Promotions

Overview

Companies can win new customers and increase their conversion rate with the hybris Promotion Module. Any type of promotion can flexibly be administered with the hybris user interface and quickly integrated to the online store. Using the **hybris Promotion Module**, you can easily create, edit and control various campaigns. It does not matter whether you wish to apply your promotions to the entire shop or just to selected products.

The **Promotion Module** consists of a code component that may be used to implement customer sales promotion functionality within web sites and a hybris Management Console(hMC) component for creating and managing the promotion data.

The website code component exposes information about available promotions for use in web site front end code and facilitates the calculation of the effect of promotions on shopping cart and order state. This functionality supports the following tasks:

- Presentation of available promotions to customers
- Application of activated promotions to shopping carts and orders
- Evaluation of partially activated promotions in shopping carts

A promotion means here a single customer sales promotion. Promotions contain a rule set that defines the conditions for activation and a set of operations which determine the effect of the promotion once activated.

Benefits

The key benefits of using the Promotion Module are as follows:

- Low-cost and effective marketing campaigns
- Gaining new customers as a result of attractive and customer-oriented promotions
- Boosting sales with product bundles and combined offers
- Special sale of discounted products (e.g remaining stock)
- Unified branding thanks to graphical and textual promotion design

Promotion Classes and Types

The following classes of promotion are included with the **promotions** extension:

- **Product level** promotions are activated based upon the products within the line items of a shopping cart or order.
- **Order level** promotions are activated based upon shopping cart or order attributes.

Additional promotion types may be developed as required.

Product Level Promotions

The complete list of out-of-the-box promotion types is given below. It is possible to identify what products qualify for a promotion by the direct assignment of products and/or categories. Any product that belongs to a qualifying category is deemed qualified for the promotion. Promotion restrictions are required to further filter the qualifying product list.

	Promotion	Description	Example
1	Bundle	Purchase one of each product from a defined set for a combined total price.	Buy A, B, and C for €50.
2	Buy X get Y free	Purchase a certain number of products from within a defined set and add further products from the same set at no additional cost.	Buy one get one free.
3	Fixed price	Purchase from within a defined set at a fixed unit price.	All shirts €5 each.
4	Multi-buy	Purchase a certain number of products from within a defined set for a fixed price.	Buy any 3 shirts for €50.
5	Stepped multi-buy	Purchase a number of products from within a defined set, there are multiple tiers of product quantities and fixed prices.	Buy any 3 shirts for €50, 5 for €65, and 7 for €75.
6	Perfect partner	Purchase a certain product from within a defined set and another partner product from a different defined set and pay a fixed price for the partner product.	Buy a game for €10 with each games console.

	Promotion	Description	Example
7	One-to-one perfect partner bundle	Purchase a certain product and another defined partner product for a fixed total price. The cart must contain the base product and the partner product to qualify.	Buy this game and the selected partner accessory together for €25.00.
8	Perfect partner bundle	Purchase a certain product along with a specified number of products from within a defined set for a combined total price.	Buy a games console and 3 accessories for €200.
9	Percentage discount	Receive a percentage discount on all products within a defined set.	20% off all cameras.

Order Level Promotions

	Promotion	Description	Example
1	Order threshold fixed discount	A fixed value discount is applied to the order when the threshold order value is exceeded.	Spend over €50 to receive a €3 discount.
2	Order threshold perfect partner	Purchase a certain product from within a defined set for a fixed price when the threshold order value is exceeded.	Spend over €50 to get any shirt for €5.
3	Order threshold free gift	A free gift is added to the order when the threshold order value is exceeded.	Spend over €50 to receive a free t-shirt.
4	Order threshold free voucher	A free voucher is given out when the order reaches a certain value.	Get a free €5 voucher when you spend over €150.00. Get a free voucher when your order subtotal is at least the threshold value
5	Order threshold change delivery mode	A different delivery mode is applied to the order when the threshold order value is exceeded.	Spend over €10 to get free shipping

Promotion Evaluation

Promotions are evaluated against a shopping cart or order by the promotions extension. There are two modes of evaluation:

- evaluate which of the available promotions have been activated or "fired"
- evaluate which of the available promotions have been partially activated or "could fire".

Partially Activated Promotions

Promotions are partially activated when some of the activation criteria have been met;

e.g. when two qualifying products from a three product multi-buy have been added to the shopping cart. Partially activated promotions do not "consume" line items from the shopping cart (a consumed line item is one that has been used in a fired promotion and cannot be used by another promotion), and these are still available for use in evaluating additional partially activated promotions. Indeed, a single product in a line item may be associated with several partially activated promotions, for example a perfect partner and a multi-buy promotion. It is useful to know that a promotion has been partially activated so that this may be communicated to customers on a website.

For instance, a message such as "Add another snowboard to your basket and receive 50% off your order" could be displayed.

Promotion Priority

Each promotion has a priority and promotions are evaluated in priority order, highest priority first. Typically product promotions are evaluated before order promotions; however this is entirely controlled by the priority.

Line items in the cart or order are "consumed" by fired promotions and become ineligible for inclusion in subsequent promotions during the evaluation.

Every promotion result has a certainty that is a fraction number between 0 and 1. The closer the promotion actually is to firing the closer it is to 1. A promotion result also has a consumed count that indicates how many items in the cart can currently be consumed by the promotion. These two attributes can be used to order or filter promotions that are not close to firing.

For instance, it may be desired that a "buy 4 get 1" promotion is only mentioned to the user when he already has 2 items in the cart.

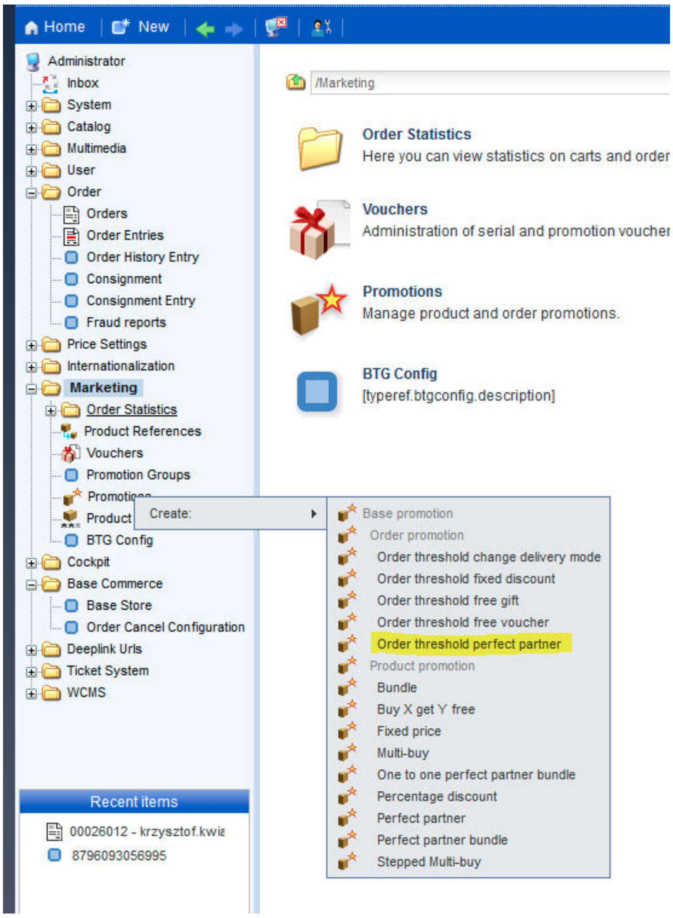
Task1:-Set up Promotion of Existing Type

Below, we will create an **Order threshold perfect partner** promotion.
This kind of promotion allows customers to get a *partner* product at a fixed price when the order subtotal is greater or equal to the threshold value.

For example: Get a USB Stick for €5.00 when you spend over €100.00 or for £4.00 when you spend over £80.00.

Create a new promotion instance in the hMC:

Expand the *Marketing* tree node, right click on the *Promotions* entry and create a new *Order threshold perfect partner* promotion:



Here you can view statistics on carts and order

Administration of serial and promotion voucher

Manage product and order promotions.

[typeref.btgconfig.description]

Fill in the promotion form :

Identifier	Give the promotion an unique identifier and description, ex.: **getUsbStickOver100**.
Promotion Group	Assign it to the **hybrisPromoGrp** promotion group.

Identifier

Give the promotion an unique identifier and description, ex.: **getUsbStickOver100**.

Enabled

Make sure this option is selected.

Priority

Provide a priority, ex.: **1000**.

Partner product

Choose **hybris USB Stick** (promoted product in our promotion)

Make sure and select the **online** version of the product.

Product price

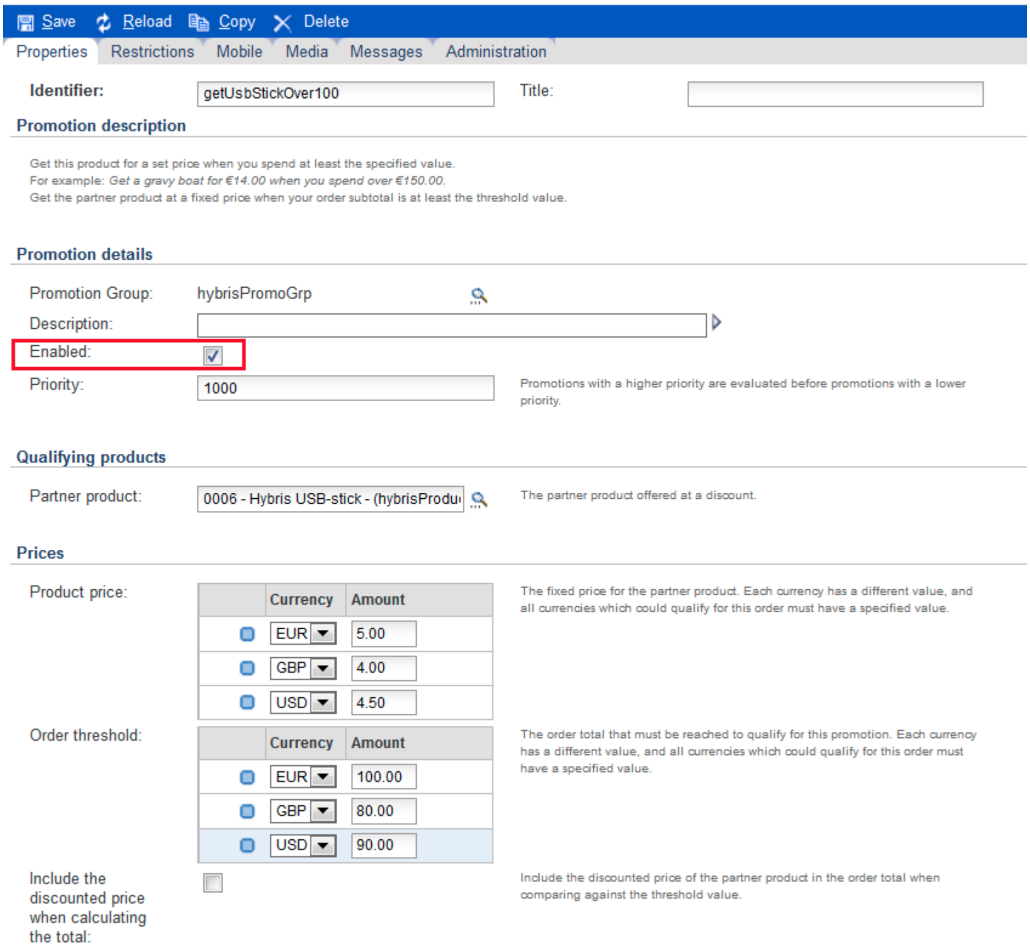
Create the following price rows by right-clicking in the table:

- **EUR**: 5.00
- **GBP**: 4.00
- **USD**: 4.50

Order threshold

Create the following price rows by right-clicking in the table:

- **EUR**: 100.00
- **GBP**: 80.00
- **USD**: 90.00



Identifier: getUsbStickOver100 Title:

Promotion description

Get this product for a set price when you spend at least the specified value.
For example: *Get a gravy boat for €14.00 when you spend over €150.00.*
Get the partner product at a fixed price when your order subtotal is at least the threshold value.

Promotion details

Promotion Group: hybrisPromoGrp

Description:

Enabled: ☑

Priority: 1000 — Promotions with a higher priority are evaluated before promotions with a lower priority.

Qualifying products

Partner product: 0006 - Hybris USB-stick - (hybrisProdu — The partner product offered at a discount.

Prices

Product price: — The fixed price for the partner product. Each currency has a different value, and all currencies which could qualify for this order must have a specified value.

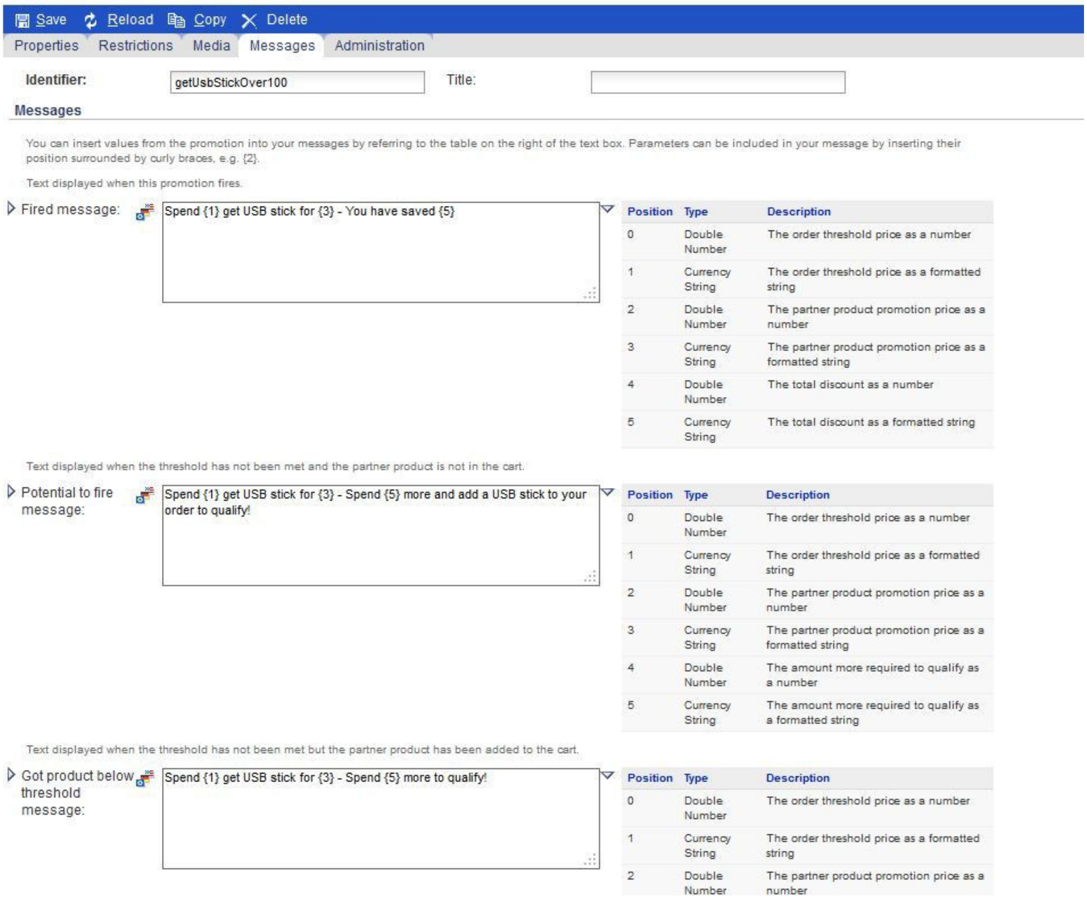
	Currency	Amount
●	EUR	5.00
●	GBP	4.00
●	USD	4.50

Order threshold: — The order total that must be reached to qualify for this promotion. Each currency has a different value, and all currencies which could qualify for this order must have a specified value.

	Currency	Amount
●	EUR	100.00
●	GBP	80.00
●	USD	90.00

Include the discounted price when calculating the total: □ — Include the discounted price of the partner product in the order total when comparing against the threshold value.

Also have a look at the *Messages* tab and change the messages accordingly (replace *productName* by **USB stick**).



Identifier: getUsbStickOver100

Title:

Messages

You can insert values from the promotion into your messages by referring to the table on the right of the text box. Parameters can be included in your message by inserting their position surrounded by curly braces, e.g. {2}.

Text displayed when this promotion fires.

Fired message: Spend {1} get USB stick for {3} - You have saved {5}

Position	Type	Description
0	Double Number	The order threshold price as a number
1	Currency String	The order threshold price as a formatted string
2	Double Number	The partner product promotion price as a number
3	Currency String	The partner product promotion price as a formatted string
4	Double Number	The total discount as a number
5	Currency String	The total discount as a formatted string

Text displayed when the threshold has not been met and the partner product is not in the cart.

Potential to fire message: Spend {1} get USB stick for {3} - Spend {5} more and add a USB stick to your order to qualify!

Position	Type	Description
0	Double Number	The order threshold price as a number
1	Currency String	The order threshold price as a formatted string
2	Double Number	The partner product promotion price as a number
3	Currency String	The partner product promotion price as a formatted string
4	Double Number	The amount more required to qualify as a number
5	Currency String	The amount more required to qualify as a formatted string

Text displayed when the threshold has not been met but the partner product has been added to the cart.

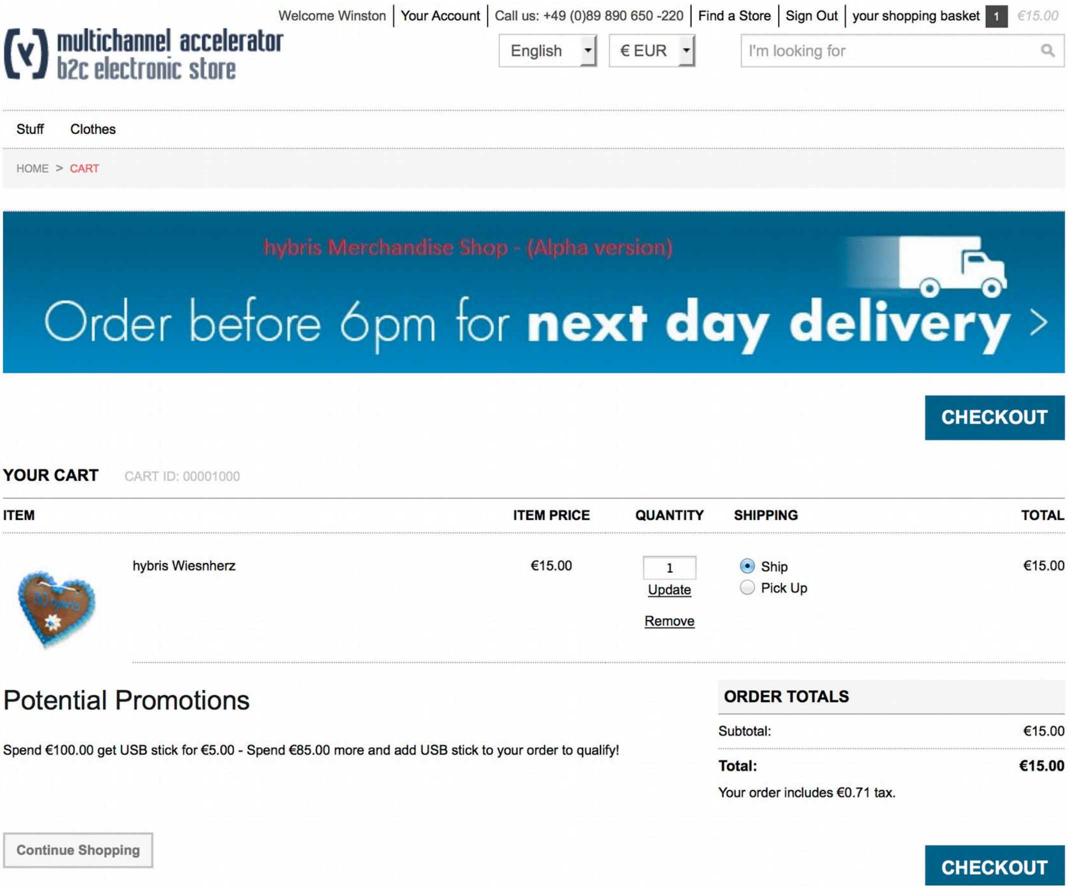
Got product below threshold message: Spend {1} get USB stick for {3} - Spend {5} more to qualify!

Position	Type	Description
0	Double Number	The order threshold price as a number
1	Currency String	The order threshold price as a formatted string
2	Double Number	The partner product promotion price as a number

Save the promotion instance by clicking on the *Save* button.

Verify the promotion in the checkout process:

Add some product to the cart along with our partner product, i.e. the usb stick. Notice the information about the promotion that you COULD qualify for!



Welcome Winston | Your Account | Call us: +49 (0)89 890 650 -220 | Find a Store | Sign Out | your shopping basket 1 €15.00

multichannel accelerator b2c electronic store

English
€ EUR
I'm looking for
Stuff
Clothes
HOME > CART
hybris Merchandise Shop - (Alpha version)
Order before 6pm for next day delivery >
CHECKOUT

YOUR CART CART ID: 00001000

ITEM		ITEM PRICE	QUANTITY		SHIPPING	TOTAL
	hybris Wiesnherz	€15.00	1 Update Remove		● Ship ○ Pick Up	€15.00

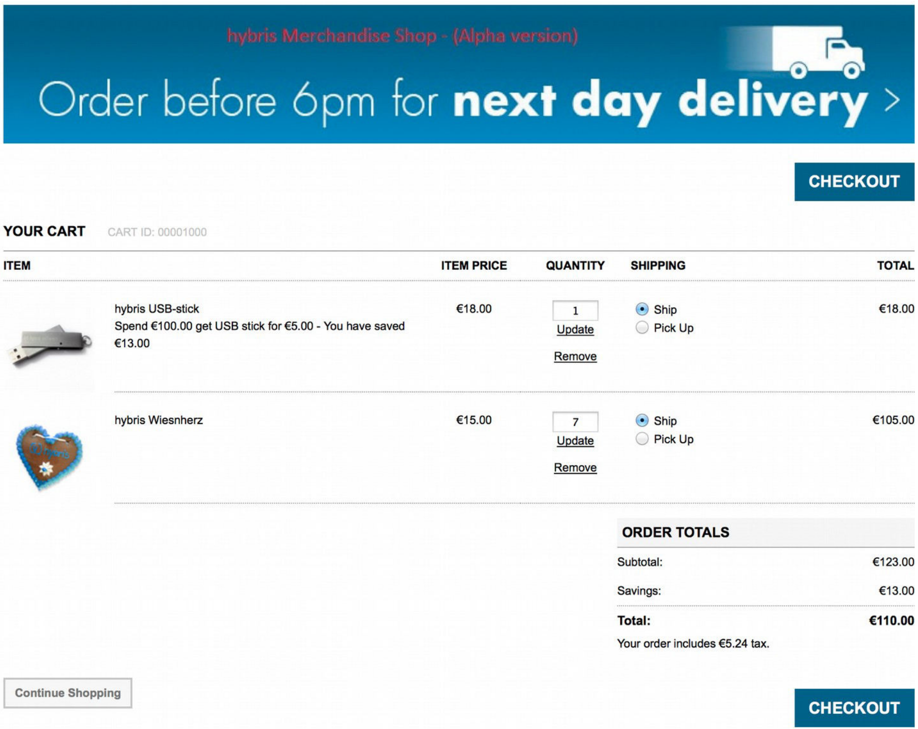
Potential Promotions

Spend €100.00 get USB stick for €5.00 - Spend €85.00 more and add USB stick to your order to qualify!

ORDER TOTALS	
Subtotal:	€15.00
Total:	**€15.00**

Your order includes €0.71 tax.

If you feel swayed by our pithy marketing pitch and add more products to the cart to reach the promotion threshold, you will see that your cart now qualifies for our new promotion:



hybris Merchandise Shop - (Alpha version)
Order before 6pm for next day delivery >
CHECKOUT

YOUR CART CART ID: 00001000

ITEM		ITEM PRICE	QUANTITY	SHIPPING	TOTAL
	hybris USB-stick Spend €100.00 get USB stick for €5.00 - You have saved €13.00	€18.00	1 Update Remove	● Ship ○ Pick Up	€18.00
	hybris Wiesnherz	€15.00	7 Update Remove	● Ship ○ Pick Up	€105.00

ORDER TOTALS	
Subtotal:	€123.00
Savings:	€13.00
Total:	**€110.00**

Your order includes €5.24 tax.

Continue Shopping
CHECKOUT

Task2:-Define New Promotion Type

Imagine that you need a new type of promotion: Buy X items of product A, get product B for free!.
The difference between our new type and the standard BOGOF (Buy 1 Get 1 Free) promotion is that ours allows the free product to be different from the qualifying products.

For instance, *Buy 5 A5 notepads, get a hybris pen for free*. Let's then define a promotion that would apply to the following cart:

Define a new promotion type

A promotion type is an entity we need to define in our project's items.xml.
Let's add a new promotion type that:

- extends from ProductPromotion
- defines **qualifyingCount** integer attribute
- defines **giftProduct** attribute of type Product
- defines the following localized String messages attributes : **messageFired** and **messageCouldHaveFired**

<YourExtetntion>commerceservices-items.xml

```
...
    <itemtype code="ProductQuantityThresholdFreeGiftPromotion"
      extends="ProductPromotion"
      jaloclass="de.hybris.merchandise.jalo.ProductQuantityThresholdFreeGiftPromotion"
      autocreate="true"
      generate="true">

      <attributes>

        <attribute
          qualifier="giftProduct"
          autocreate="true"
          type="Product">
          <description>The partner product.</description>
          <modifiers read="true" write="true" search="true" optional="true"/>
          <persistence type="property"/>
        </attribute>

        <attribute
          qualifier="qualifyingCount"
          autocreate="true"
          type="java.lang.Integer">
          <defaultvalue>Integer.valueOf(3)</defaultvalue>
          <description>The number of base products required to qualify for the free
gift.</description>
          <modifiers read="true" write="true" search="true" optional="true"/>
          <persistence type="property"/>
        </attribute>

        <attribute qualifier="messageFired" type="localized:java.lang.String">
          <description>The message to show when the promotion has fired.</description>
          <modifiers read="true" write="true" optional="true" />
          <persistence type="property">
            <columntype database="oracle">
              <value>varchar2(4000)</value>
            </columntype>
            <columntype database="mysql">
              <value>text</value>
            </columntype>
            <columntype database="sqlserver">
              <value>nvarchar(max)</value>
            </columntype>
            <columntype database="hsqldb">
              <value>LONGVARCHAR</value>
            </columntype>
            <columntype>
              <value>varchar</value>
            </columntype>
          </persistence>
        </attribute>

        <attribute qualifier="messageCouldHaveFired" type="localized:java.lang.String">
          <description>The message to show when the promotion could have potentially
fire.</description>
          <modifiers read="true" write="true" optional="true" />
          <persistence type="property">
            <columntype database="oracle">
              <value>varchar2(4000)</value>
            </columntype>
            <columntype database="mysql">
              <value>text</value>
            </columntype>
            <columntype database="sqlserver">
              <value>nvarchar(max)</value>
            </columntype>
            <columntype database="hsqldb">
              <value>LONGVARCHAR</value>
            </columntype>
            <columntype>
              <value>varchar</value>
            </columntype>
          </persistence>
        </attribute>
        </attributes>
    </itemtype>
...
```

We will need this type and its attributes to be localized (the hMC will use these value for its prompts):

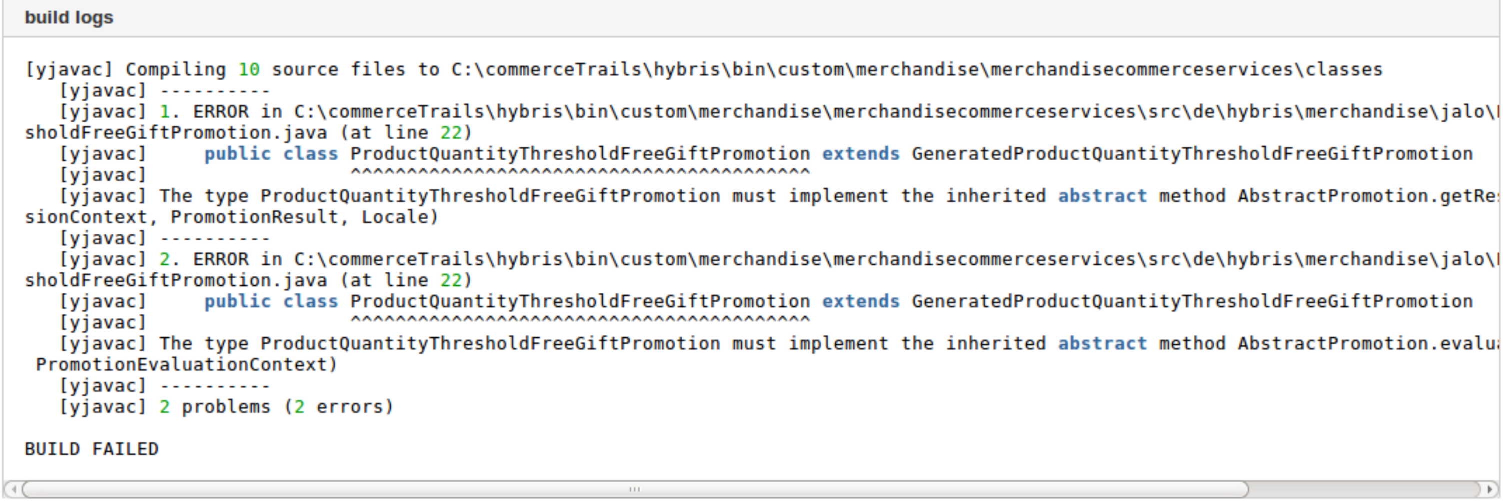
```
type.productquantitythresholdfreegiftpromotion.name=Buy x Items of Product A, get Product B as a Gift
type.productquantitythresholdfreegiftpromotion.giftProduct.name=Free Gift
type.productquantitythresholdfreegiftpromotion.qualifyingCount.name=Qualifying Count of Base Product
type.productquantitythresholdfreegiftpromotion.messageFired.name=Message for fired promotion
type.productquantitythresholdfreegiftpromotion.messageCouldHaveFired.name=Message if promotion could have fired
```

Generate a (JALO) class for the promotion logic

Not only does hybris generate servicelayer model classes to represent hybris types, but the ant compilation process also generates utility classes that serve as the underpinning of these model classes. These utility classes are referred to as **jalo layer** classes. The jalo layer (the name comes from **Ja**karta **Lo**gic) is normally invisible, and hybris developers are in fact discouraged from using it, at it is expected to be deprecated in an upcoming hybris release.

In spite of this, the promotion extension mechanism requires you to place new code in the jalo layer. We will do so, keeping in mind that this is an exception to our recommendation to avoid placing code in the jalo layer.

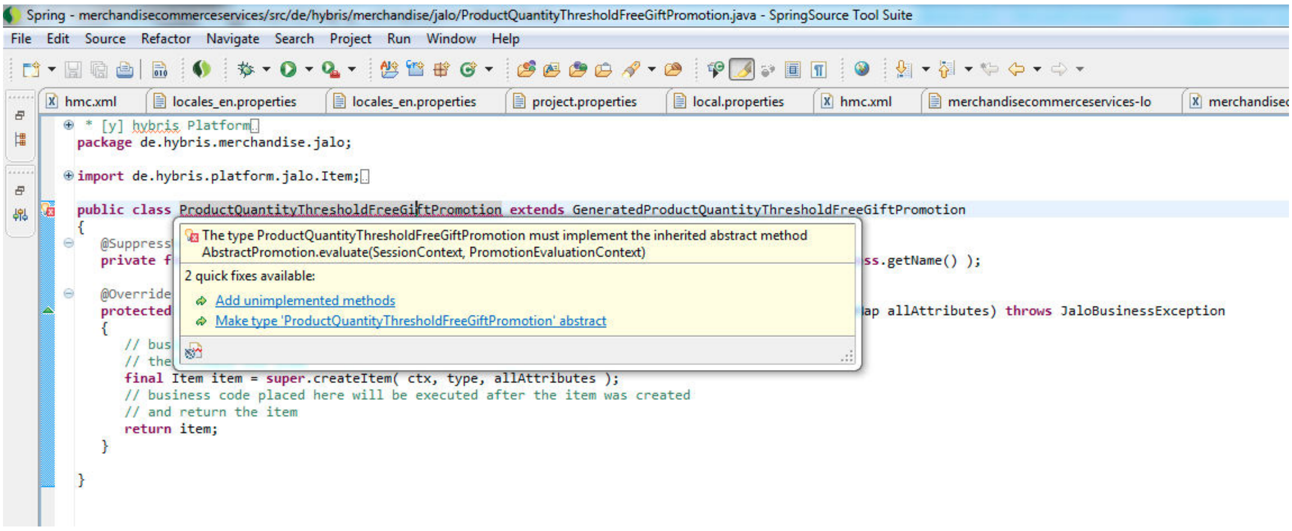
- Run **ant all** which generates the jalo class **ProductQuantityThresholdFreeGiftPromotion**. Notice that abstract methods in the parent class are not generated automatically (that is, after all, the point of abstract methods: the extending class must provide an implementation for them).



build logs

```
[yjavac] Compiling 10 source files to C:\commerceTrails\hybris\bin\custom\merchandise\merchandisecommerceservices\classes
   [yjavac] ----------
   [yjavac] 1. ERROR in C:\commerceTrails\hybris\bin\custom\merchandise\merchandisecommerceservices\src\de\hybris\merchandise\jalo\
sholdFreeGiftPromotion.java (at line 22)
   [yjavac]     public class ProductQuantityThresholdFreeGiftPromotion extends GeneratedProductQuantityThresholdFreeGiftPromotion
   [yjavac]                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   [yjavac] The type ProductQuantityThresholdFreeGiftPromotion must implement the inherited abstract method AbstractPromotion.getRe
sionContext, PromotionResult, Locale)
   [yjavac] ----------
   [yjavac] 2. ERROR in C:\commerceTrails\hybris\bin\custom\merchandise\merchandisecommerceservices\src\de\hybris\merchandise\jalo\
sholdFreeGiftPromotion.java (at line 22)
   [yjavac]     public class ProductQuantityThresholdFreeGiftPromotion extends GeneratedProductQuantityThresholdFreeGiftPromotion
   [yjavac]                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   [yjavac] The type ProductQuantityThresholdFreeGiftPromotion must implement the inherited abstract method AbstractPromotion.evalu
 PromotionEvaluationContext)
   [yjavac] ----------
   [yjavac] 2 problems (2 errors)

BUILD FAILED
```

Abstract methods inherited from parent classes need to be created manually. Refresh the merchandisecommerceservices extension and open the generated **ProductQuantityThresholdFreeGiftPromotion** class.



Spring - merchandisecommerceservices/src/de/hybris/merchandise/jalo/ProductQuantityThresholdFreeGiftPromotion.java - SpringSource Tool Suite

```
 * [y] hybris Platform
package de.hybris.merchandise.jalo;

import de.hybris.platform.jalo.Item;

public class ProductQuantityThresholdFreeGiftPromotion extends GeneratedProductQuantityThresholdFreeGiftPromotion
{
	@Suppress                                                                    ss.getName() );
	private f

	@Override
	protected                                                                   ap allAttributes) throws JaloBusinessException
	{
		// bus
		// the
		final Item item = super.createItem( ctx, type, allAttributes );
		// business code placed here will be executed after the item was created
		// and return the item
		return item;
	}

}
```

The type ProductQuantityThresholdFreeGiftPromotion must implement the inherited abstract method AbstractPromotion.evaluate(SessionContext, PromotionEvaluationContext)

2 quick fixes available:

- Add unimplemented methods
- Make type 'ProductQuantityThresholdFreeGiftPromotion' abstract

The only method generated automatically is **createItem**; since it merely calls super(), please feel free to remove it. Replace it with the following code, which provides an empty (for now) implementation for the two abstract methods of the **GeneratedProductQuantityThresholdFreeGiftPromotion** class, **evaluate()** and **getResultDescription()**:

merchandisecommerceservices/src/de/hybris/merchandise/jalo/ProductQuantityThresholdFreeGiftPromotion.java

...
import java.util.List;
import java.util.Locale;

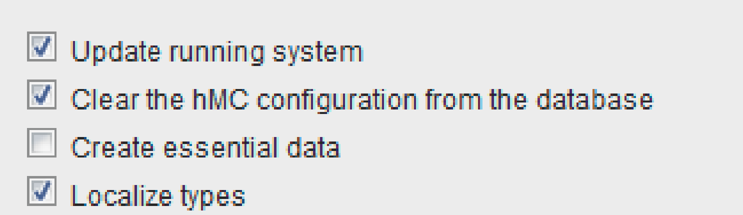
```
public class ProductQuantityThresholdFreeGiftPromotion extends
GeneratedProductQuantityThresholdFreeGiftPromotion
{
   @SuppressWarnings("unused")
   private final static Logger LOG =
Logger.getLogger(ProductQuantityThresholdFreeGiftPromotion.class.getName());

   @Override
   public List<PromotionResult> evaluate(final SessionContext ctx, final
PromotionEvaluationContext promoContext)
   {
      // YTODO Auto-generated method stub
      return null;
   }

   @Override
   public String getResultDescription(final SessionContext ctx, final
PromotionResult promotionResult, final Locale locale)
   {
      // YTODO Auto-generated method stub
      return null;
   }

}
```

Rebuild and Verify

Rebuild the code base by calling **ant all**. Then restart hybris server.

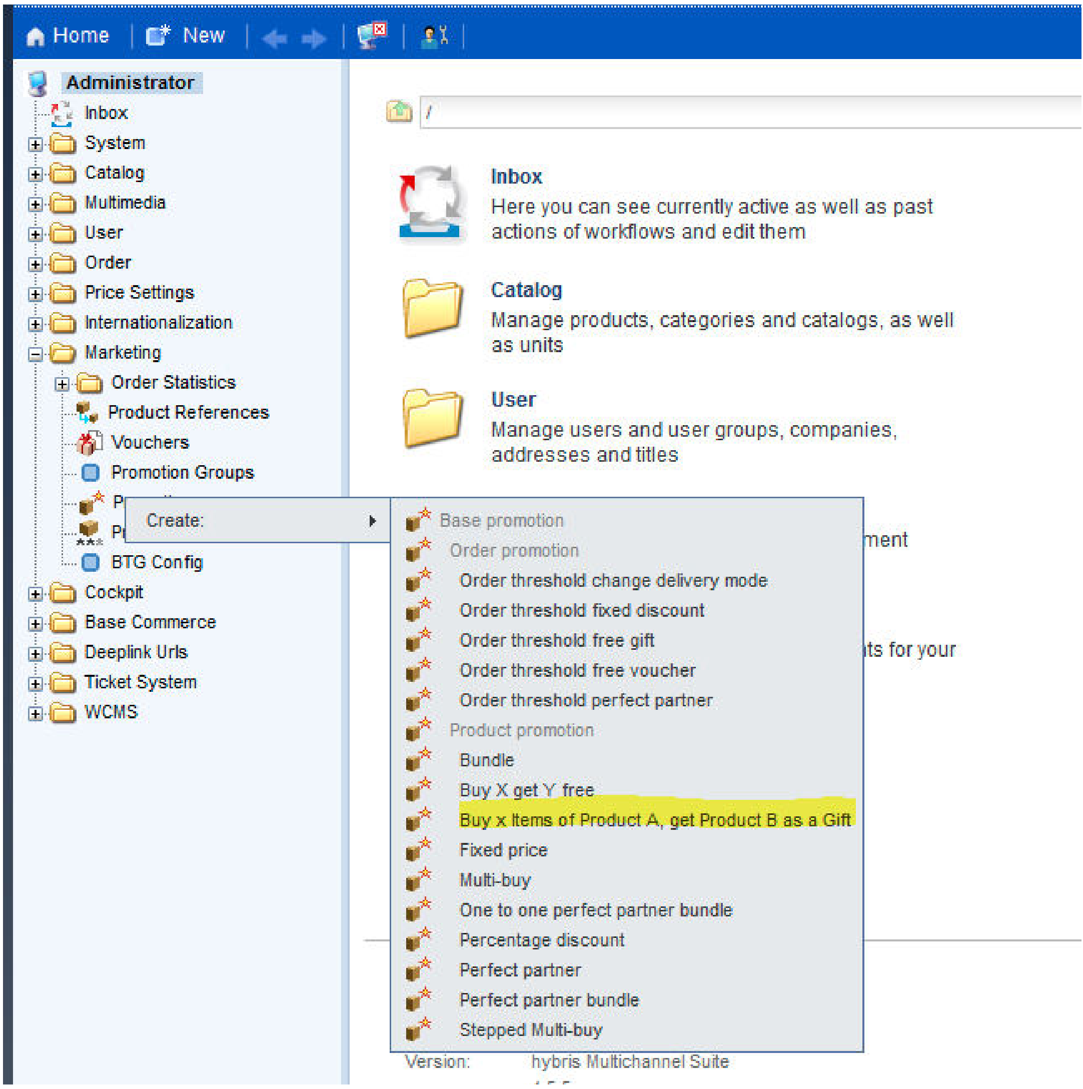
We need to perform a system update to:

- create deployment for our new promotion type
- localize our new type
- load our hmc configuration changes

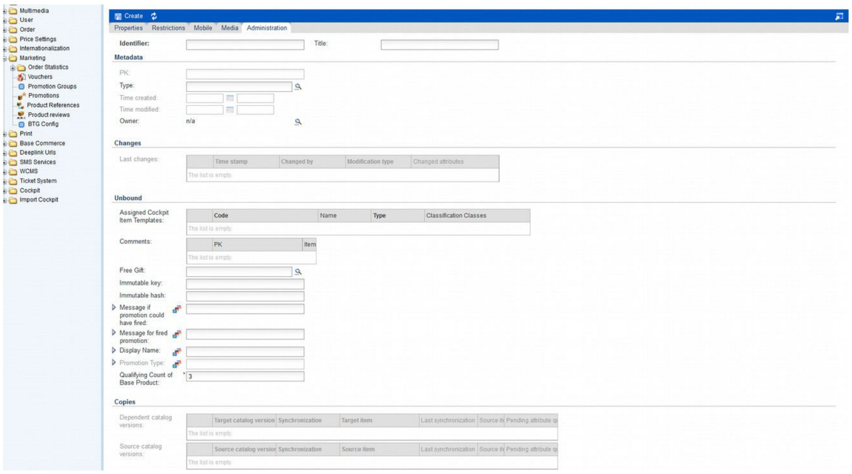


☑ Update running system
☑ Clear the hMC configuration from the database
□ Create essential data
☑ Localize types

After updating the system, you can go to the hmc and verify if you can create a new instance of our promotion type.


Home
New
Administrator
Inbox
System
Catalog
Multimedia
User
Order
Price Settings
Internationalization
Marketing
Order Statistics
Product References
Vouchers
Promotion Groups
BTG Config
Cockpit
Base Commerce
Deeplink Urls
Ticket System
WCMS
/
Inbox
Here you can see currently active as well as past actions of workflows and edit them
Catalog
Manage products, categories and catalogs, as well as units
User
Manage users and user groups, companies, addresses and titles
Create:
Base promotion
Order promotion
Order threshold change delivery mode
Order threshold fixed discount
Order threshold free gift
Order threshold free voucher
Order threshold perfect partner
Product promotion
Bundle
Buy X get Y free
Buy x Items of Product A, get Product B as a Gift
Fixed price
Multi-buy
One to one perfect partner bundle
Percentage discount
Perfect partner
Perfect partner bundle
Stepped Multi-buy
Version:
hybris Multichannel Suite

Notice that we created a required attribute *Qualifying Count of Base Product* on the Administration tab:



Identifier:
Title:

Metadata

PK:
Type:
Time created:
Time modified:
Owner: n/a

Changes

Last changes:

	Time stamp	Changed by	Modification type	Changed attributes
The list is empty.				

Unbound

Assigned Cockpit Item Templates:

	Code	Name	Type	Classification Classes
The list is empty.				

Comments:

	PK	Item
The list is empty.		

Free Gift:
Immutable key:
Immutable hash:
Message if promotion could have fired:
Message for fired promotion:
Display Name:
Promotion Type:
Qualifying Count of Base Product: * 3

Copies

Dependent catalog versions:

	Target catalog version	Synchronization	Target item	Last synchronization	Source it	Pending attribute q
The list is empty.						

Source catalog versions:

	Source catalog versior	Synchronization	Source item	Last synchronization	Source it	Pending attribute q
The list is empty.						

Implement the promotion's logic

We have previously supplied empty methods for our promotion class:

- **evaluate** - which evaluates the current cart, its products and current promotion evaluation context to respond with a collection of **PromotionResult**s
- **getResultDescription** - which composes promotion marketing messages on the frontend.

Promotion evaluation

In the **evaluate** method we have to:

- get the **base** product for the current promotion instance (A5 notepad)
- get the qualifying quantity of **base** product for the current promotion instance (5)
- check the quantity of the base product in the session cart
- add a **PromotionAction** to the promotion result if the real quantity is greater or equal to the qualifying quantity

- if the **base** product appears in the cart but the quantity does not qualify - add PromotionResult without PromotionAction (less than 1 , more that 0 - which will display the "couldHaveFired" marketing message to the customer.)

PromotionAction - is an executable entity that represents an action which is performed if user's cart qualifies for promotion.

The promotion extension comes with a set of promotion actions out-of-the-box. You may define your own PromotionAction type by creating a new type that extends *AbstractPromotionAction*.
However, in this trail, we will use an existing action: *PromotionOrderAddFreeGiftAction*

Promotion messages creation

In **getResultDescription**, we must resolve the parameter values for the parametrized **promotion fired** and **promotion could have fired** messages.

We will allow messages with 1 parameter, the qualifying quantity of the base product, in order that users know how many items need to added to the cart to get the promotion fired.

Note:
Please note that the parameters we add to the messages and their order need to be described in localized text entries (in the hMC):

- text.productquantitythresholdfreegiftpromotion.messagefiredargs
- text.productquantitythresholdfreegiftpromotion.messagecouldhavefiredargs

Provide the complete implementation for the ProductQuantityThresholdFreeGiftPromotion class:

ProductQuantityThresholdFreeGiftPromotion.java

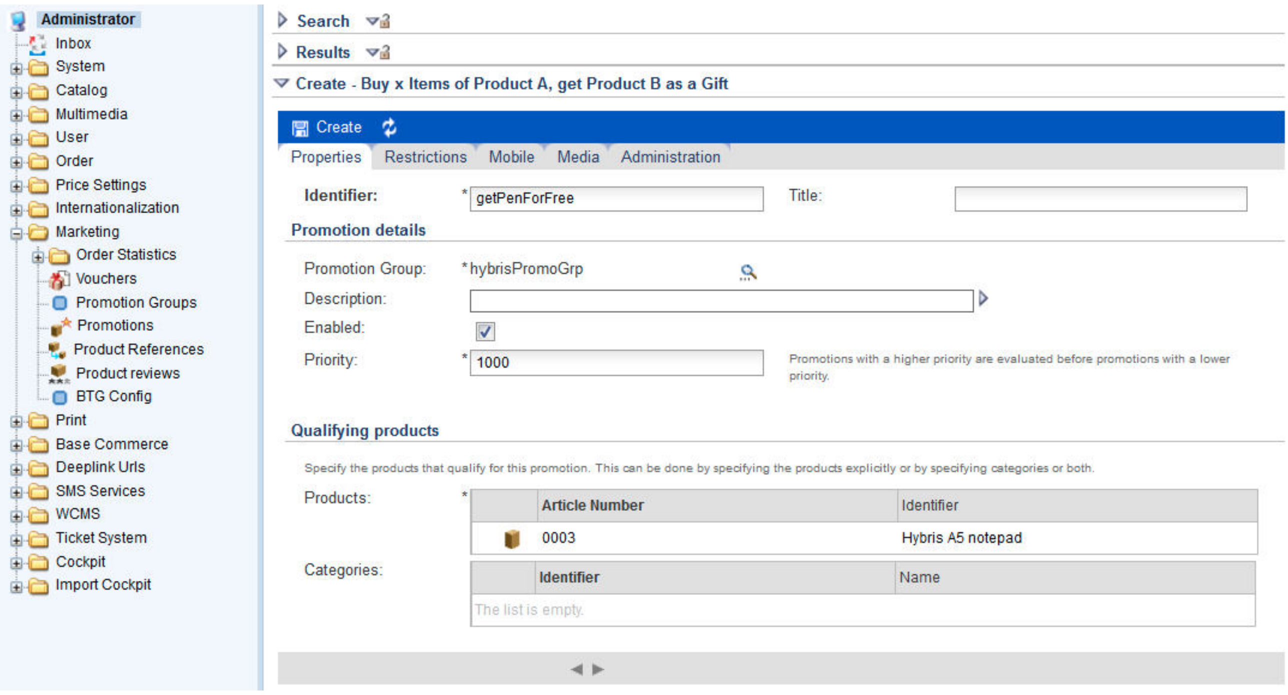
```java
/*
 *
 * [y] hybris Platform
 *
 * Copyright (c) 2000-2015 hybris AG
 * All rights reserved.
 *
 * This software is the confidential and proprietary information of hybris
 * ("Confidential Information"). You shall not disclose such Confidential
 * Information and shall use it only in accordance with the terms of the
 * license agreement you entered into with hybris.
 *
 */
package de.hybris.merchandise.jalo;

import de.hybris.platform.jalo.SessionContext;
import de.hybris.platform.jalo.product.~~Product~~;
import de.hybris.platform.promotions.jalo.PromotionResult;
import de.hybris.platform.promotions.jalo.PromotionsManager;
import de.hybris.platform.promotions.result.PromotionEvaluationContext;
import de.hybris.platform.promotions.result.PromotionOrderView;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import org.apache.log4j.Logger;


public class ProductQuantityThresholdFreeGiftPromotion extends GeneratedProductQuantityThresholdFreeGiftPromotion
{
    private final static Logger LOG = Logger.getLogger(ProductQuantityThresholdFreeGiftPromotion.class.getName());


    @Override
    public List<PromotionResult> evaluate(final SessionContext ctx, final PromotionEvaluationContext promoContext)
    {
        if (getProducts() != null && !getProducts().isEmpty())
        {
            // the base product is the first of related products
            final ~~Product~~ baseProduct = getProducts().iterator().next();

            final List<PromotionResult> results = new ArrayList<PromotionResult>();

            // Find the eligible products, and apply any restrictions
            final PromotionsManager.RestrictionSetResult restrictResult = findEligibleProductsInBasket(ctx, promoContext);

            // If the restrictions did not reject this promotion, and there are still products allowed after the restrictions
            if (restrictResult.isAllowedToContinue() && !restrictResult.getAllowedProducts().isEmpty())
            {
                final int qualifyingCount = this.getQualifyingCount(ctx).intValue();

                // Create a view of the order containing only the allowed products
                final PromotionOrderView orderView = promoContext.createView(ctx, this, restrictResult.getAllowedProducts());

                // Get the real quantity of the base product in the cart
                final long realQuantity = orderView.getQuantity(ctx, baseProduct);

                if (realQuantity >= qualifyingCount)
                {
                    if (LOG.isDebugEnabled())
                    {
                        LOG.debug("(" + getPK() + ") evaluate: product quantity " + realQuantity + ">" + qualifyingCount
                                    + ".  Creating a free gift action.");
                    }
                    final PromotionResult result = PromotionsManager.getInstance().createPromotionResult(ctx, this,
                            promoContext.getOrder(), 1.0F);

                    //Apply free gift promotion action
                    final ~~Product~~ product = this.getGiftProduct(ctx);
                    result.addAction(ctx,
PromotionsManager.getInstance().createPromotionOrderAddFreeGiftAction(ctx, product, result));

                    results.add(result);
                }
                else
                {
                    if (LOG.isDebugEnabled())
                    {
                        LOG.debug("(" + getPK() + ") evaluate: product quantity " +
realQuantity + ">" + qualifyingCount
                                + ".  Creating a free gift action.");
                    }
                    final float certainty = (realQuantity / qualifyingCount);
                    final PromotionResult result =
PromotionsManager.getInstance().createPromotionResult(ctx, this,
                            promoContext.getOrder(), certainty);
                    results.add(result);
                }
                return results;
            }
        }
        return null;
    }


    @Override
    public String getResultDescription(final SessionContext ctx, final PromotionResult promotionResult, final
Locale locale)
    {
        final Object[] args =
        { this.getQualifyingCount() };
        if (promotionResult.getFired(ctx))
        {
            return formatMessage(this.getMessageFired(ctx), args, locale);
        }
        else if (promotionResult.getCouldFire(ctx))
        {
            return formatMessage(this.getMessageCouldHaveFired(ctx), args, locale);
        }
        return "";
    }

}
```

Rebuild and Verify

Rebuild the code base and restart the server.

Create an instance of our promotion type in the hmc:

- Identifier: **GetPensForFree**
- Promotion Group: **hybrisPromoGrp**
- Enabled: **yes**
- Priority: **1000**
- Products: **Hybris A5 Notepad** (from the Online Product Catalog)



Create - Buy x Items of Product A, get Product B as a Gift

Identifier: * getPenForFree Title:

Promotion details

Promotion Group: *hybrisPromoGrp

Description:

Enabled: ☑

Priority: * 1000

Promotions with a higher priority are evaluated before promotions with a lower priority.

Qualifying products

Specify the products that qualify for this promotion. This can be done by specifying the products explicitly or by specifying categories or both.

Products: *

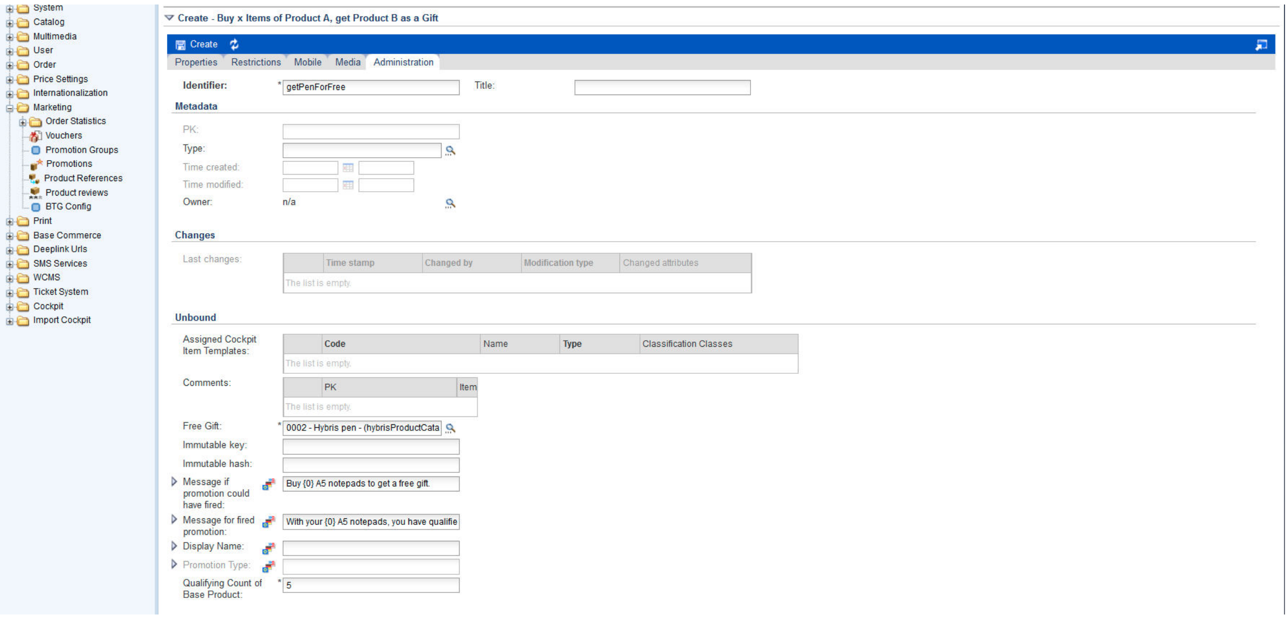
Article Number	Identifier
0003	Hybris A5 notepad

Categories:

Identifier	Name
The list is empty.	

And in the *Administration* tab:

- Free Gift: **hybris Pen** (from the Online Product Catalog)
- Message if promotion could have fired: **Buy {0} A5 Notepads to get a free gift**
- Message for fired promotion: **With your {0} A5 Notepads, you have qualified for a free pen.**



Create - Buy x Items of Product A, get Product B as a Gift

Identifier: * getPenForFree   Title:

Metadata

PK:

Type:

Time created:

Time modified:

Owner: n/a

Changes

Last changes:

	Time stamp	Changed by	Modification type	Changed attributes
The list is empty.				

Unbound

Assigned Cockpit Item Templates:

	Code	Name	Type	Classification Classes
The list is empty.				

Comments:

	PK	Item
The list is empty.		

Free Gift: * 0002 - Hybris pen - (hybrisProductCata

Immutable key:

Immutable hash:

Message if promotion could have fired: Buy {0} A5 notepads to get a free gift.

Message for fired promotion: With your {0} A5 notepads, you have qualifie

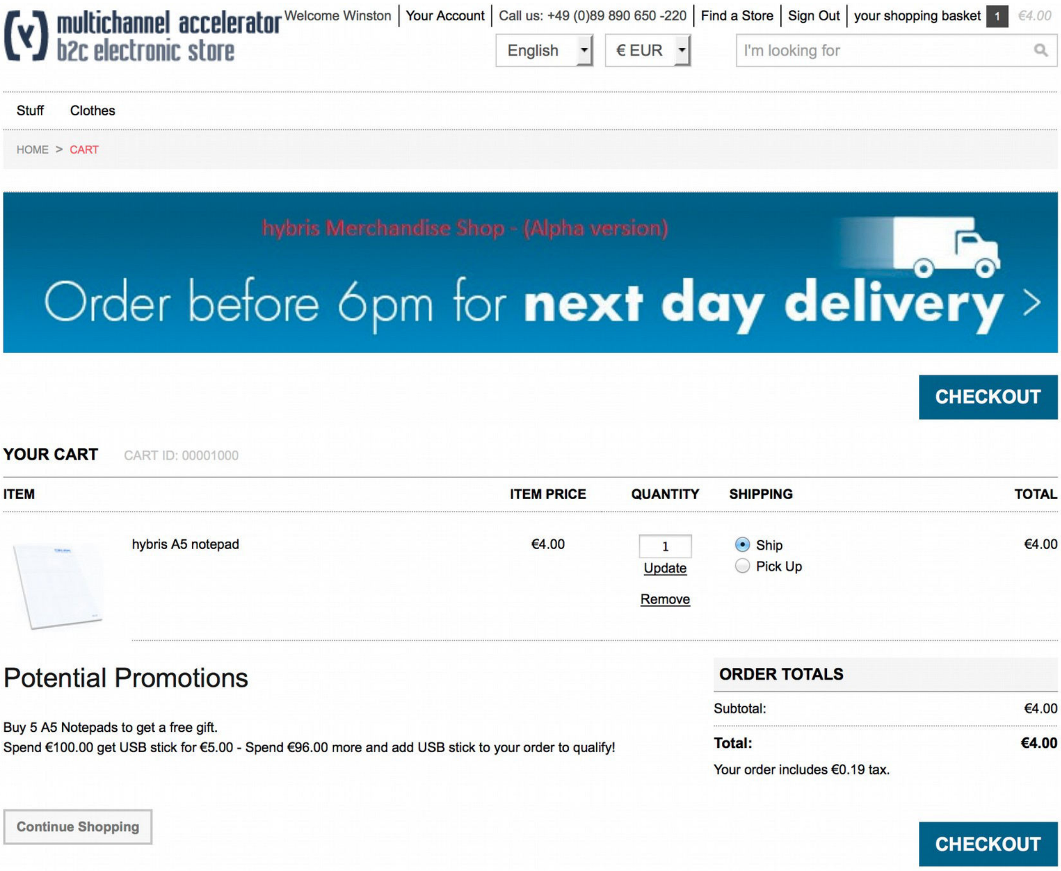
Display Name:

Promotion Type:

Qualifying Count of Base Product: * 5

Go to the store front end and compose a

- "promotion could have fired" cart (with a quantity of less than 5 *hybris A5 Notepads*)



YOUR CART   CART ID: 00001000

ITEM	ITEM PRICE	QUANTITY	SHIPPING	TOTAL
hybris A5 notepad	€4.00	1	● Ship ○ Pick Up	€4.00

Potential Promotions

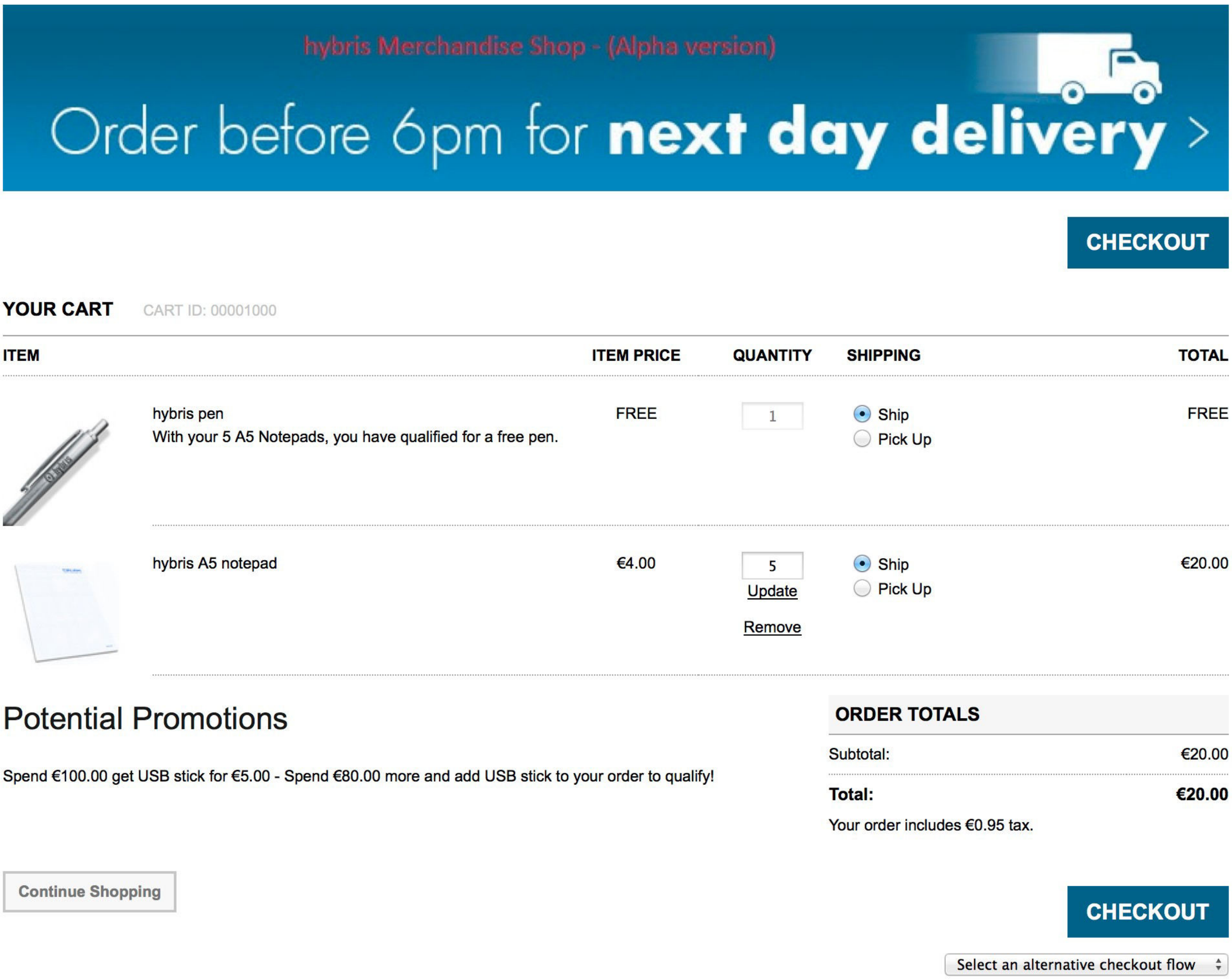
Buy 5 A5 Notepads to get a free gift.
Spend €100.00 get USB stick for €5.00 - Spend €96.00 more and add USB stick to your order to qualify!

ORDER TOTALS

Subtotal:	€4.00
Total:	**€4.00**

Your order includes €0.19 tax.

"promotion fired" cart (with at least 5 *hybris A5 Notepads*)


hybris Merchandise Shop - (Alpha version)
Order before 6pm for next day delivery >
CHECKOUT

YOUR CART CART ID: 00001000

ITEM		ITEM PRICE	QUANTITY	SHIPPING	TOTAL
	hybris pen With your 5 A5 Notepads, you have qualified for a free pen.	FREE	1	● Ship ○ Pick Up	FREE
	hybris A5 notepad	€4.00	5 Update Remove	● Ship ○ Pick Up	€20.00

Potential Promotions

Spend €100.00 get USB stick for €5.00 - Spend €80.00 more and add USB stick to your order to qualify!

ORDER TOTALS	
Subtotal:	€20.00
Total:	**€20.00**

Your order includes €0.95 tax.