unit. You can also remove any sales unit by hovering over the name of the sales unit you wish to remove, and then clicking the red "x" that appears in the right side of that sales unit's field.

   Clicking the **Options** button (three horizontal dots) to the right of the blank field below the **Sales Unit** field launches the **Reference Search** window. For best results, set the **Global Operator** field to **And**. To select an item from the search results, click on the gray checkmark to the left of the item so that the checkmark turns blue. You can then click the **Select** button.

9. When you have finished entering all necessary information, click **Done**.

Adding an Employee to a Sales Unit

Learn how to add an employee to a sales unit in Backoffice.

Context

Users that belong to **salesadmingroup** or **salesmanagergroup**can add employees to sales units.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see [Logging in to Backoffice Sales Organization Perspective].

2. Locate and select the sales unit you want to add an employee to.

   For more information, see [Locating and Viewing a Sales Unit].

3. Click the **Employees** tab.
4. In the search bar, start typing the name of the employee you want to add.
5. When you have found the employee you want to add, select the employee and click **Save**.

Editing an Employee

Learn how to edit an employee in Backoffice.

Context

Users that belong to **salesadmingroup** or **salesmanagergroup**can edit employees.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see [Logging in to Backoffice Sales Organization Perspective].

2. Locate and select the sales unit that contains the employee you want to edit.

   For more information, see [Locating and Viewing a Sales Unit].

3. Click the **Employees** tab.
4. In the row of the employee you want to edit, click the options button, which appears as three vertical dots ⋮, then select **Edit Details**.
5. Edit any field in the various tabs that appear.

   As soon as you make a change to any of the fields, the **Save** button becomes active.

   Clicking in any field will allow you to search for, and select from, all of the available options for that particular field. For example, by clicking in the blank field below the **Assigned Sales Units** field, a drop-down list appears, allowing you to paginate through all of the available sales units. You can also enter a search term in the field to narrow down the search results.

   Clicking the **Options** button (three horizontal dots) to the right of most fields will, in most cases, launch the **Reference Search** window. For best results, set the **Global Operator** field to **And**. To select an item from the search results, click on the gray checkmark to the left of the item so that the checkmark turns blue. You can then click the **Select** button.

6. Click **Save**.

> **Note**
> You may have to navigate away from the sales unit and return to it before seeing the change appear.

Removing an Employee from a Sales Unit

Learn how to remove an employee from a sales unit in Backoffice.

Context

Users that belong to **salesadmingroup** or **salesmanagergroup**can remove employees from sales units.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see Logging in to Backoffice Sales Organization Perspective.
2. Locate and select the sales unit that contains the employee you want to remove.

   For more information, see Locating and Viewing a Sales Unit.
3. Click the **Employees** tab.
4. In the row of the employee you want to remove, click the options button, which appears as three vertical dots ⋮ , then select **Remove**.

   The employee is removed from the sales unit.

Adding an Account to a Sales Unit

Learn how to add an account to a sales unit in Backoffice.

Context

Users that belong to **salesadmingroup** or **salesmanagergroup**can add accounts to sales units.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see Logging in to Backoffice Sales Organization Perspective.
2. Locate and select the sales unit you want to add an account to.

   For more information, see Locating and Viewing a Sales Unit.
3. Click the **Accounts** tab.
4. In the search bar, start typing the name of the account you want to add.
5. When you have found the account you want to add, select the account and click **Save**.

Editing an Account

Accounts cannot be edited from the Sales Organization perspective in Backoffice.

An account in the Sales Organization perspective is the equivalent of a B2B Unit in the B2B Commerce perspective. To edit a B2B Unit, you must log in to Backoffice as an administrator and modify the B2B Unit from the B2B Commerce perspective.

For more information, see Working with B2B Units.

Removing an Account from a Sales Unit

Learn how to remove an account from a sales unit in Backoffice.

Context

Users that belong to **salesadmingroup** or **salesmanagergroup**can remove accounts from sales units.

Procedure

1. Log in to Backoffice Sales Organization perspective.

   For more information, see Logging in to Backoffice Sales Organization Perspective.

2. Locate and select the sales unit that contains the account you want to remove.

   For more information, see Locating and Viewing a Sales Unit.

3. Click the **Accounts** tab.

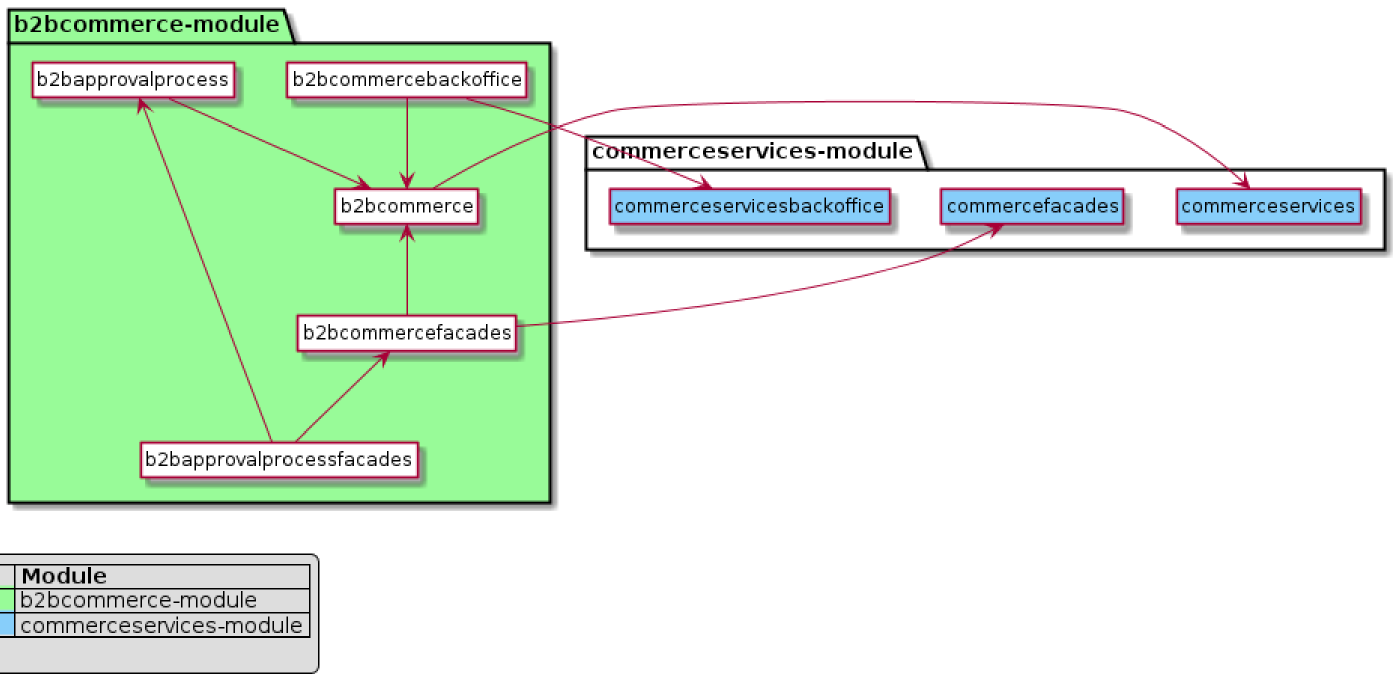
4. In the row of the account you want to remove, click the options button, which appears as three vertical dots ⋮ , then select **Remove**.

   The account is removed from the sales unit.

B2B Commerce Architecture

B2B Commerce is a set of extensions providing business-to-business functionality to SAP Commerce.

Dependencies



Dependencies Diagram

Recipes

The following recipes contain the B2B Commerce module:

- `b2b_acc_plus`
- `b2b_c4c`
- `b2b_china`
- `b2c_b2b_acc_cpq`
- `b2c_b2b_acc_dp`
- `b2c_b2b_acc_oms`
- `sap_aom_som_b2b_b2c`
- `sap_oms_aom_b2b_b2c`

For further information, see Installer Recipes.

Extensions

The B2B Commerce module consists of the following extensions:

- b2bapprovalprocess Extension
- b2bapprovalprocessfacades Extension
- b2bcommerce Extension
- b2bcommercefacades Extension
- b2bcommercebackoffice Extension

b2bapprovalprocess Extension

The `b2bapprovalprocess` extension provides two example approval processes and permissions with sample code in the source code. They can be used and referenced as a guideline for custom implementations, or as a starting set to be extended, or to deliver detailed information on implementation details that are beyond the scope of the technical documentation.

b2bapprovalprocessfacades Extension

The `b2bapprovalprocessfacades` provides a number of facades that are backed by the services in the `b2bapprovalprocess` extension.

b2bcommerce Extension

The `b2bcommerce` extension enables you to provide B2B functionality to your organization.

b2bcommercefacades Extension

The `b2bcommercefacades` extension provides a number of facades that are backed by the services in the `b2bcommerce` extension.

b2bapprovalprocess Extension

The `b2bapprovalprocess` extension provides two example approval processes and permissions with sample code in the source code. They can be used and referenced as a guideline for custom implementations, or as a starting set to be extended, or to deliver detailed information on implementation details that are beyond the scope of the technical documentation.

i **Note**

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

A business process describes a sequence of connected steps. To manage a process automatically or manually you need to identify and define it. This document provides an example of how to create such a process in scope of order management. The business process is modeled as a flowchart with loops and parallel paths joining a number of actions.

Find information on how to create an order approval process using the Backoffice Administration Cockpit in Managing Order Permissions in Backoffice.

For more information on workflow refer to Workflow Overview.

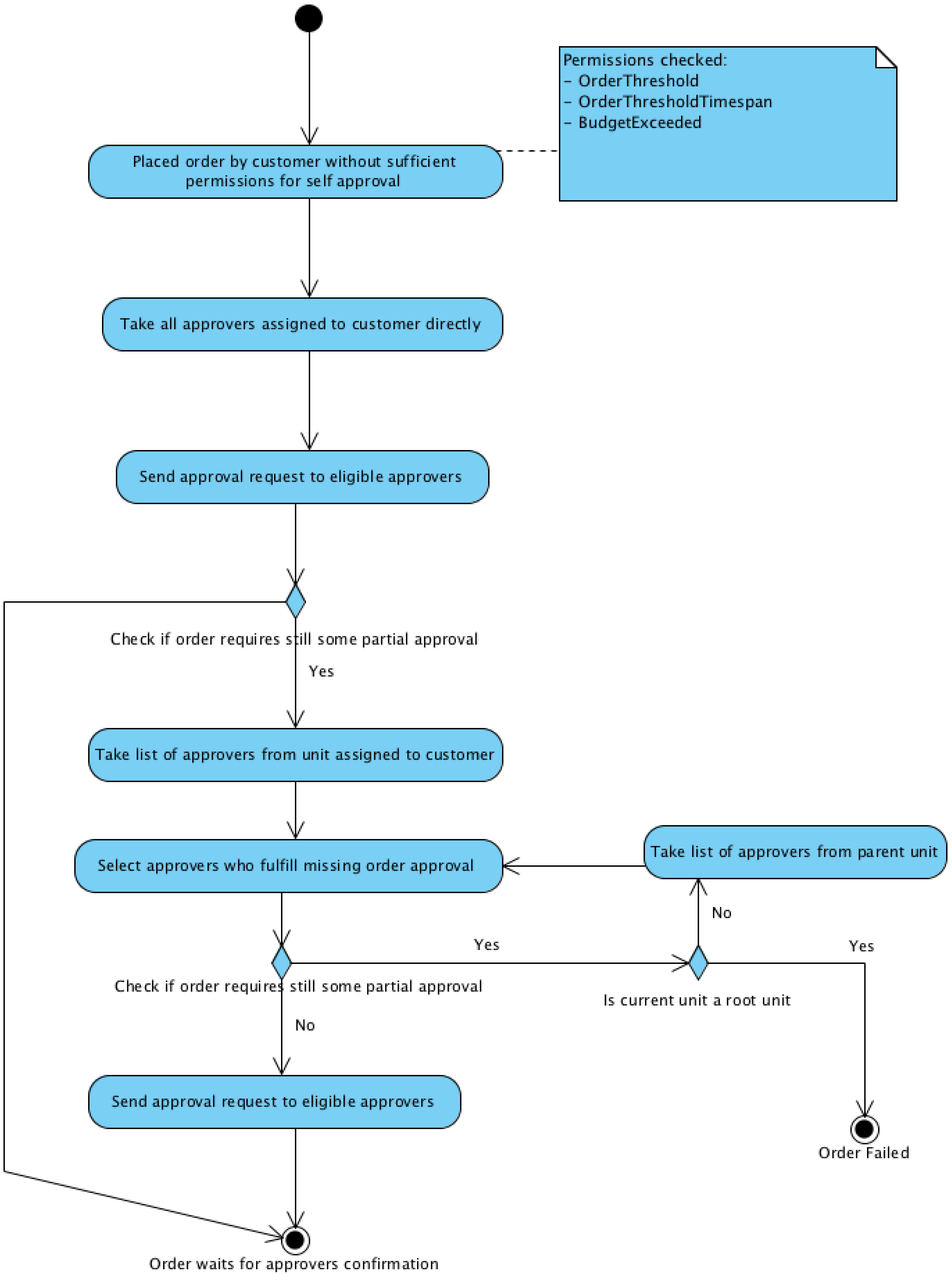
Definition of the Order Approval Process

To define an order approval process you need to know how the process looks like and what steps are necessary to complete it. This also includes the actions to take in case an error occurs during the process.

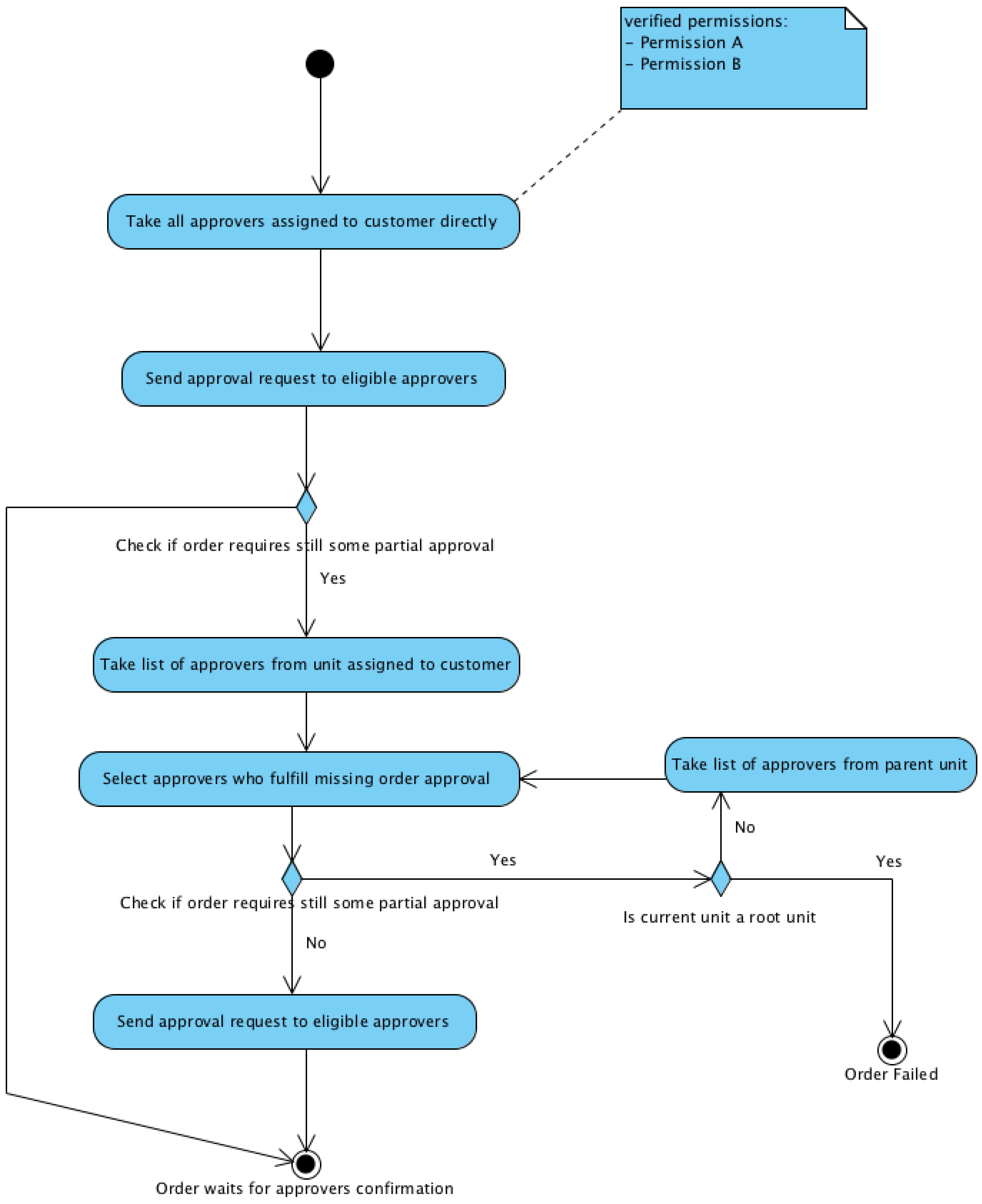
An example order approval process is defined in the `b2bapprovalprocess` extension in an XML format. It defines the process an order goes through within a B2B scenario to get fulfilled by being approved or becomes eligible for resubmission if rejected by an approver.

There are two approval processes provided by the extension: **Simple Approval** process and **Two Pair Of Eyes**.

This diagram illustrates the Simple Approval process.


Permissions checked:
- OrderThreshold
- OrderThresholdTimespan
- BudgetExceeded
Placed order by customer without sufficient permissions for self approval
Take all approvers assigned to customer directly
Send approval request to eligible approvers
Check if order requires still some partial approval
Yes
Take list of approvers from unit assigned to customer
Select approvers who fulfill missing order approval
Take list of approvers from parent unit
No
Yes
Yes
Check if order requires still some partial approval
Is current unit a root unit
No
Send approval request to eligible approvers
Order Failed
Order waits for approvers confirmation

This diagram illustrates the Two Pair of Eyes Approval process.


verified permissions:
- Permission A
- Permission B
Take all approvers assigned to customer directly
Send approval request to eligible approvers
Check if order requires still some partial approval
Yes
Take list of approvers from unit assigned to customer
Select approvers who fulfill missing order approval
Take list of approvers from parent unit
No
Yes
Yes
Check if order requires still some partial approval
Is current unit a root unit
No
Send approval request to eligible approvers
Order Failed
Order waits for approvers confirmation

Placing an Order

When an order is created in the SAP Commerce system by a user, who is of type **B2BCustomer**, a **B2BOrderApproval** process is triggered.

The process is responsible for verification if the order can go to the warehouse for fulfillment. The user permissions are checked and if all are satisfied, the order is approved automatically. The `b2bapprovalprocess` extension comes with two sets of default permissions. The first set of permission are used with the Simple Order Approval process. They are:

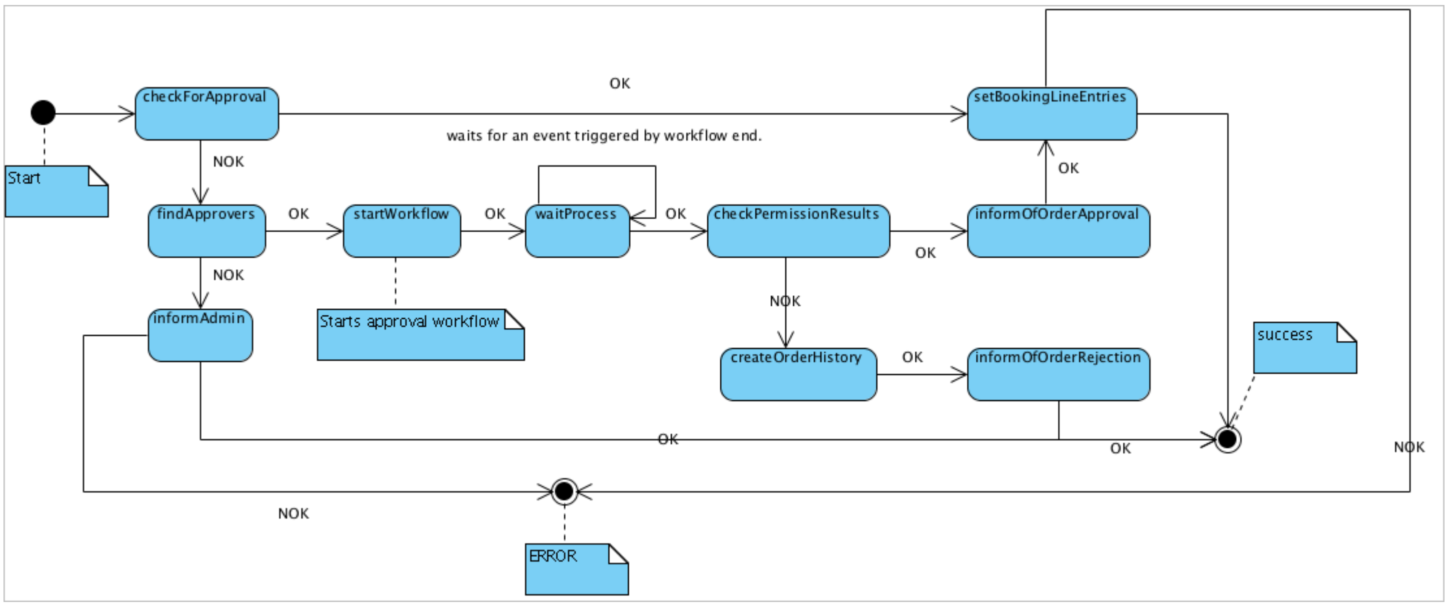
- B2BOrderThresholdPermission

- B2BOrderThresholdTimespanPermission
- B2BBudgetExceededPermission.

The second set of default permissions is used with the Two Pair of Eyes Approval process. They are:

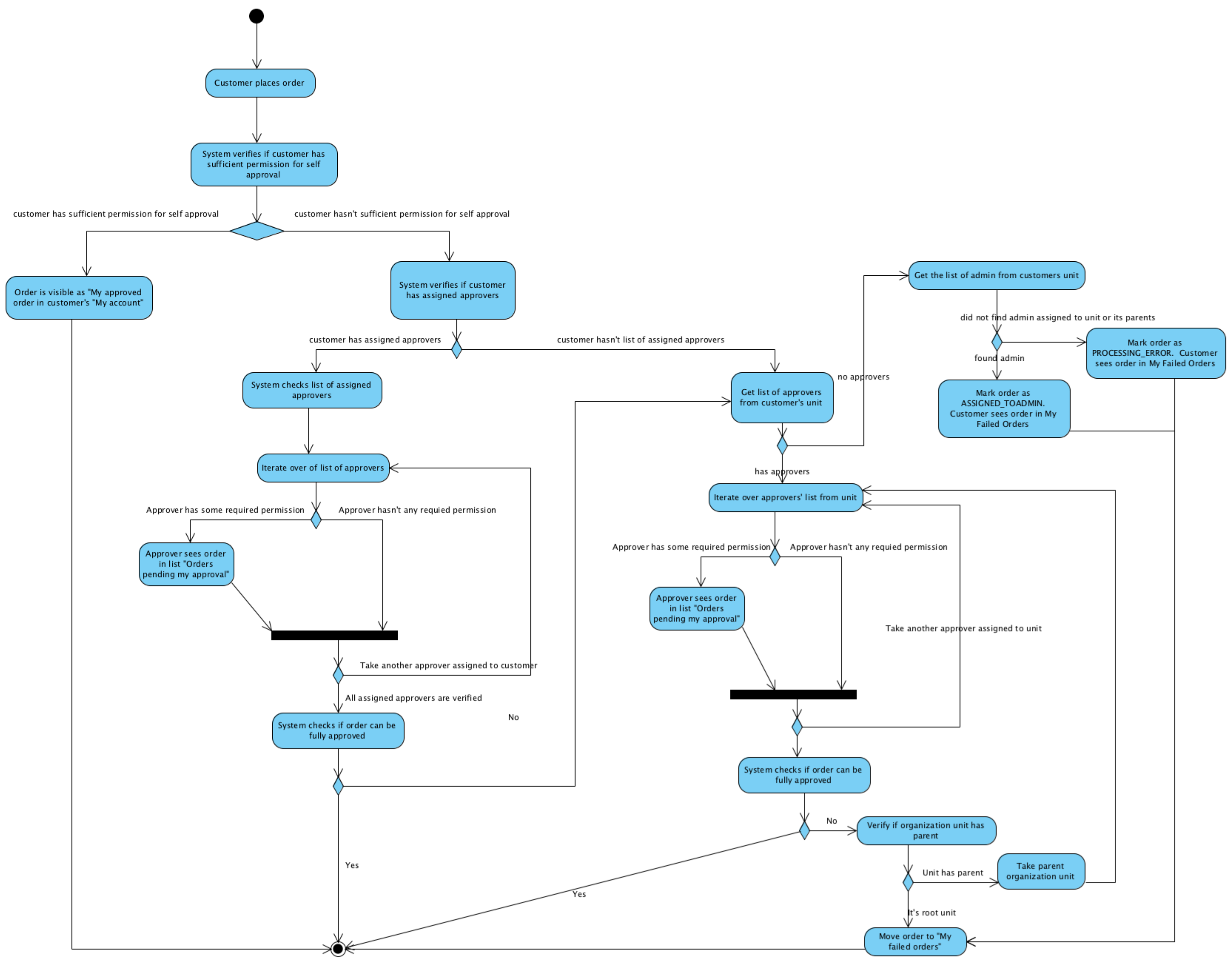
- B2B2POEPermissionA
- B2B2POEPermissionB

On the other hand, a user may not fulfill all the necessary requirements and their order goes into an approval work flow. This diagram illustrates a default **B2BOrderApproval** process.



Finding an Approver

The diagram below presents in details the algorithm used for finding an approver. The algorithm is a crucial part of the **findApprovers** element.


Customer places order
System verifies if customer has sufficient permission for self approval
customer has sufficient permission for self approval
customer hasn't sufficient permission for self approval
Order is visible as "My approved order in customer's "My account"
System verifies if customer has assigned approvers
customer has assigned approvers
customer hasn't list of assigned approvers
System checks list of assigned approvers
Iterate over of list of approvers
Approver has some required permission
Approver hasn't any requied permission
Approver sees order in list "Orders pending my approval"
Take another approver assigned to customer
All assigned approvers are verified
System checks if order can be fully approved
No
Yes
Get list of approvers from customer's unit
no approvers
has approvers
Get the list of admin from customers unit
did not find admin assigned to unit or its parents
found admin
Mark order as ASSIGNED_TOADMIN. Customer sees order in My Failed Orders
Mark order as PROCESSING_ERROR. Customer sees order in My Failed Orders
Iterate over approvers' list from unit
Approver has some required permission
Approver hasn't any requied permission
Approver sees order in list "Orders pending my approval"
Take another approver assigned to unit
System checks if order can be fully approved
No
Yes
Verify if organization unit has parent
Unit has parent
Take parent organization unit
It's root unit
Move order to "My failed orders"

Defining a Process

The first step in the process creation is to define it in the `b2borderapproval.xml` file, which you can find in the *<HYBRIS_BIN_DIR>*`/ext-commerce/b2bapprovalprocess/resources/process` folder. The `b2bapprovalprocess` extension contains an example of **B2BApprovalProcess**. Note how all the nodes in the process diagrams above are represented in the process definition and indicate the next process steps subject to their results.

You can find an example of the **B2BOrderApproval** process used in the Simple Order Approval process in the `b2borderapproval.xml` file.

You can find an example of the **B2BOrderApproval** process used in the Two Pair of Eyes Approval process in the `b2bTwoPairOfEyesApproval.xml` file, also located in *<HYBRIS_BIN_DIR>*`/modules/b2b-commerce/b2bapprovalprocess/resources/process`.

For detailed information on how a process should be defined, see The SAP Commerce processengine .

Defining Actions

The next step defines the actions that are specified in the `bean` attribute of the action nodes. Design of an action depends on the results an action should provide. The `processengine` functionality offers three standard abstract actions. For a description of them, see The SAP Commerce processengine.

When you have to implement action behavior, you can simply create a class that implements the action interface. Find more information on the action interface, see The SAP Commerce processengine. If you inherit from one of the actions specified below, you have a set of useful functionality at hand. In our example all three types of actions are included.

Action Name	Description	Code Snippet

`AbstractProceduralB2BOrderAproveAction` (example: startWorkflow)	For some actions there exists only one possible result that triggers the next event in the process chain. For these actions, the best solution is to define it to inherit from `AbstractProceduralB2BOrderAproveAction`. All you need to do is to implement the provided code snippet.	`@Override`
		`public voic`
		`{`
		`//some logi`
		`}`

`AbstractSimpleB2BApproveOrderDecisionAction` (example: CheckForApproval)	The most common type is the action that has two possible results. Such an action should inherit from `AbstractSimpleB2BApproveOrderDecisionAction`.	`public Trar`
		`{`
		`//some logi`
		`return Trar`
		`// or retur`
		`}`

`AbstractAction`	In this action we have four possible results. Since there is no defined action template for such a case, the best practice is to define an enumeration inside the class that defines all possible codes.	`public enum`
		`{`
		`OK, PARTIAL`
		
		`public stat`
		`{`
		`final Set<S`
		
		`for (final`
		`{`
		`res.add(t.t`
		`}`
		`return res;`
		`}`
		`}`

	The method `getStringValues()` is defined in the Action interface. Its implementation is generic, so if you need to define your own transition you can simply copy the code snippet, redefining only the possible result codes for the action.	`@Override`
		`public Set<`
		`{`
		`return Trar`
		`}`

All you need to do in action class implementation is to fill in the execute method. Simply put your logic in this method and return one of the results specified in `Transition` enum.

Spring Integration

Registration of Process Definition

To make the process work by defining the spring beans with the names specified in the process definition, and join them with the classes defined in the step before.

The following snippet from `b2bapprovalprocess-spring.xml` is an example bean definition for registering the process with the `processing` extension by providing a classpath to the resource:

```
<bean id="approvalProcess"
                class="de.hybris.platform.processengine.definition.ProcessDefinitionResource">
                <property name="resource" value="classpath:/process/b2borderapproval.xml" />
        </bean>
```

Defining Process Actions

You can find an example of the spring bean definitions used in the **B2BOrderApproval** process in the **b2bapprovalprocess-spring.xml** file, located in <*HYBRIS_BIN_DIR*>`/modules/b2b-commerce/b2bapprovalprocess/resources`.

Overwriting the b2bapprovalprocess Extension Default Process

To overwrite the default process:

1. Create a Custom Process Definition File, such as `custom-process.xml`. The process configuration file can be placed in <*HYBRIS_BIN_DIR*>`/ext-hybris/b2bapprovalprocess/resources/process`. The following is an example.

```
<?xml version="1.0" encoding="utf-8"?>
<process xmlns="http://www.hybris.de/xsd/processdefinition"
        start="checkForApproval" name="customApproval"
        processClass="de.hybris.platform.b2b.process.approval.model.CustomB2BApprovalProcessMode
        <action id="checkForApproval" bean="customCheckForApproval">
                <transition name="NOK" to="failed" />
                <transition name="OK" to="success" />
        </action>

        <end id="error" state="ERROR">All went wrong.</end>
        <end id="failed" state="FAILED">Order process failed.</end>
        <end id="success" state="SUCCEEDED">Order process finished.</end>
</process>
```

2. Implement `AbstractSimpleB2BApproveOrderDecisionAction` in `CheckForApproval.java`, as follows:

```
/*
 * [y] hybris Platform
 *
 * Copyright (c) 2000-2011 hybris AG
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of hybris
 * ("Confidential Information"). You shall not disclose such Confidential
 * Information and shall use it only in accordance with the terms of the
 * license agreement you entered into with hybris.
 *
 *
 */
package de.hybris.platform.b2b.process.approval.actions;

import de.hybris.platform.b2b.enums.PermissionStatus;
import de.hybris.platform.b2b.model.B2BCustomerModel;
import de.hybris.platform.b2b.model.B2BPermissionResultModel;
import de.hybris.platform.b2b.process.approval.model.B2BApprovalProcessModel;
import de.hybris.platform.b2b.services.impl.DefaultB2BPermissionService;
import de.hybris.platform.core.enums.OrderStatus;
import de.hybris.platform.core.model.order.OrderModel;
import de.hybris.platform.task.RetryLaterException;

import java.util.Set;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Required;

/**
 * Checks if the order requires an approver.
 *
 * @author DZ
 */
public class CheckForApproval extends AbstractB2BApproveOrderDecisionAction
{

        private static final Logger LOG = Logger.getLogger(CheckForApproval.class);
        private DefaultB2BPermissionService b2bPermissionService;

        @Override
        public Transition executeAction(final B2BApprovalProcessModel process) throws RetryLater
        {
                OrderModel order = null;
                try
                {
                        order = getOrderForProcess(process);
                        final B2BCustomerModel orderUser = (B2BCustomerModel) order.getUser();
                        final Set<B2BPermissionResultModel> permissionResults = b2bPermissionSer

                        // set only the results that need approval.
                        order.setPermissionResults(permissionResults);
                        order.setStatus(OrderStatus.PENDING_APPROVAL);
                        Transition transition = Transition.OK;

                        // NOK transition points to findApprovers action.
                        for (final B2BPermissionResultModel b2bPermissionResultModel : order.get
                        {
                                if (PermissionStatus.OPEN.equals(b2bPermissionResultModel.getSta
                                {
                                        transition = Transition.NOK;
                                }
                                else if (PermissionStatus.ERROR.equals(b2bPermissionResultModel.
                                {
                                        // this can happen if budget was not found or any error
                                        throw new RuntimeException("Failed to evaluate permissio
                                }
                                else
                                {
                                        b2bPermissionResultModel.setStatus(PermissionStatus.APPR
                                }

                                if (LOG.isDebugEnabled())
                                {
                                        LOG.debug(String.format("PermissionResult %s|%s|%s ", b2
                                                        b2bPermissionResultModel.getStatus(), b2
                                }
                        }

                        if (transition.equals(Transition.NOK))
                        {
                                //Do nothing here as we will approve it in checkapproval process
                                this.modelService.save(order);
                        }

                        return transition;
                }
                catch (final Exception e)
                {
                        LOG.error(e.getMessage(), e);
                        this.handleError(order, e);

                        return Transition.ERROR;
                }
        }

        public void handleError(final OrderModel order, final Exception e)
        {

                if (order != null)
                {
                        this.setOrderStatus(order, OrderStatus.B2B_PROCESSING_ERROR);
                }
                LOG.error(e.getMessage(), e);
        }

        public DefaultB2BPermissionService getB2bPermissionService()
        {
                return b2bPermissionService;
        }

        @Required
        public void setB2bPermissionService(final DefaultB2BPermissionService b2bPermissionServi
        {
                this.b2bPermissionService = b2bPermissionService;
        }

}
```

3. Configure the action as a Spring bean and inject the dependencies in your `custom-process-spring.xml` file, as follows:

```
<bean id="customCheckForApproval"
                class="de.hybris.platform.b2b.process.approval.actions.CheckForApproval"
                 parent="abstractB2BApproveOrderDecisionAction">
                <property name="b2bPermissionService" ref="b2bPermissionService" />
        </bean>
```

4. Register the process definition using Spring configuration in your `custom-process-spring.xml` file, as follows:

```
<bean id="customProcess"
                class="de.hybris.platform.processengine.definition.ProcessDefinitionResource"
                >
                <property name="resource" value="classpath:/process/custom-process.xml" />
        </bean>
```

5. Configure `B2BPlaceOrderStrategy` with the process name `customApproval`that was defined above in your `custom-process-spring.xml` file, as follows:

```
<alias alias="b2bplaceOrderStrategy" name="defaultB2BPlaceOrderStrategy" />
        <bean id="defaultB2BPlaceOrderStrategy"
                class="de.hybris.platform.b2b.strategies.impl.DefaultB2BPlaceOrderStrategy"
                parent="abstractBusinessService" >
                <property name="placeOrderStrategy" ref="placeOrderStrategy" />
                <property name="businessProcessService" ref="businessProcessService" />
                <property name="processCodeGenerator" ref="processCodeGenerator" />
                <property name="approvalProcessName" value="customCheckForApproval" />
        </bean>
```

B2B Organization Services

New or migrated services provided with the `b2bapprovalprocess` extension implement paginated lookups using the `commerceservices` extension search infrastructure.

Service Name	Lookup Item	Description
`B2BCommercePermissionService`	`B2BPermission`	This service provides methods to retrieve an instance of `B2BPermissionModel` by its code, retrieve a paginated list of permissions and to add permissions to, or remove permissions from `B2BCustomers` and `B2BUserGroups`.
`B2BApproverService`	`B2BCustomer`	This service provides methods to manage and lookup approvers (of type `B2BCustomerModel`) for a business unit, such as adding or removing approvers to or from `B2BUnits` and `B2BCustomers`, or retrieving a paginated list of approvers for a unit.
`B2BApprovalProcessService`	N/A	This service provides just one method to look up available approval process codes for a given base store.

[B2B Credit Management](#)

Credit limit management in SAP Commerce defines the maximum amount of credit that a B2B organization or B2B unit can have. Credit limits are useful if merchants do not want to process orders above a specific amount within a defined time period. If an order exceeds the credit limit, the merchant is notified and has to approve the order before it is processed.

[B2B Permissions](#)

B2B permissions are evaluated during the order approval process to check if an order needs additional approval or to find an approver for the order, if required. Learn about the default B2B permission types, as well as how to create new ones

Related Information

[B2B Commerce Module](#)

B2B Credit Management

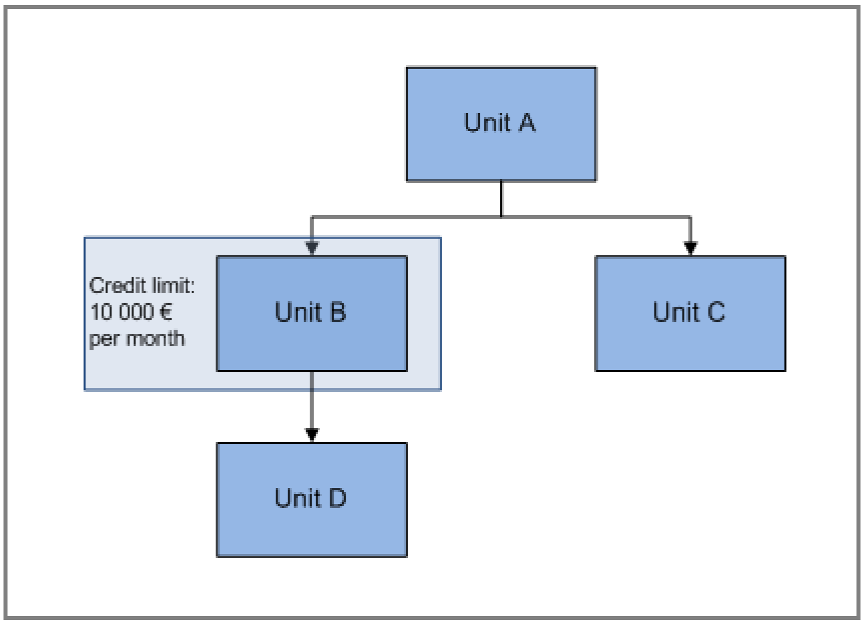
Credit limit management in SAP Commerce defines the maximum amount of credit that a B2B organization or B2B unit can have. Credit limits are useful if merchants do not want to process orders above a specific amount within a defined time period. If an order exceeds the credit limit, the merchant is notified and has to approve the order before it is processed.

B2B Credit Limit Overview

The credit limit check is a part of the B2B Order Approval Process, where an organization credit limit assigned by an account manager is checked. Note that B2B customers cannot view the credit limit assigned by the merchant.

The credit limit check is the last step of the order approval process. If the credit limit is not reached, the order is processed by default. If an order exceeds the credit limit alert threshold, the B2B merchant account manager receives a credit limit alert. If the order exceeds the credit limit amount, a workflow action item is created for the B2B merchant account manager. Credit limits can be specified for a day, week, month, quarter, year, or date range. If the credit limit parameters are changed, all new orders use the updated parameters to check against the credit limit.

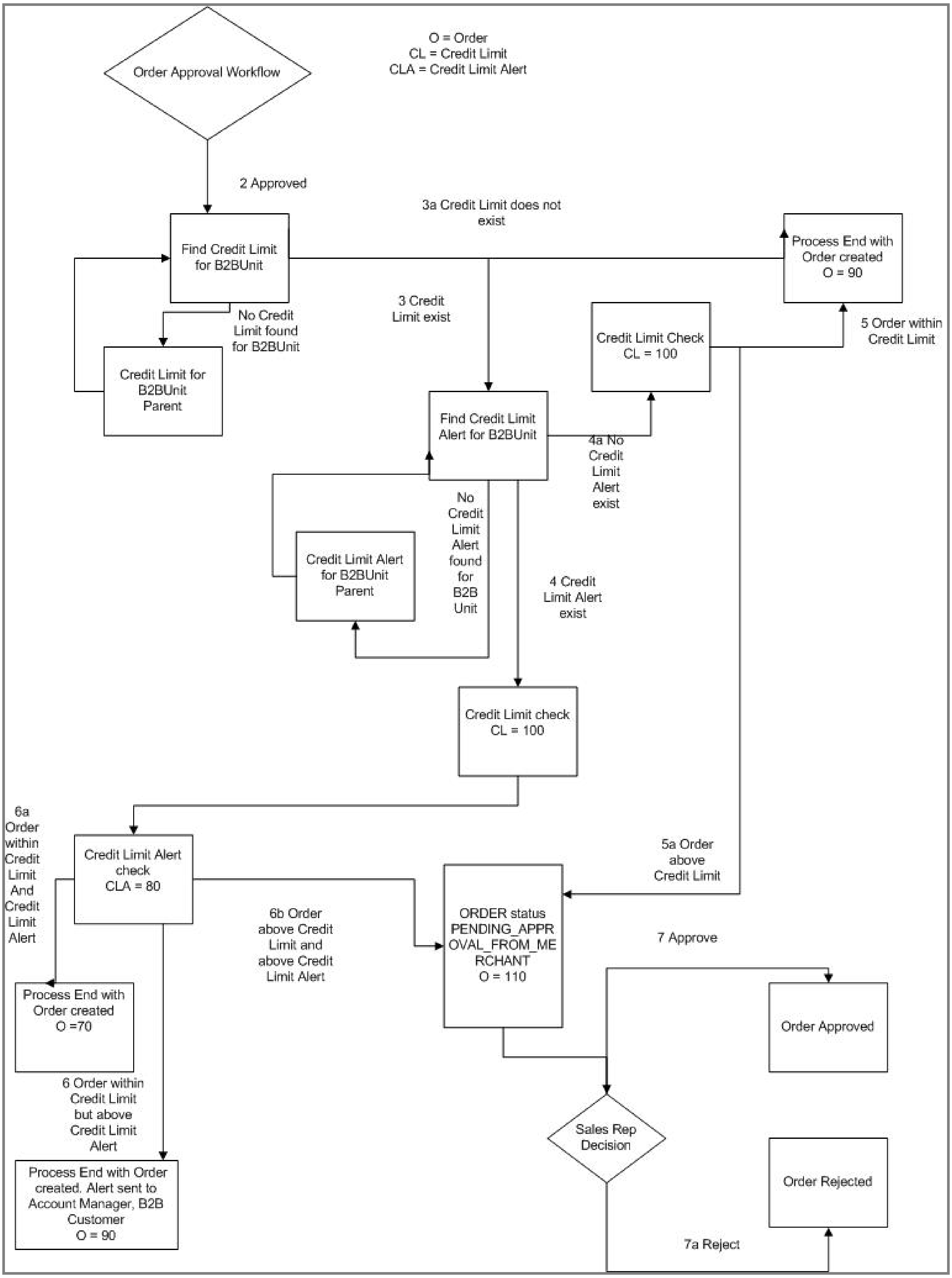
This diagram represents an example of the credit check logic:



Unit B has a limit of 10 000 EUR per month. Unit D already placed orders for 1000 USD in the current month. A customer is placing an order for 1000 EUR in Unit B.

The total order value in EUR for units B and D is 1700 EUR (1000 USD unit D order converted to EUR + 1000 EUR from the order placed in unit B). In this case, the order is processed because the 1700 EUR total is less than the 10 000 EUR credit limit assigned.

This diagram illustrates the B2B credit limit check flow:


Order Approval Workflow
O = Order
CL = Credit Limit
CLA = Credit Limit Alert
2 Approved
3a Credit Limit does not exist
Find Credit Limit for B2BUnit
Process End with Order created O = 90
No Credit Limit found for B2BUnit
3 Credit Limit exist
Credit Limit Check CL = 100
5 Order within Credit Limit
Credit Limit for B2BUnit Parent
Find Credit Limit Alert for B2BUnit
4a No Credit Limit Alert exist
No Credit Limit Alert found for B2B Unit
Credit Limit Alert for B2BUnit Parent
4 Credit Limit Alert exist
Credit Limit check CL = 100
6a Order within Credit Limit And Credit Limit Alert
Credit Limit Alert check CLA = 80
5a Order above Credit Limit
6b Order above Credit Limit and above Credit Limit Alert
ORDER status PENDING_APPROVAL_FROM_MERCHANT O = 110
7 Approve
Process End with Order created O =70
Order Approved
6 Order within Credit Limit but above Credit Limit Alert
Sales Rep Decision
Process End with Order created. Alert sent to Account Manager, B2B Customer O = 90
Order Rejected
7a Reject

For more information, see the following:

- Managing Credit Limits

- Working with Orders
- Managing Order Permissions in Backoffice
- Workflow Overview

Defining a Credit Limit Check

To define a credit limit check you need to be familiar with the process and the required steps to complete it. This also includes the actions to take in case an error occurs during the process.

The credit limit check is a part of the approval process, so the business process defined through the `b2borderapproval.xml` file is accordingly modified to accommodate it.

Credit Limit Check

The credit limit check is done for the unit where the order has been placed first, and if a credit limit is not found for that unit, the system moves up in the hierarchy to the point where it can find an assigned credit limit.

If a credit limit is found, the order total of the unit along with all units in same branch is calculated to check if the threshold value is reached.

The default evaluation strategy is located in the `DefaultB2BCreditLimitEvaluationStrategy.java` file.

To modify the default evaluation strategy, implement `DefaultB2BMerchantCheckService` and inject the strategy. There can be multiple strategies that you need for complete evaluation depending on different business scenarios. The following is an example of the `DefaultB2BMerchantCheckService.java` file.

```
package de.hybris.platform.b2b.services.impl;

import de.hybris.platform.b2b.model.B2BCustomerModel;
.........................................................
.........................................................

public class DefaultB2BMerchantCheckService extends AbstractBusinessService implements B2BMerchantChec
{

        private static final Logger LOG = Logger.getLogger(DefaultB2BMerchantCheckService.class);
        private Set<EvaluateStrategy<B2BMerchantCheckResultModel, AbstractOrderModel, B2BCustomerModel
        protected List<String> merchantCheckTypes;
        private UserService userService;

        @Override
        public Set<B2BMerchantCheckResultModel> evaluateMerchantChecks(final AbstractOrderModel order,
        {
                final Set<B2BMerchantCheckResultModel> merchantCheckResults = new HashSet<B2BMerchantC
                getSessionService().executeInLocalView(new SessionExecutionBody()
                {
                        @Override
                        public void executeWithoutResult()
                        {
                                getUserService().setCurrentUser(getUserService().getAdminUser());
                                for (final EvaluateStrategy<B2BMerchantCheckResultModel, AbstractOrder
                                {
                                        final B2BMerchantCheckResultModel evaluationResult = evaluateS
                                        merchantCheckResults.add(evaluationResult);

                                }
                        }
                });

                return merchantCheckResults;

        }
        @Required
        public void setEvaluateStrategies(
                        final Set<EvaluateStrategy<B2BMerchantCheckResultModel, AbstractOrderModel, B2
        {
                this.evaluateStrategies = evaluateStrategies;
        }

        .........................................................
        .........................................................
}
```

Defining a Process

In defining a process, The first step in defining a process is to modify the existing process definition `b2borderapproval.xml` file, located in the `<HYBRIS_BIN_DIR>/modules/b2b-commerce/b2bapprovalprocess/resources/process` directory. The `b2bapprovalprocess` extension contains an example of `B2BApprovalProcess`. Note how all the nodes in the process diagrams above are represented in the process definition and indicate the next process steps subject to their results.

For more information on how a process should be defined, see The SAP Commerce processengine.

To see an example of an `OrderApproval` process in which the credit limit or merchant check is accomodated, refer to the `b2borderapproval.xml` file.

Defining Actions

The next step is to define the actions that are specified in the `bean` attribute of the action nodes. Most of the actions are outlined in the b2bapprovalprocess Extension document. Here we will discuss the actions for credit limit only, which are `performMerchantCheck` and `checkCreditLimitResult`. Design of an action depends on results an action is to provide. The `processengine` functionality offers three standard abstract actions, you can find description of actions in The SAP Commerce processengine.

When you have to implement action behavior, you simply create a class that implements the action interface. Find more information on Action Interface in The SAP Commerce processengine. If you inherit from one of the actions specified below, you have a set of useful functionality at hand. In our example all three types of actions are included.

The sample implementation of the `PerformMerchantCheck` actions can be viewed in the `PerformMerchantCheck.java` file.

You also need to fill in the execute method. Simply put your logic in this method and return one of the results specified in the `Transition` enum. If the order needs an approver, or in this case an Account Manager's action because the order reached its credit limit, a workflow template is created for the approver to accept or reject the order.

The next step is to implement the `CheckCreditResults` of type `AbstractSimpleB2BApproveOrderDecisionAction`, which is executed when an approver makes a decision and accordingly sets the status to **APPROVED** or **REJECTED_BY_MERCHANT**.

The following is an example of the `CheckCreditLimitResults.java` file:

```
package de.hybris.platform.b2b.process.approval.actions;

import de.hybris.platform.b2b.process.approval.model.B2BApprovalProcessModel;
.........................................................
.........................................................
public class CheckCreditLimitResults extends AbstractSimpleB2BApproveOrderDecisionAction
{

        private static final Logger LOG = Logger.getLogger(CheckCreditLimitResults.class);

        @Override
        public Transition executeAction(final B2BApprovalProcessModel process) throws RetryLaterExcept
```

```
	{
		OrderModel order = null;
		try
		{
			order = process.getOrder();

			if (OrderStatus.REJECTED.equals(order.getStatus()))
			{
				order.setStatus(OrderStatus.REJECTED_BY_MERCHANT);
				this.modelService.save(order);
				return Transition.NOK;
			}

			order.setStatus(OrderStatus.APPROVED);
			this.modelService.save(order);
			return Transition.OK;

		}
		catch (final Exception e)
		{
			LOG.error(e.getMessage(), e);
			this.handleError(order, e);
			return Transition.NOK;
		}
	}

	private void handleError(final OrderModel order, final Exception e)
	{

		if (order != null)
		{
			this.setOrderStatus(order, OrderStatus.B2B_PROCESSING_ERROR);
		}
		LOG.error(e.getMessage(), e);
	}

}
```

Spring Integration

Registration of Process Definition

The registration process is exactly the same as for the rest of the approval process.

For more information, see b2bapprovalprocess Extension.

Defining Process Actions

The following is an example Spring bean definition used in the `B2BOrderApproval` process, as seen in the `b2bapprovalprocess-spring.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.or
	xsi:schemaLocation="http://www.springframework.org/schema/beans
	   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
	   http://www.springframework.org/schema/aop
	   http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">
....................................................
....................................................
....................................................
	<bean id="performMerchantCheck"
		class="de.hybris.platform.b2b.process.approval.actions.PerformMerchantCheck"
		parent="abstractSimpleB2BApproveOrderDecisionAction">
		<property name="b2bMerchantCheckService" ref="b2bMerchantCheckService" />
		<property name="userService" ref="userService" />
		<property name="b2bUnitService" ref="b2bUnitService" />
		<property name="b2bApproverService" ref="b2bApproverService" />
```

```
        </bean>

        <bean id="checkCreditLimitResults"
                class="de.hybris.platform.b2b.process.approval.actions.CheckCreditLimitResults"
                parent="abstractSimpleB2BApproveOrderDecisionAction">
        </bean>

.....................................................
.....................................................
.....................................................
</beans>
```

Related Information

Managing Credit Limits
b2bcommerce Extension
Business Process Management
basecommerce Extension
Workflow Overview
workflow Extension
The SAP Commerce processengine
The Task Service
yacceleratorfulfilmentprocess Extension

B2B Permissions

B2B permissions are evaluated during the order approval process to check if an order needs additional approval or to find an approver for the order, if required. Learn about the default B2B permission types, as well as how to create new ones

The following are the default permission types that come with the `b2bapprovalprocess` extension and are meant as a guideline for you to implement organization-specific permission logic:

- **B2BOrderThresholdPermission**
- **B2BOrderThresholdTimespanPermission**
- **B2BBudgetExceededPermission**
- **B2B2POEPermissionA**
- **B2B2POEPermissionB**

For more information on how to create and manage permissions and permission groups, see Managing Order Permissions in Backoffice.

Permission Types and Spring Definition

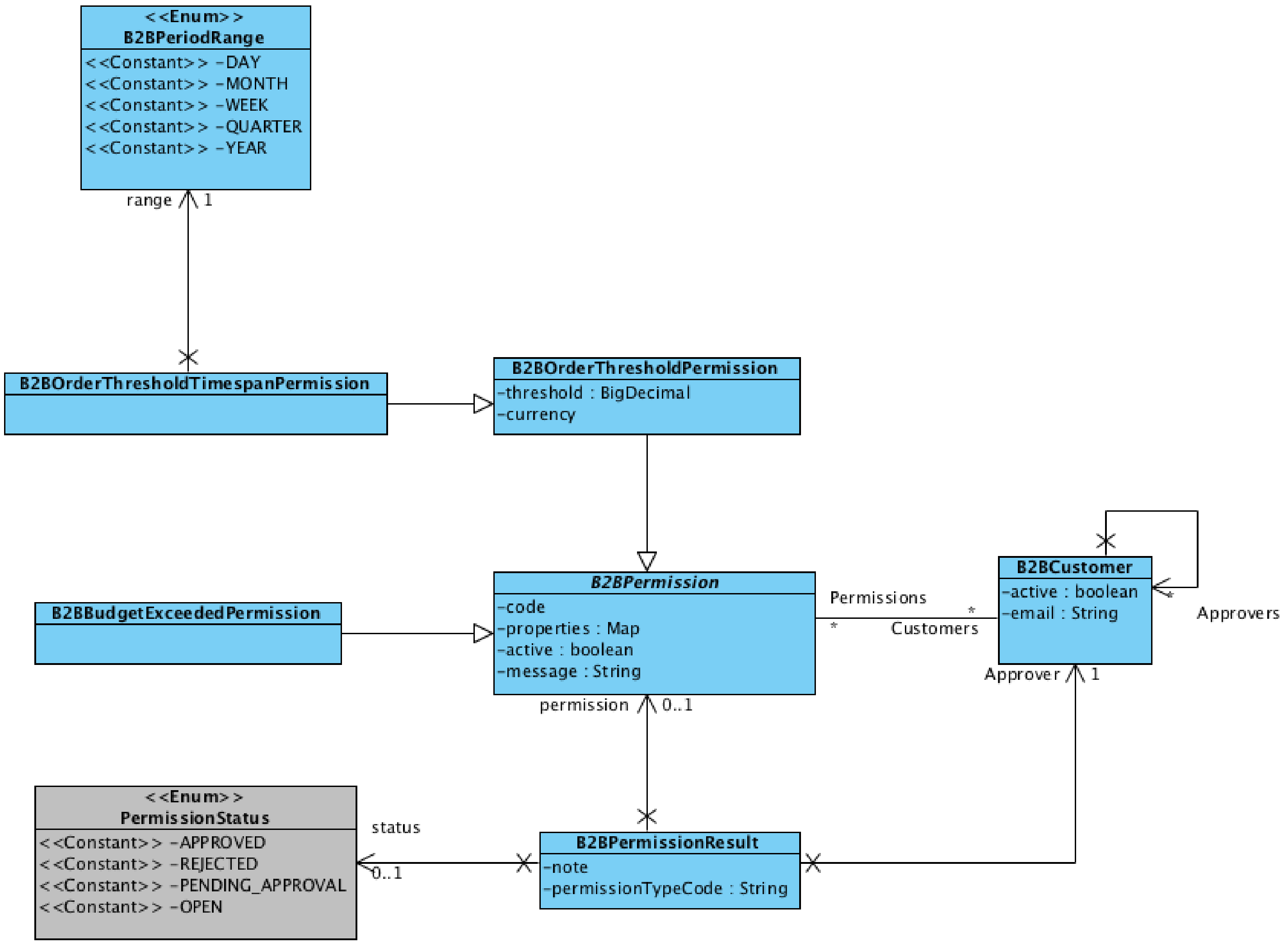
This section outlines the item types created for permissions by the `b2bapprovalprocess` extension. It includes excerpts from `b2bapprovalprocess-items.xml` as well as spring configuration for strategies that know how to evaluate different permissions types.

Definition of B2B Permission Types

You define **b2bpermission** types in the `b2bapprovalprocess-items.xml` file.

Class Diagram

The following class diagram shows the inheritance hierarchy of the permissions, and also shows how the **B2BPermissionResult** relates to permissions and a B2B customer.



Spring Definition

Permissions are checked by **B2BPermissionService**. The logic for how each permission is evaluated is encapsulated in a strategy. The following strategies are provided by default:

- **b2bBudgetExceededEvaluationStrategy**
- **b2bOrderThresholdEvaluationStrategy**
- **b2bOrderThresholdTimespanEvaluationStrategy**

You can see how these strategies are injected into the service by viewing the `b2bapprovalprocess-spring.xml` file. These strategies are provided as a guideline. For your specific project, you would most likely need to write your own logic. The following sections describe how to do just that.

i Note

For orders to be placed successfully, the three permissions associated with these evaluation strategies (**B2BBudgetExceededPermission**, **B2BOrderThresholdPermission**, and **B2BOrderThresholdTimespanPermission**) need to be created in the Backoffice Administration Cockpit and assigned to at least one customer and one approver. If you do not wish to use a specific permission, you need to remove the associated evaluation strategy from the `evaluateStrategies` property of the **defaultB2BPermissionService** bean that is defined in `b2bapprovalprocess-spring.xml`.

Note also that permissions created in the Backoffice Administration Cockpit need to satisfy the associated evaluation strategy in order for orders to be placed successfully.

Adding a New Permission

The permission types provided do not cover all scenarios. It is worth noting that the **B2BPermission** type definition is flexible as far as storing data you may need. The `B2BPermission.properties` file uses a **KeyValueCollection** type, which means this attribute can store any key-value pair. You can use this attribute if you want to be generic, or extend from **B2BPermission** as shown in the section below.

Adding Created Permission Type to the System

In this example, the volume of orders a customer can place is limited. To do it, create a permission type that holds the state needed to achieve checking for number of orders placed by customer. In the following `your_extension_items.xml` file, the name of the example permission is **B2BOrderValueExceededPermission**.

```
<itemtype code="B2BOrderValueExceededPermission"
                        jaloclass="de.hybris.platform.b2b.jalo.B2BOrderValueExceededPermission"
                        autocreate="true" generate="true" extends="B2BPermission">
                        <attributes>
                                <attribute qualifier="orderVolume" type="java.lang.Long">
                                        <modifiers read="true" write="true" search="true"
                                                optional="false" />
                                        <persistence type="property" />
                                </attribute>
                                <attribute qualifier="currency" type="Currency">
                                        <modifiers read="true" write="true" initial="false"
                                                optional="false" search="true" />
                                        <persistence type="property" />
                                </attribute>
                        </attributes>
                </itemtype>
```

Creating an Evaluation Strategy

The strategy that you create extends `AbstractEvaluationStragegy` and implements an `EvaluateStrategy` interface.

You place your logic for permission evaluation in the evaluate method which should return a **B2BPermissionResultModel** with the correct status **OPEN**. It means the permission was not satisfied and **PENDING_APPROVAL** means the permission was fulfilled, in this example this would mean the order created did not reach maximum value.

```
public class DefaultOrderValueExccededEvaluationStrategy extends AbstractEvaluationStragegy implements
                EvaluateStrategy<B2BPermissionResultModel, AbstractOrderModel, B2BCustomerModel>
{
        @Override
        public B2BPermissionResultModel evaluate(final AbstractOrderModel order, final B2BCustomerMode
        {
                PermissionStatus status = PermissionStatus.OPEN;
                /*
                * add logic here to derive the status for the result

                */

                final B2BPermissionResultModel result = this.getModelService().create(B2BPermissionRes
                result.setApprover(employee);
                result.setPermission(permissionToEvaluate);
                result.setPermissionTypeCode(B2BCommerceConstants.TC.B2BORDERVALUEEXCEEDEDPERMISSION);
                result.setStatus(status);

                return result;
        }

        /**
         * Gets the permission to evaluate.
         *
         * @param permissions
         *           the permissions
         * @param type
         *           the type
         * @return the permission to evaluate
         */
        public B2BOrderValueExceededPermission getPermissionToEvaluate(final List<B2BPermissionModel>
                        final Class<? extends B2BPermissionModel> type)
        {
                return (B2BOrderValueExceededPermission) CollectionUtils.find(permissions, PredicateUt
        }

        /*
         * (non-Javadoc)
         *
         * @see de.hybris.platform.b2b.strategies.EvaluateStrategy#getPermissionType()
         */
        @Override
        public Class<? extends B2BPermissionModel> getPermissionType()
        {
                return B2BOrderValueExceededPermission.class;
        }
```

Spring Integration

In order for the new permission strategy to get integrated into the **PermissionService** you need to inject the **orderValueExccededEvaluationStrategy** into **de.hybris.platform.b2b.services.impl.DefaultB2BPermissionService** you should give a permission service definition a unique ID name, for example **XYZB2BPermissionService** and alise it with alias **b2bPermissionService** so that it overwrites the default definition.

```
 <alias alias="b2bPermissionService" name="XYZB2BPermissionService" />
        <alias alias="orderValueExccededEvaluationStrategy" name="defaultOrderValueExccededEvaluationS

        <bean id="XYZB2BPermissionService"
                class="de.hybris.platform.b2b.services.impl.DefaultB2BPermissionService"
                parent="abstractBusinessService">
                <property name="evaluateStrategies">
                        <list>
                                <ref bean="b2bOrderThresholdEvaluationStrategy" />
                                <ref bean="b2bOrderThresholdTimespanEvaluationStrategy" />
                                <ref bean="b2bBudgetExceededEvaluationStrategy" />
                                <!-- inject new strategy here. -->
                                <ref bean="orderValueExccededEvaluationStrategy" />
                        </list>
                </property>
                <property name="b2bApproverService" ref="b2bApproverService" />
                <property name="permissionResultHelper" ref="permissionResultHelper" />
                <property name="userService" ref="userService" />
        </bean>

        <bean id="defaultOrderValueExccededEvaluationStrategy"
                class="de.hybris.platform.b2b.strategies.impl.DefaultOrderValueExccededEvaluationStrat
                parent="abstractEvaluationStragegy">
        </bean>
```

Related Information

Business Process Management

yacceleratorfulfilmentprocess Extension

basecommerce Extension

The SAP Commerce processengine

The Task Service

Managing Order Permissions in Backoffice

b2bapprovalprocessfacades Extension

The `b2bapprovalprocessfacades` provides a number of facades that are backed by the services in the `b2bapprovalprocess` extension.

This facade has been migrated from `b2bacceleratorfacades` and refactored into smaller facades, where applicable, in order to have a cleaner separation of concern.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

B2B Facades

The following facades expose B2B services from the `b2bapprovalprocess` extension:

- `B2BApprovalProcessFacade`
- `B2BApproverFacade`
- `B2BPermissionFacade`

b2bcommerce Extension

The `b2bcommerce` extension enables you to provide B2B functionality to your organization.

For information on how to create an order approval process, see b2bapprovalprocess Extension.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

Data Access Security

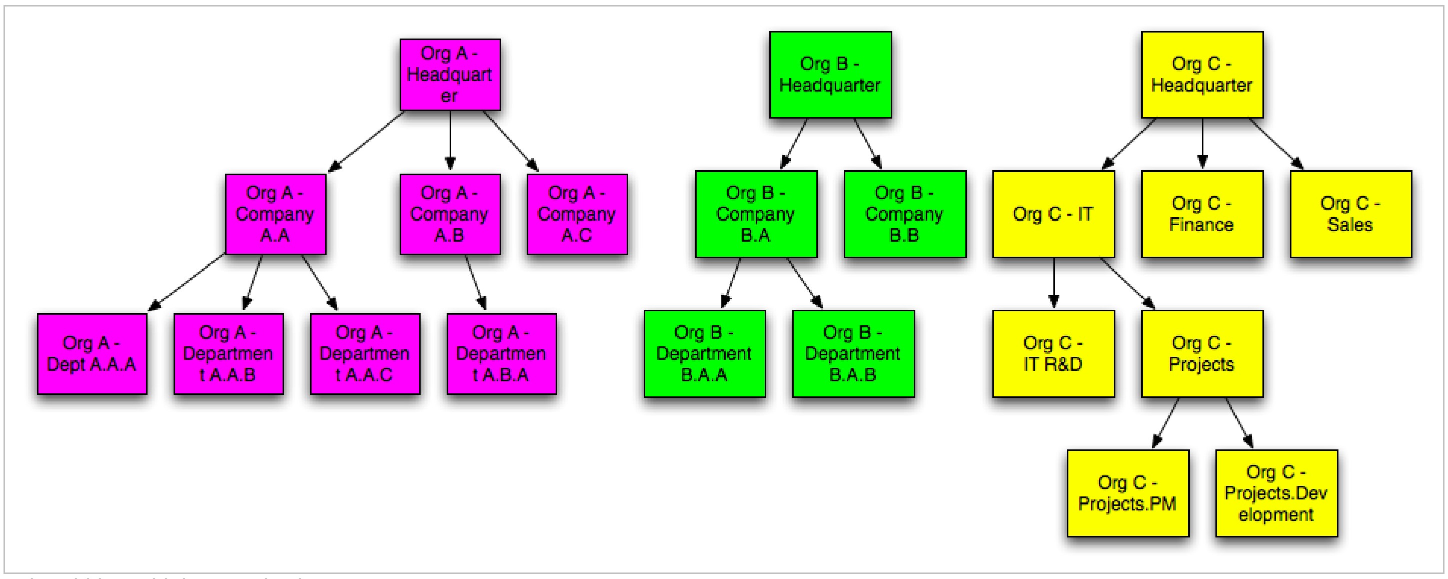
The data security in B2B scenarios requires enhanced mechanisms to support multiple users from within one B2B organization as well as preventing access to data from one B2B organization to another.

Visibility

i Note

You can assign customers to more than one unit. User visibility restrictions apply to all of their assigned organizations.

The following image is a visualization of potential organization configuration in a B2B scenario. All the organizations (A-C) are configured and setup in the B2B eCommerce system to view prices, place orders, setup new employees, and organizational units.



Units Within Multiple Organizations

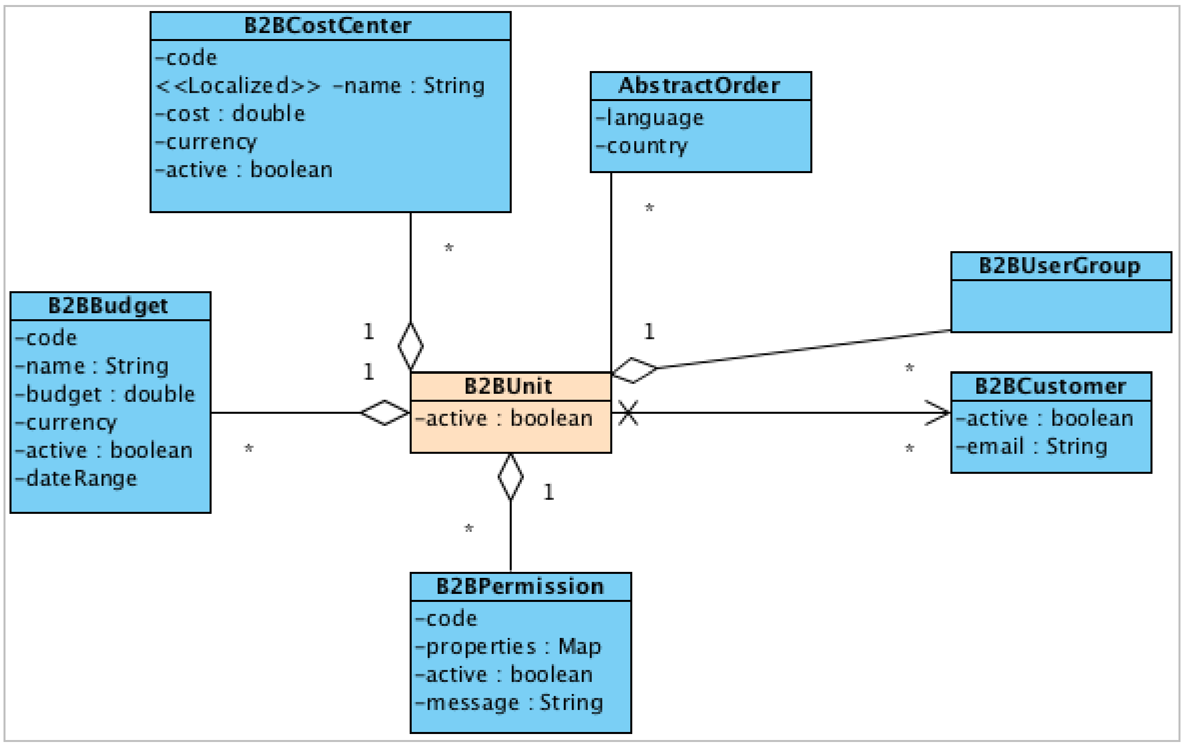
B2B customer of the B2B Organization A (Org A) cannot see information (prices, orders, cost center information) from any other B2B organizations or B2B units (Org B and Org C), place orders on behalf of other B2B organizations or create B2B units in other B2B organization branches.

The restriction in visibility is bound to the **B2BGroup** item. Every B2B customer has to be a member of this group.

i Note

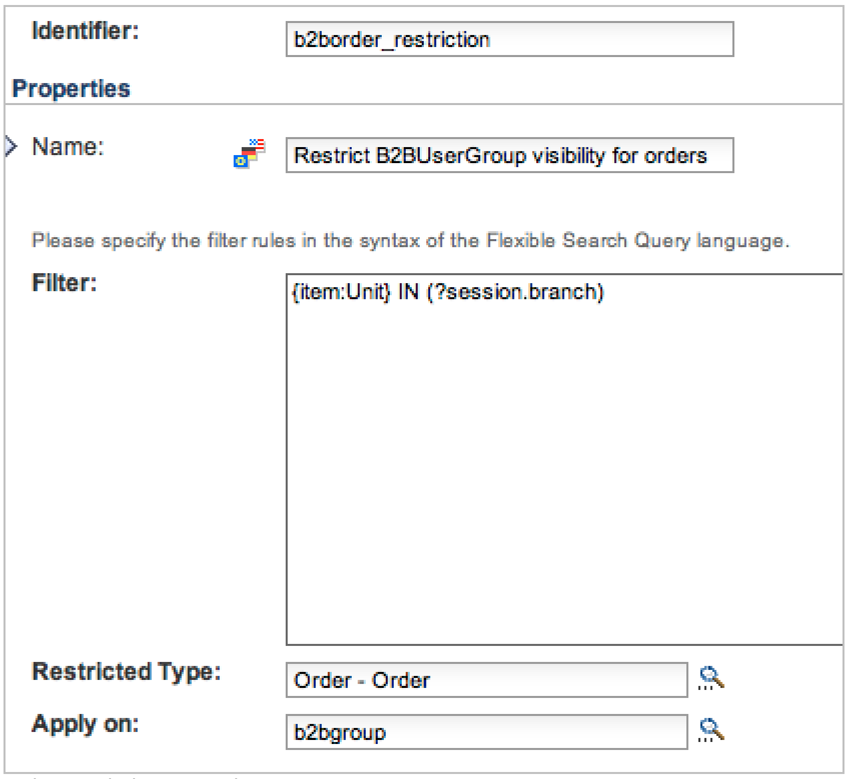
The reason to have only one group with the restrictions configured is maintainability. With every custom restriction (for example, B2B customers can only see their orders) that is introduced, the restriction has to be applied to every item. Maintaining the simple rule is easier and still very flexible and allows for most use cases regarding organization.

If you want to restrict visibility, use standard SAP Commerce personalization.



Structure Behind B2B Unit Personalization

All objects that should apply to personalization rules are related to the B2B Unit and a personalization rule is applied.



Identifier: b2border_restriction

Properties

Name: Restrict B2BUserGroup visibility for orders

Please specify the filter rules in the syntax of the Flexible Search Query language.

Filter: {item:Unit} IN (?session.branch)

Restricted Type: Order - Order

Apply on: b2bgroup

Order Restriction Example

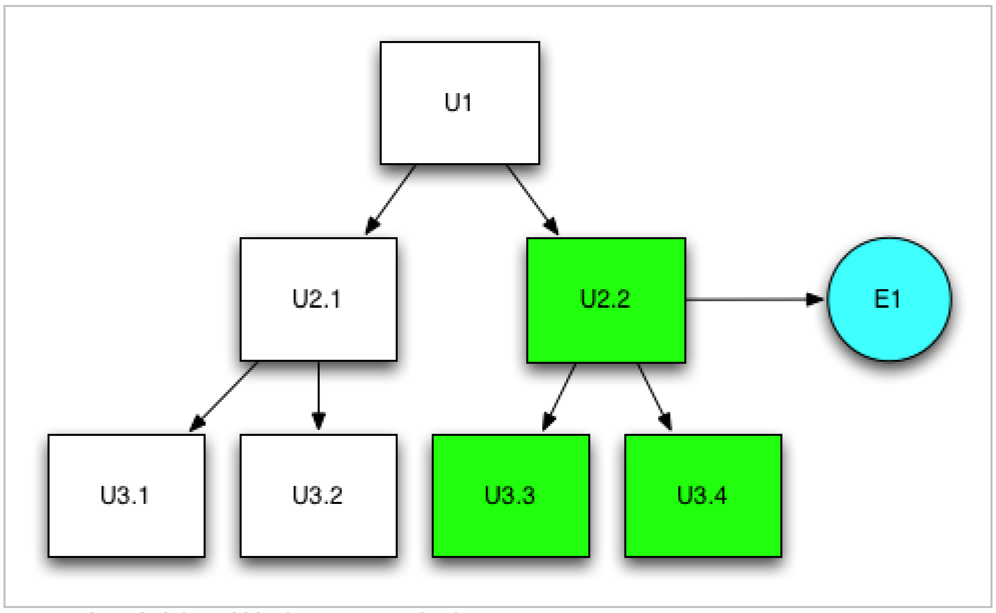
Setting the Branch

The personalization reads branch information from session **session.branch**. Calculating the branch on every request would come with some cost. The information about the branch is set when the user session in **AfterSessionCreationListener** or **AfterSessionUserChangeListener**.

```
protected void onEvent(final AfterSessionUserChangeEvent event)
        {
                final JaloSession jaloSession = (JaloSession) event.getSource();
                final UserModel currentUser = userService.getCurrentUser();
                b2bUnitService.updateBranchInContext(jaloSession, currentUser);
                resetCurrency(currentUser);
        }
```

Propagation of Rights

Access rights within a B2B Organization are propagated down in the hierarchy.



Propagation of Rights Within the B2B Organization

In the previous example, Employee (E1) has the right to view orders for the B2B Unit (U2.2). This right gets propagated down the tree and the employee has the right to view orders for B2B Units (U3.3) and (U3.4) as well.

B2B customers always have the right to view all the information they are granted access to for their B2B branch (the assigned B2B unit and the descendants).

Related Information

B2B Commerce Module
b2bapprovalprocess Extension
basecommerce Extension

B2B Organization Services

New or migrated services provided with the `b2bcommerce` extension implement paginated lookups using the `commerceservices` extension search infrastructure.

Service Name	Lookup Item	Description
`B2BBudgetService`	`B2BBudget`	This service provides methods to look up instances of `B2BBudgetModel` by UID and getting a paginated list of budgets.
`B2BCommerceB2BUserGroupService`	`B2BUserGroup`	This service provides methods to manage and look up instances of `B2BUserGroupModel`, like assigning members to a usergroup, getting a paginated list of usergroups, and more.
`B2BCommerceCostCenterService`	`B2BCostCenter`	This service provides methods to manage and look up instances of `B2BCostCenterModel` by UID and getting a paginated list of cost centers.
`B2BCommerceUnitService`	`B2BUnit`	This service provides some common manipulation methods for instances of `B2BUnitModel`, such as disabling units and editing unit delivery addresses.

Service Name	Lookup Item	Description
`B2BCommerceUserService`	`B2BCustomer`	This service provides some common methods for `B2BCustomerModel` instances. In addition to searching for customers by UID and performing paginated searches, this service provides methods for disabling customers and adding user roles.

B2B Types

B2B types represent fundamental entities in the B2B transaction domain.

Every B2B item type should have limited visibility across a B2B organization. It has to have a relation to the B2B unit, as the visibility check is based on the branch the B2B unit is parent of.

Note the following if the relation to the B2B unit is altered (for example deleted, moved, disabled, or orphaned) with respect to master and transaction data.

- Delete: Master data cannot be deleted (cost center, budget, unit, employee). If transactional data is deleted it is removed from the reports as well.
- Move (when possible): Transactional and master data assigned to a master data item (like a B2B unit) is moved with it.
- Disabled: All data is still there, and you can create reports that include or exclude data from disabled units.
- Orphaned: All data is still there and can be part of a report. When it loses the connection to a B2B unit it's only accessible by a merchant admin.

Common B2B Item Fields

B2B items are found in the B2B Commerce section in the Backoffice Administration Cockpit. For example, to view the fields for a B2B unit, navigate to B2B Commerce B2B Unit , and then select a B2B unit from the list that appears. The details for the units are displayed in the bottom portion of the page.

All B2B items have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B item. The required fields may not be null, but may be modified.

Name	Required	Type	Location in Backoffice
Name	Yes	Text	General Essential
ID (unique)	Yes	Text	General Essential
Description	Optional	Text	General Properties

Note that the Active status (in the Administration tab) is automatically set to active. Disabling it is permanent on B2B units and has a cascading effect on all child B2B units.

B2B Unit

A B2B unit is an organizational unit representing a company node, department node, or branch node. It holds relations to almost every B2B type because the visibility restriction is based on a B2B Unit. Units can have one parent B2B unit assigned and they can have multiple children assigned. The B2B unit that has no parent B2B unit is the B2B root unit of a B2B organization.

A B2B unit and its descendants form a branch of the B2B organization. The branch of the B2B root unit is the B2B organization. Orders are assigned to B2B units, so if you move an employee from one B2B unit to another, the order is still assigned to the same B2B unit. B2B units can have an approver group assigned, which is a list of potential approvers of an order with a set of predefined permissions. A B2B unit is a subtype of the SAP Commerce type company, for more information on SAP Commerce types, see Users in the Platform.

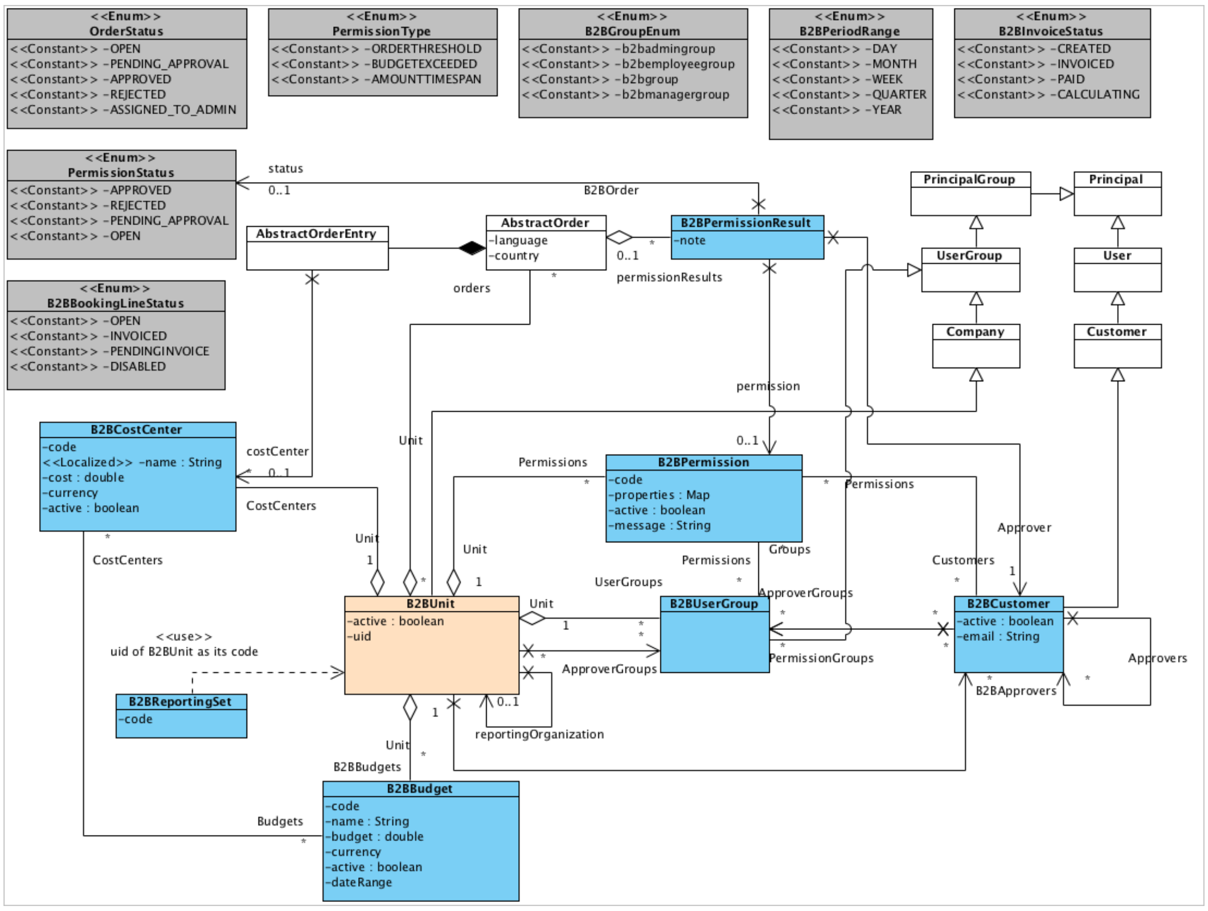
B2B units are accessed by navigating to **B2B Commerce B2B Unit** and then selecting a unit. B2B units have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B unit. The required fields may not be null, but may be modified:

Field	Required	Type	Location in Backoffice	Comments
Parent B2BUnit	Yes	B2BUnit	Organization Organizational Units	Required for all but the root B2B unit
VAT ID	No	Text	General IDs	n/a
DUNS ID	No	Text	General IDs	n/a
Contact Person	No	B2BCustomer	General Properties	n/a
Cost Centers	No	B2BCostCenter(s)	Cost Center	n/a
Approver Groups	No	B2BPermissionGroup(s)	Approvers Approvers	n/a
Approvers	No	B2BCustomer(s)	Approvers Approvers	n/a

For more information, see Users in the Platform.

B2B Core Types

This diagram presents all core types for B2B Commerce:



B2B Order

An order is placed for a B2B organization. The B2B order is assigned to a B2B unit, still holding its owner as the person who placed the order.

An order can only use one currency. B2B customers have no limit on the currency to use, but they are limited by the currency of the cost centers they can access, and by the permissions they are assigned. A permission of order threshold has a currency and the currency gets checked when the order gets placed.

For a B2B order the `OrderLineItem` could have references to cost centers.

B2B orders are accessed by navigating to Order Orders and then selecting an order. B2B orders have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B order. The required fields may not be null, but may be modified:

Field	Required	Type	Location in Backoffice
User	Yes	B2BCustomer	Positions and Prices Essential
Order Status	Yes	Enumeration	Payment and Delivery Status
B2B Comment	No	Text	Positions and Prices B2B Comment

B2B Customer

B2B customers are assigned to a B2B unit. They are members of at least one B2B group. Their visibility and actions are limited to the B2B branch they are assigned to.

B2B customers can be active or inactive. When inactive, they are not able to log in, and therefore cannot approve orders, place orders, or perform any action on the system. Only an admin can reactivate a user. When they have access to items, this right is propagated down the branch, so they can see all items for their B2B unit and descendants.

B2B customers have a set of permissions assigned. A B2B customer with permissions must also be assigned to the appropriate user group that grants the corresponding CRUD rights for the item the customer wants to modify.

You can move a B2B customer from one B2B unit to another within the B2B organization by updating the parent B2B unit. New budgets and cost centers are applied accordingly. However, B2B permission groups are transferred and may need to be adjusted. Order history is maintained both on B2B customer and B2B unit at the time order was placed, so historical data is maintained.

B2B customers are accessed by navigating to B2B Commerce B2B Customer and then selecting a customer. B2B customers have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B customer. The required fields may not be null, but may be modified:

Field	Required	Type	Location in Backoffice
Groups (defines the parent B2B unit)	Yes	B2BUnit	General Properties
Email	Yes	Text	Addresses Addresses
Approvers	No	B2B Customer(s)	Approvers Approvers
Approver Groups	No	B2B Permission Group(s)	Approvers Approvers
Groups	No	B2B Group(s)	General Properties

B2B Cost Center

B2B cost centers are assigned to a B2B budget and have `orderLineItems` assigned to them. The calculation of cost centers total is done by summing up all the `orderLineItems`. When compared to a budget the date range of the budget is taken into consideration. If the cost center's budget is disabled, all subsequent orders billed to the cost center trigger the `BudgetExceededPermission` approval process.

B2B cost centers do not sum up over branches or hierarchy levels. Each B2B cost center has its own budget and order line items assigned. A B2B customer can see all cost centers for his B2B branch

B2B cost centers are accessed by navigating to B2B Commerce B2B Cost Center and then selecting a cost center. B2B cost centers have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B cost center. The required fields may not be null, but may be modified:

Field	Required	Type	Location in Backoffice
Parent B2BUnit	Yes	B2BUnit	General Essential and Hierarchy Essential
Budget	No	B2BBudget	Hierarchy Parent B2B Unit
Active (default: **true**)	Yes	Boolean	General Properties

B2B Budget

A B2B budget is assigned to one or many cost centers. Purchases are billed to budgets using the cost centers.

If a budget is disabled, all subsequent orders trigger the `BudgetExceeded` order approval workflow. All orders approved prior to a budget being modified are not affected. If orders are in pending states, once they become approved by their approver, they are re-tested against the budget before moving on to approved state.

B2B budgets are accessed by navigating to B2B Commerce B2B Budget and then selecting a budget. B2B budgets have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B budget The required fields may not be null, but may be modified:

Field	Required	Type	Location in Backoffice
Parent B2BUnit	Yes	B2BUnit	General Essential and Hierarchy Essential
Currency	Yes	Currency	General Properties
Budget	Yes	BigDecimal	General Properties
Date Range Start / End	Yes	Date	General Properties

B2B Escalation Cron Job

A B2B escalation cron job checks orders in **pending approval** state and escalates them to next approver level if the idle period is exceeded.

B2B Permissions

B2B permission is a type used to verify whether an order placed by a B2B customer can proceed without further approval. There are three permission provided out of the box; you need to implement and configure additional permissions. If a B2B customer does not have the necessary permissions to place an order, order approvers are notified and an approver must manually approve the order. Order approvals themselves must have the necessary permissions to approve the specific order.

B2BApproveOrderPermission / B2BAmountTimeSpanPermission

`B2BApproveOrderPermission`: Possessors of this permission are allowed to approve orders that are below the specified threshold.

`B2BAmountTimeSpanPermission`: Possessors of this permission are allowed to spend a specific amount over a specified number of days.

B2B permissions are accessed by navigating to B2B Approval Process B2B Permission and then selecting a permission. B2B permissions have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B permission. The required fields may not be null, but may be modified:

Field	Required	Type	Location in Backoffice
Parent B2B Unit	Yes	B2BUnit	General Essential and Hierarchy Essential
Threshold	Yes	Number	General Properties
Currency	Yes	Currency	General Properties

B2BBudgetExceededPermission

Possessors of this permission are allowed to place orders that would result in exceeding the remaining budget.

This table lists the field used by this permission:

Field	Required	Type	Location in Backoffice
Parent B2B Unit	Yes	B2BUnit	General Essential and Hierarchy Essential

B2B User Groups

Every B2B customer is member of at least one B2B user group. A B2B user group is a Principal Group which contains a list of B2B permissions. All B2B customers that are members of a group are granted its permissions.

B2B user groups are accessed by navigating to B2B Commerce B2B User Groups and then selecting a user group. B2B user groups have the following required and optional fields. The **Location** column shows which tab and section to access the field for a selected B2B user group. The required fields may not be null, but may be modified:

Field	Required	Type	Location in Backoffice
Parent B2B Unit	Yes	B2BUnit	General Essential
B2B Permission(s)	No	B2BPermission	General Properties

B2B Units and Organization Tree

Customer-side organizations are composed of B2B units which are in a tree structure with one root node. A B2B unit may represent a department, sub-company or division within an organization.

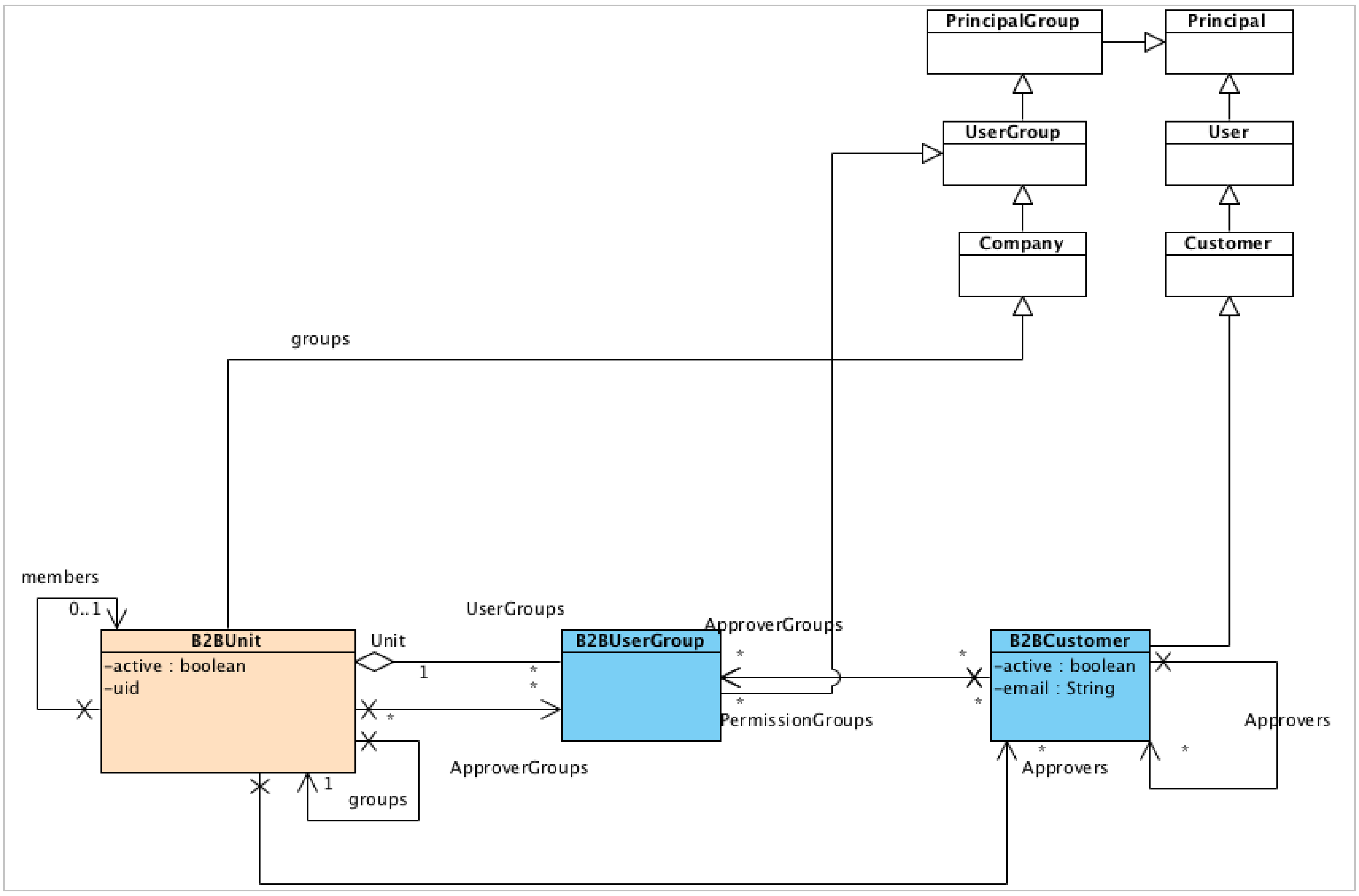
The root node is itself a B2B unit with no parent unit. Administrators on the customer-side can manage B2B units within their organizations or branches. For example, they can create and disable child units. Only an administrator on the merchant side can create a root node.

The structure items of B2B Commerce, such as budgets, cost centers, customers, and permissions are associated to a B2B unit and may also be managed by customer-side administrators. Visibility is based on branches, where the branch of the root node is in effect the whole organization. Customers cannot see or manage B2B Items which are not in their branch.

For B2B units and the organization tree, personalization rules and visibility restrictions, see b2bcommerce Extension. For more information on how to manage units and the organization tree, see Working with B2B Units.

B2B Types for Organization Tree

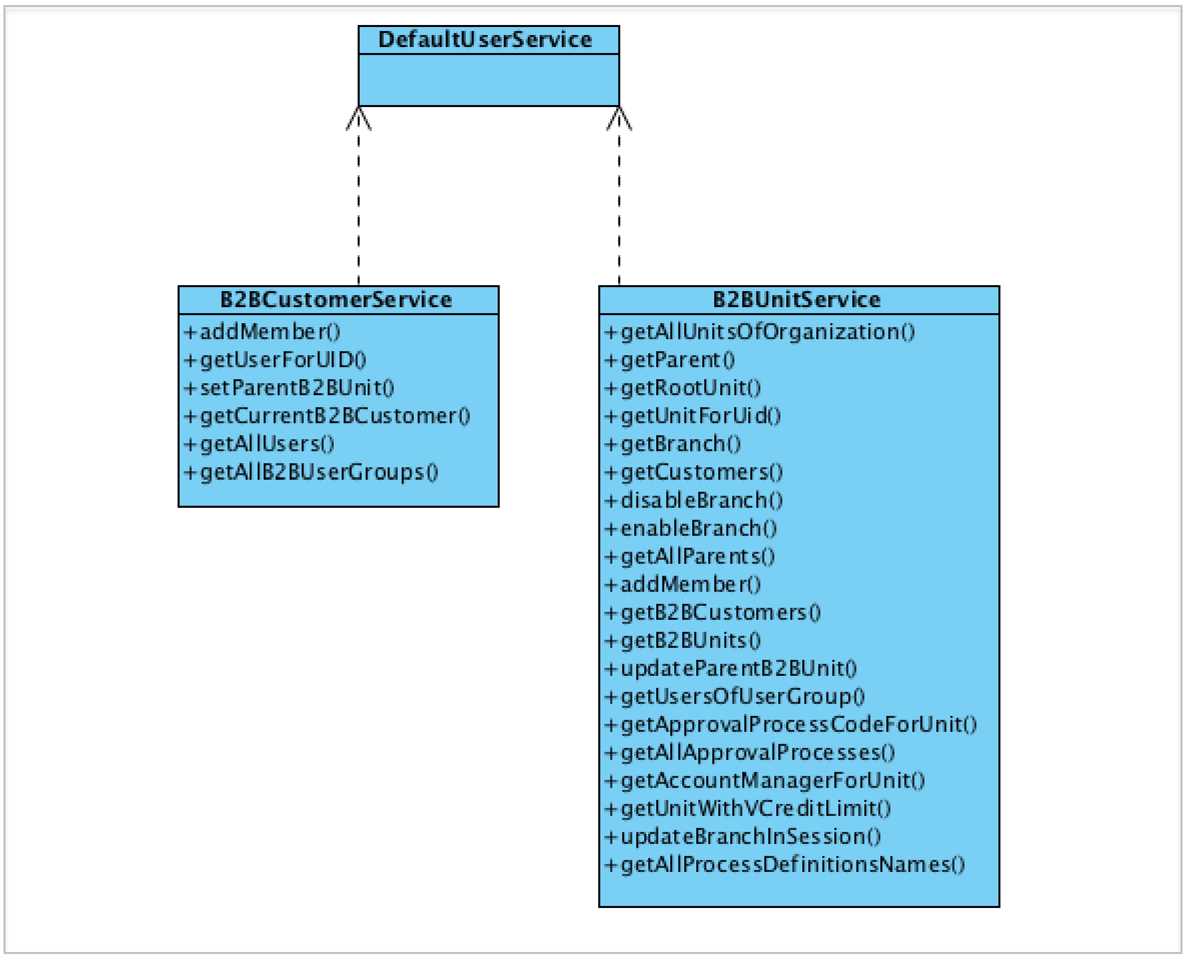
This section shows all types and services connected to B2B units.



`B2BUnit`, which extends `B2BUserGroup`, must be a member of one `B2BUnit` unless it is a root node.

B2B Services for Managing Organization

This section shows all methods behind `B2BUnitService` and `B2BCustomerService`.



Managing B2B Units with Services

The following sections provide examples on how to use some of the selected methods below.

Creating a B2B Unit

Only the system administrator can create a root B2B unit, which is a B2B unit without a parent unit. Customer-side administrators may create and add child units of this unit in their organization. If you attempt to create a B2B unit without a parent it fails unless you are the system administrator.

```
/*
             *  Sample method which creates a B2BUnit
             */
            public B2BUnitModel createSubB2BUnit(B2BUnitModel parent)
            {
                 final B2BUnitModel unit = (B2BUnitModel) modelService.create(B2BUnitModel.class);
                 unit.setUid("uniqueID");
                 unit.setName("AnyName");
                 unit.setGroups(Collections.EMPTY_SET);
                 b2bUnitService.addMember(parent, unit); // Set parent
                 this.modelService.save(unit);
                 return unit;
            }
```

Creation of the root B2B unit requires administration rights. You can use the `SessionService.executeInLocalView` to run code inside its own session context without affecting the current session.

You need to inject `sessionService` into your bean.

```
sessionService.executeInLocalView(new SessionExecutionBody()
                 {
                        @Override
                        public void executeWithoutResult()
                        {
                                sessionService.setAttribute(SessionContext.USER, userService.getUser(C
                               // Create root B2BUnit here
                               final B2BUnitModel unit = (B2BUnitModel) modelService.create(B2BUnitMoc
                               unit.setUid("uniqueID");
                               unit.setName("AnyName");
                               unit.setGroups(Collections.EMPTY_SET);
                               this.modelService.save(unit);
                        }
                });
```

Finding a B2B Unit

Finding a B2B Unit by UID

You can find a B2B unit by its UID, although if a B2B unit exists in a different branch than you are trying to load it to, system returns `null` as if it does not exist at all.

```
b2bUnitService.getUnitForUid(String uid);
```

Checking If UID for B2B Unit Is Unique

Both B2B Customers and B2B units are SAP Commerce principals. `UniqueID` must be unique across all principal types. Checking if a UID is already taken is used, for example for validation purposes. Visibility rules are suspended for such call.

```
b2bCustomerService.principalExists("some uniqueId")
```

Finding a B2B Unit by Session User

You can search for a B2B unit using Session User:

```
B2BCustomerModel b2bCustomer = b2bcustomerService.getCurrentB2BCustomer();
B2BUnitModel parent = b2bUnitService.getParent(b2bCustomer);
```

Disabling a B2B Unit

Deleting a B2B unit is not advisable due to dependencies. B2B Commerce provides a way to disable a B2B unit. When a B2B unit is disabled, all its B2B Customers, sub-units and their customers are also disabled. This ensures once organization or branch is disabled, commerce, and permissions are suspended.

```
b2bUnitService.disableBranch(final B2BUnitModel unit)
```

Related Information

[Working with B2B Units]

B2B Customer and Customer Rights

B2B customers can have permissions assigned to them through four different user groups. Each user group has specific permissions that are granted to its members.

Customers in the B2B Commerce Module are members of a B2B unit. They are represented by the **B2BCustomer** type. There are four special B2B groups that grant abilities and rights to their members:

- **B2BAdminGroup**: Members can create and manage other B2B items, such as cost centers, budgets, and customers.
- **B2BManagerGroup**: Members can only view reports.
- **B2BApproverGroup**: Members approve or reject orders.
- **B2BCustomerGroup**: New customers are automatically placed into this group, which grants the permissions to purchase.

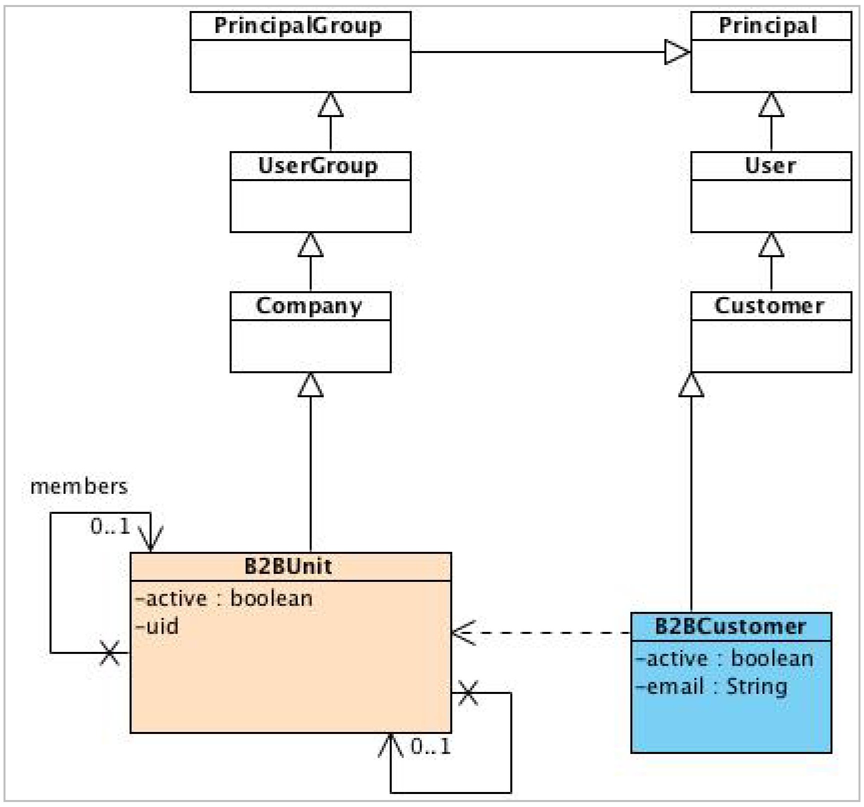
As with other B2B types, B2B customers are subject to visibility rules. Merchant administrators cannot view or manage B2B items not associated directly to their branch.

For information on how to manage customers and customers rights, see Managing Customer User Groups in Backoffice.

For B2B Units and Organization visibility restrictions, see B2B Units and Organization Tree.

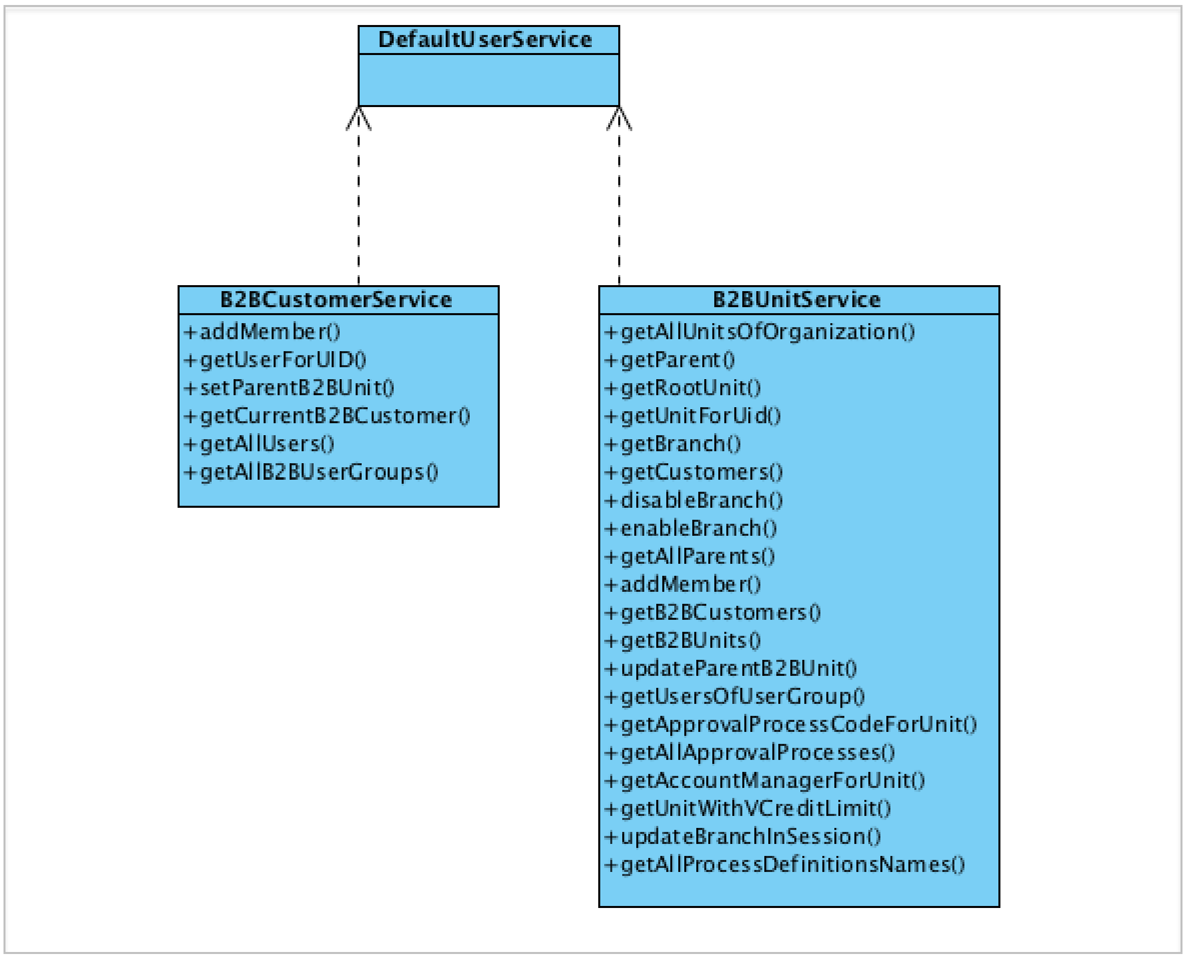
Types

This diagram shows all types and services connected to B2B customers.



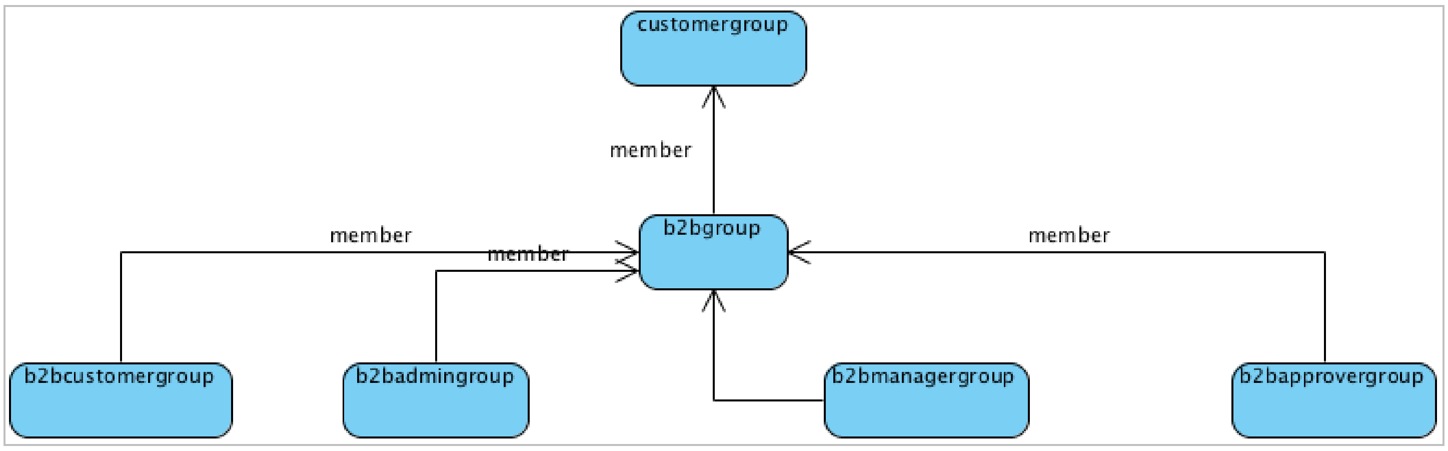
Services

This diagram shows all methods behind `B2BUnitService` and `B2BCustomerService`.



Propagation of Customer Rights

There are four special B2B user groups that grant rights:



You can find their IDs in `B2BCommerceConstants`:

```
public B2BCommerceConstants
{ ..
        public static final String B2BCUSTOMERGROUP = "b2bcustomergroup";
        public static final String B2BAPPROVERGROUP = "b2bapprovergroup";
        public static final String B2BADMINGROUP = "b2badmingroup";
        public static final String B2BMANAGERGROUP = "b2bmanagergroup";
}
```

The four user groups have following rights:

User Group	Description
b2bcustomergroup	Members of this group can view cost centers, create and view their orders.
b2bapprovergroup	Members of this group have all privileges of **b2bcustomergroup**, and can also approve or reject orders according to order approval workflow.
b2badmingroup **b2bmanagergroup**	Members of these groups have all privileges of **b2bcustomergroup**, and can also manage all items within their branch of organization tree.

Creating B2B Customers

Checking If UID Is Unique

B2B Customers and B2B Units are both principals. IDs must be unique to all principals. You may want to check if the UID is unique:

```
// Visibility rules do not apply.
     boolean isUnique = b2bCustomerService.principalExists("some uniqueId");
```

For more information, see Users in the Platform.

Adding a B2B Customer

Find example code snippet showing how to create a B2B customer:

```
B2BCustomerModel customer = modelService.create(B2BCustomerModel.class);
     customer.setName("Customer Name");
     customer.setUid("uniqueID");                                                        // Required,
     customer.setEmail("customer.name@example.com);                                      // Required
     b2bCustomerService.addMember(itemModel, b2bUnitService.getUnitForUid("parentID"));   // Parent B2
     modelService.save(item);
```

Extending a B2B Customer

You can either append attributes to the **B2BCustomer** item or extend from it. The following example from the `items.xml` file shows how to add an attribute to the customer:

```
<itemtype code="B2BCustomer" generate="true"
                        jaloclass="de.hybris.platform.b2b.jalo.B2BCustomer" extends="Customer"
                        autocreate="true">
                        <attributes>
                                <attribute qualifier="active" type="java.lang.Boolean">
                                        <modifiers read="true" write="true" search="true"
                                                optional="false" />
                                        <defaultvalue>java.lang.Boolean.TRUE</defaultvalue>
                                        <persistence type="property" />
                                </attribute>
                                <attribute qualifier="email" type="java.lang.String">
                                        <modifiers read="true" write="true" search="true"
                                                optional="false" />
                                        <persistence type="property" />
                                </attribute>
                        </attributes>
                </itemtype>
```

The following is an example from `items.xml` of how to extend a `B2BCustomer`:

```
<itemtype code="CustomB2BCustomer" generate="true"
                        jaloclass="com.mycompnay.jalo.CustomB2BCustomer" extends="B2BCustomer"
                        autocreate="true">
                        <attributes>
                                <attribute qualifier="customField" type="String">
                                        <modifiers read="true" write="true" search="true" />
                                        <persistence type="property" />
                                </attribute>
                        </attributes>
                </itemtype>
```

Adding Customer Rights to a B2B User Group

You can add certain rights to a B2B User Group.

Here is an example of adding rights:

```
B2BCustomerModel customer =  b2bItemFacade.getB2BCustomerForCode("uniqueID");
                final Set<PrincipalGroupModel> groups = new HashSet<PrincipalGroupModel>(itemModel.get
                groups.add(UserService.getUserGroupForUID(B2BCommerceConstants.B2BADMINGROUP));
                customer.setGroups(groups);
                modelService.save(item);
```

Disabling a B2B Customer

You can delete B2B customers effectively by setting their active status to disabled. They do not show up in subsequent searches and users are not able to log in and therefore place orders.

Here is an example of deleting of a B2B customer:

```
B2BCustomerModel customer = b2bCustomerService.getB2BCustomerForCode("uniqueID");
       customer.setActive(Boolean.FALSE);
       modelService.save(item);
```

Related Information

Managing Customer User Groups in Backoffice

B2B Budgets and Cost Centers

Learn about the types and services associated with B2B budgets and cost centers, as well as how to create a budget and a cost center.

Companies need to manage their cost centers and budgets. Cost centers are used for accounting purposes. Orders or line items of an order are charged against a cost center. Budgets help the user monitor their spending. Budgets are assigned to cost centers. You can assign right to maintain cost centers and budgets to the company administrator or any other user.

By default users who belong to **b2badmingroup** or **b2bmanager** group are able to create and edit cost centers and budgets.

For information on how to manage budgets and cost centers using the Backoffice Administration Cockpit or B2B Store front end, see Working with Budgets.

Visibility of Cost Centers and Budgets

Only the budgets and cost centers assigned to the B2B Unit of the logged in B2B Admin or B2B Manager and the units of their branch can be edited.

i Note

Cost centers and budgets are currency sensitive. You need to make sure to create a separate cost center and budget for each currency which you use.

Retrieving Available Cost Centers for the Current Session

The code snippet below demonstrates how to retrieve cost centers of a B2B branch for a specific currency:

```
b2bCostCenterService.getCostCentersForUnitBranch(b2bUnitService.getParent((B2BCustomerModel) userModel
```

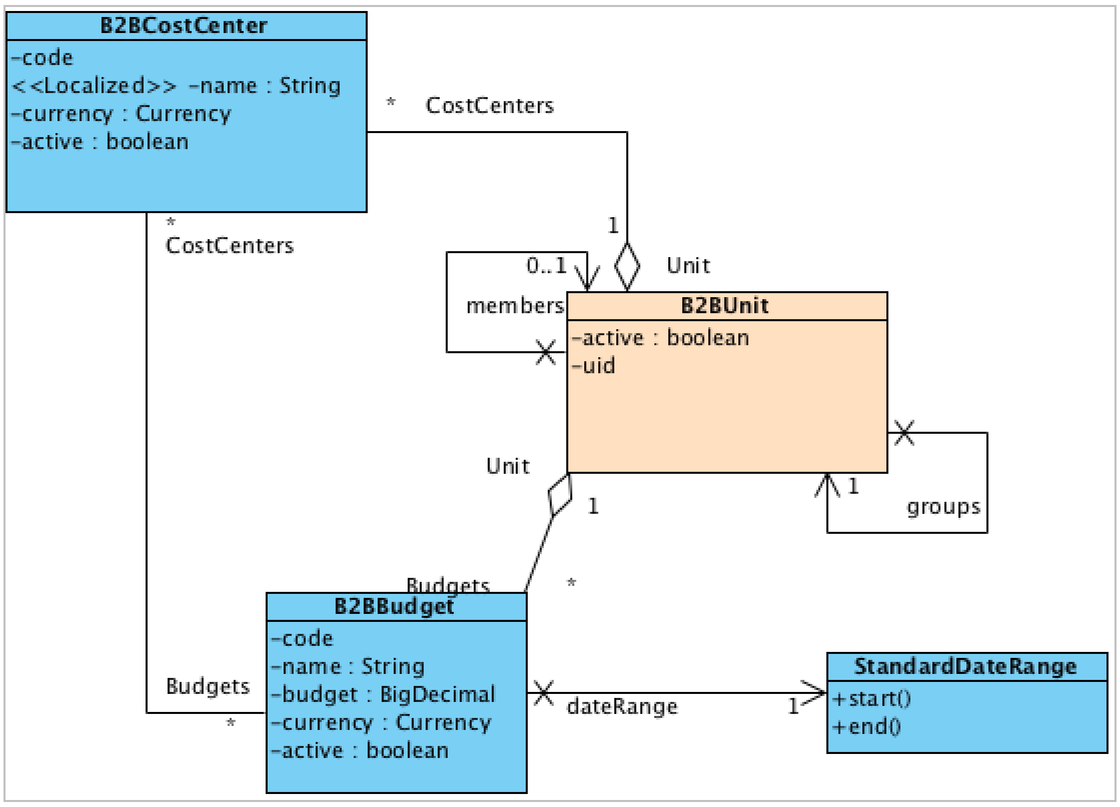
Retrieving a Budget from a Cost Center That Is Active and Has Not Expired

The code snippet below demonstrates how to retrieve a budget from a cost center associated with an **OrderEntry**:

```
// aggregate cost centers from order entries
final Set<B2BCostCenterModel> b2bCostCenters = orderUtils.getB2BCostCenters(order.getEntries());
for (final B2BCostCenterModel b2bCostCenterModel : b2bCostCenters)
{
   final B2BBudgetModel currentBudget = b2bCostCenterService.getCurrentBudget(b2bCostCenterModel);
}
```

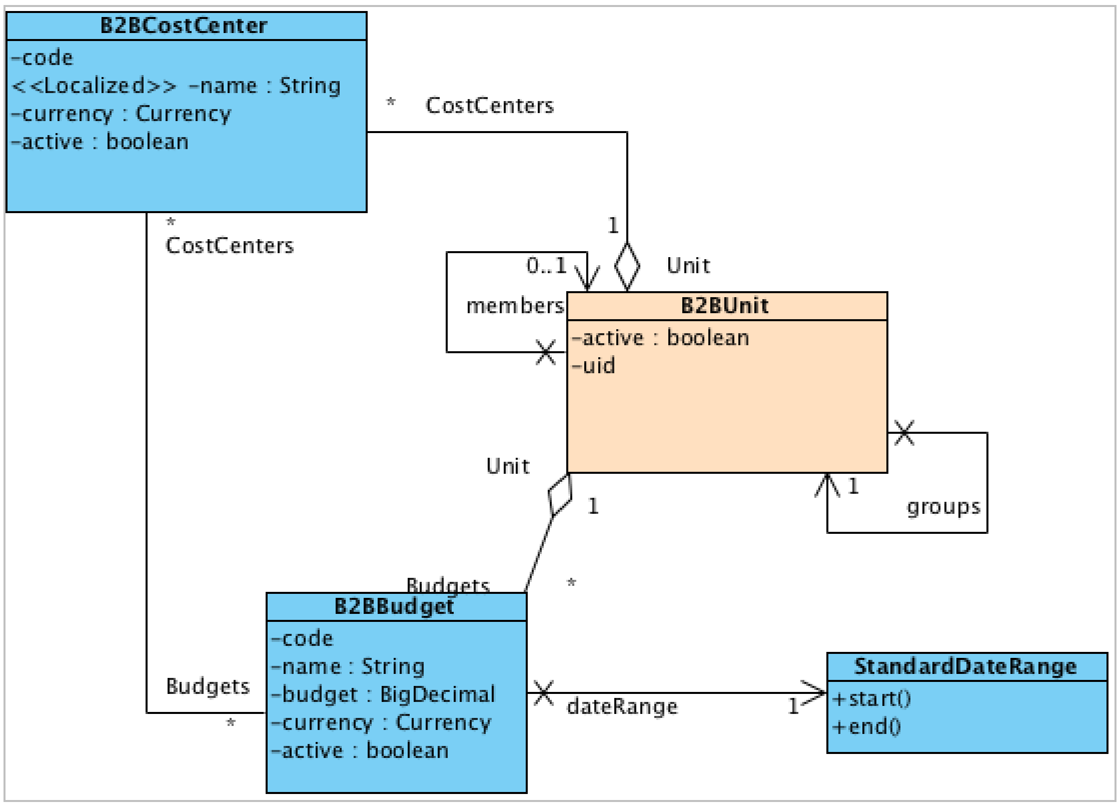
Types

This section shows all types and services connected to B2B Cost Centers and Budgets.



Services

`B2BCostCenterService` provides methods for looking up cost centers available to a branch or a B2B Unit. It can also retrieve available currencies based on the session.



Creating a Cost Center and a Budget

Find an example of creating a cost center and a budget below:

```
final UserModel user = userService.getUser("b2badmin");
                        final B2BUnitModel unit  = b2bUnitService.getParent(((B2BCustomerModel)user));
                        final B2BCostCenterModel costCenter = (B2BCostCenterModel) modelService.create
                        costCenter.setCode("costcenter1");
                        costCenter.setUnit(unit);
                        costCenter.setCurrency(I18NService.getCurrency("USD"));

                        final B2BBudgetModel budgetModel = (B2BBudgetModel) modelService.create(B2BBuc
                        budgetModel.setCode("budget1");
                        budgetModel.setUnit(unit);
                        budgetModel.setBudget(java.math.BigDecimal.valueOf(budget.doubleValue()));
                        budgetModel.setCurrency(I18NService.getCurrency("USD"));
                        budgetModel.setDateRange(B2BDateUtils.createDateRange(B2BPeriodRange.YEAR));

                        final HashSet<B2BBudgetModel> b2bBudgetModels = new HashSet();
                        b2bBudgetModels.add(budget);
                        costCenter.setBudgets(b2bBudgetModels);
                        modelService.save(costCenter);
```

Related Information

Working with Budgets

B2B Price, Product and Catalog Management

Price, product, and catalog management within the B2B Commerce Module is conducted using the `impex` extension. The following sections describe B2B data management using ImpEx.

Product and catalog management within the B2B Commerce Module uses technology from the `catalog` extension.

Setting Up B2B Unit-Specific Pricing

This example is based on data available in the `sampledata` extension. It shows how to assign prices for the product HW2120-0341 for 3 B2B units: IC Sales DE, IC Sales UK, and IC Sales US. This is achieved by creating price rows with a **userPriceGroups** and then associating those user price groups to **B2BUnit**.

```
$catalogVersion=catalogVersion(catalog(id),version)[unique=true];
###############################################################################
# Create UserPriceGroups
###############################################################################
INSERT_UPDATE UserPriceGroup;code[unique=true];name[lang=en];
;IC_UK;IC UK PRICING;
;IC_EUROPE;IC EUROPE PRICING;
;IC_US;IC US PRICING;

###############################################################################
# Create price rows and associate them to user price groups.
###############################################################################
INSERT_UPDATE PriceRow; product(code, catalogVersion(catalog(id),version))[unique=true];ug(code)[uniqu
;HW2120-0341:hwcatalog:Staged;IC_UK;GBP;111;hwcatalog:Staged;1;
;HW2120-0341:hwcatalog:Staged;IC_UK;GBP;99;hwcatalog:Staged;3;
;HW2120-0341:hwcatalog:Staged;IC_UK;GBP;77;hwcatalog:Staged;5;
;HW2120-0341:hwcatalog:Staged;IC_EUROPE;EUR;111;hwcatalog:Staged;1;
;HW2120-0341:hwcatalog:Staged;IC_EUROPE;EUR;99;hwcatalog:Staged;3;
;HW2120-0341:hwcatalog:Staged;IC_EUROPE;EUR;77;hwcatalog:Staged;5;
;HW2120-0341:hwcatalog:Staged;IC_US;EUR;111;hwcatalog:Staged;1;
;HW2120-0341:hwcatalog:Staged;IC_US;EUR;99;hwcatalog:Staged;3;
;HW2120-0341:hwcatalog:Staged;IC_US;EUR;77;hwcatalog:Staged;5;

###############################################################################
# Assign userprice group to b2bUnits
###############################################################################
"update B2BUnit";"uid[unique=true,allownull=true]";"userPriceGroup(code,itemtype(code))"
;"IC Sales DE";"IC_EUROPE:UserPriceGroup"
;"IC Sales UK";"IC_UK:UserPriceGroup"
;"IC Sales US";"IC_US:UserPriceGroup"
```

Note

After you load the script, you have to synchronize the data. For more information on synchronization, see Synchronizing Catalogs.

Visibility of Products Based on B2B Unit

The example below illustrates how to assign the visibility of a category to a **B2BUnit** of GC organization, and how to assign products to the category.

```
$catalogversion=catalogversion(catalog(id[default='hwcatalog']),version[default='Staged'])[unique=true
$supercategories=supercategories(code,catalogversion(catalog(id[default='hwcatalog']),version[default=
$thumbnail=thumbnail(code,catalogversion(catalog(id[default='hwcatalog']),version[default='Staged']))

INSERT_UPDATE category;code[unique=true];$catalogversion;name[lang=de];name[lang=en];$supercategories;
;specials;;;Specials;;;;;5

INSERT_UPDATE category;code[unique=true];$catalogversion;allowedPrincipals(uid);
;specials;;GC;

########## mainboards ############
INSERT_UPDATE product;code[unique=true];$catalogversion;$supercategories;
;HW2200-0561;;specials
;HW2200-0623;;specials
;HW2200-0812;;specials
;HW2200-0521;;specials
;HW2200-0878;;specials
```

Note

After you load the script, you have to synchronize the data. For more information on synchronization, see Synchronizing Catalogs.

Related Information

catalog Extension
ImpEx
Catalog Guide
Setting Customized Prices
Setting Product Visibility

B2B Orders

The following sections discuss the technical aspects of order management within the B2B Commerce Module. The technology behind orders is based on the `basecommerce` extension services.

For more information on the `basecommerce` extension, see basecommerce Extension.

Order Replenishment

The order replenishment is used to create orders that should be placed on a scheduled basis. For replenishment orders the same rules apply as when placing an order using the front end, especially the approval process should be executed. You can find more information on order replenishment in About Replenishment and Order Scheduling.

In order to make sure that the configured approval process is executed when an order is placed using the scheduled order job, the `b2bOrderService` is injected into the `orderUtility`, which in turn is used by the order replenishment when placing an order. This is shown in the following example from `b2bstore-spring.xml`.

```
<!-- for order replenishment we need to inject b2b version of order service
                into this bean. -->
        <bean id="orderUtility"
                class="de.hybris.platform.orderscheduling.impl.DefaultOrderUtilityImpl">
                <property name="modelService" ref="modelService" />
                <property name="orderService" ref="b2bOrderService" />
        </bean>
```

To avoid side effects, the `b2bOrderService` is made context-aware and checks if an order is placed in a B2B context by checking the user being from type `b2bcustomer`.

The following is an example from the `DefaultCartFacade` of the B2B Commerce Module that creates a scheduled order from a cart:

```
public CartToOrderCronJobModel scheduleOrderFromCart(final AddressData deliveryAddress, final TriggerM
                        throws InvalidCartException
        {
                final CartModel cart = b2bCartService.getSessionCart();
                final CartModel clone = modelService.clone(cart);
                <...>
                b2bCartService.removeSessionCart();
                return scheduleOrderService.createOrderFromCartCronJob(clone, delivery, null, null, Cc
        }
```

Cloning the cart is necessary to make sure the cart does not get removed when the session is closed. There were no extra WCMS components necessary to include scheduling functionality it was incorporated as a part of the checkout process.

For more information on creating a scheduled order, see About Replenishment and Order Scheduling.

Returns

The returns feature of B2B Commerce uses the order management return service. For more information, see Return Service Configuration.

Here is an example taken from the B2B store. Note that the return process was simplified, so all return requests get an immediate refund for reason **GOODWILL**, with no need for Customer Service interaction or additional WCMS components. To request a return, the B2B store simply adds a drop-down on the order view in question to each remaining product in order. The following is an example of the `OrderReturnController`.

```
private String prepareReturns(final OrderReturnInfo data)
        {
                ReturnRequestModel request = null;
                final String rma = null;

                for (int i = 0; i < data.getReturnQuantities().size(); i++)
                {
                        if (data.getReturnQuantities().get(i) > 0)
                        {
                                final OrderModel order = orderApprovalFacade.getOrderByCode(data.getOr
                                if (request == null)
                                {
                                        request = returnService.createReturnRequest(order);
                                        request.setRMA(returnService.createRMA(request));
                                }
                                // This automatically created a refund
                                returnService.createRefund(request, order.getEntries().get(i), "no.1",
                                                Long.valueOf(data.getReturnQuantities().get(i)), Retur
                        }
                }
                return rma;
        }
```

Checkout

The main difference between the B2B checkout and the B2C checkout scenarios is the need to have a cost center set on the order line items, and that an approval process is started after the order is placed.

The B2B checkout process is based on the standard checkout process offered with SAP Commerce. For more information about order services in the hybris `basecommerce` extension, see basecommerce Extension.

Cost Center

The cost center is set when a product is added to the cart:

```
public void addToCart(final String productCode, final long quantity, final String costCenterCode)
    {
        final B2BCostCenterModel costCenter = b2bCostCenterService.getCostCenterForCode(costCenterCode
        <...>
```

Approval Process

After placing an order, the approval process starts. The following is an example of the `DefaultB2BPlaceOrderStrategy`.

```
public OrderModel placeOrder(final CartModel cart, final AddressModel deliveryAddress, final AddressMo
                        final PaymentInfoModel paymentInfo) throws InvalidCartException
        {
                final OrderModel order = this.getPlaceOrderStrategy().placeOrder(cart, deliveryAddress

                final Map<String, Object> contextParams = new HashMap<String, Object>();
                contextParams.put(ProcessengineConstants.EVENT_AFTER_WORKFLOW_PARAM_NAME,
                                B2BCommerceConstants.APPROVAL_WORKFLOW_COMPLETE_EVENT);

                final B2BApprovalProcessModel aprovalProcess = (B2BApprovalProcessModel) businessProce
                                String.valueOf(processCodeGenerator.generate()), this.getApprovalProce
                aprovalProcess.setOrder(order);
                this.getModelService().save(aprovalProcess);
                businessProcessService.startProcess(aprovalProcess);
                return order;
        }
```

Using the injected strategy, the configured process is started. You define which process the system starts in the `approvalProcessName` Spring property. The following is an example from `b2bcommerce-spring.xml`.

```
 <bean id="defaultB2BPlaceOrderStrategy"
                class="de.hybris.platform.b2b.strategies.impl.DefaultB2BPlaceOrderStrategy"
                parent="abstractBusinessService">
                <property name="placeOrderStrategy" ref="placeOrderStrategy" />
                <property name="businessProcessService" ref="businessProcessService" />
                <property name="processCodeGenerator" ref="processCodeGenerator" />
                <property name="approvalProcessName" value="approval" />
        </bean>
```

This way you can configure and execute custom processes.

For more information on order approvals, see b2bapprovalprocess Extension

Delivery

B2B Commerce delivery makes use of default order management classes. Find more information in basecommerce Extension documentation.

Invoicing

The B2B Commerce invoicing uses the standard Web Services API.

Related Information

About Replenishment and Order Scheduling
Return Service Configuration

b2bcommercefacades Extension

The `b2bcommercefacades` extension provides a number of facades that are backed by the services in the `b2bcommerce` extension.

This facade has been migrated from `b2bacceleratorfacades` and refactored into smaller facades, where applicable, in order to have a cleaner separation of concern.

i Note

An SAP Commerce extension may provide functionality that is licensed through different SAP Commerce modules. Make sure to limit your implementation to the features defined in your contract license. In case of doubt, please contact your sales representative.

B2B Facades

The following facades expose B2B services from the `b2bcommerce` extension:

- `B2BBudgetFacade`
- `B2BCostCenterFacade`
- `B2BUnitFacade`
- `B2BUserFacade`
- `B2BUserGroupFacade`