

# **Personalizing Content of the Shop**

In this trail you will learn how to use advanced personalization, as well as creating and using customized personalization conditions.

## **Advanced Personalization in Brief**

Advanced Personalization Module may provide benefits both to the end customers and to the hybris customers. Find out more about how you can profit of using hybris AP(Advanced Personalization ) Module.

### **End Customer Benefits**

Advanced Personalization is a pro-active approach toward these consumers who want a quick and informative shopping experience meeting their exact requirements

Advanced Personalization enables your customers to find what they are looking for and helps them discover products and solutions they may not necessarily have considered in the first place.

Advanced Personalization Module supports you with providing personalized on-line shopping experience to your customers. By using personalization, you can measure and reward the on-line behavior of your customers

- Determine a customers on-line history and behavior
- Organize your site and navigation around your customers needs
- Individualize layout and site content for each and every customer
- Generate targeted promotions

### **hybris Customer Benefits**

- Know your customers behavior
- Know your customers preferences
- Guide your customers seamlessly through your website page flow
- Adapt the content presented to your customer depending on the customer choices
- Target the needs of the particular group of selected customer
- Offer your customers customized content, product, service
- Increase your revenue by providing customers with exactly these product, services and content they may be interested in
- Increase your revenue by providing customers with shopping suggestions based on the similar behavior of other customers

- Customize your approach to keep your customers on your site
- Recognize returning customers

## **Customer Segmentation**

Advanced Personalization Module provides an easy to use tool to create the containers called Customer Segments

These segments can be viewed as the simple and efficient means to differentiate your customers into several individualized target groups

For each segment you may create rules and the actions that are triggered after the customers meet all rules conditions.

**Customer Segmentation have two types of actions :**

**1.Customer Segment Rules**

**2.Output Actions**

### **Customer Segment Rules:**

Customer Segment Rules: Create one rule or create several rules that would contain specific conditions for the customers in order to classify them into personalized groups

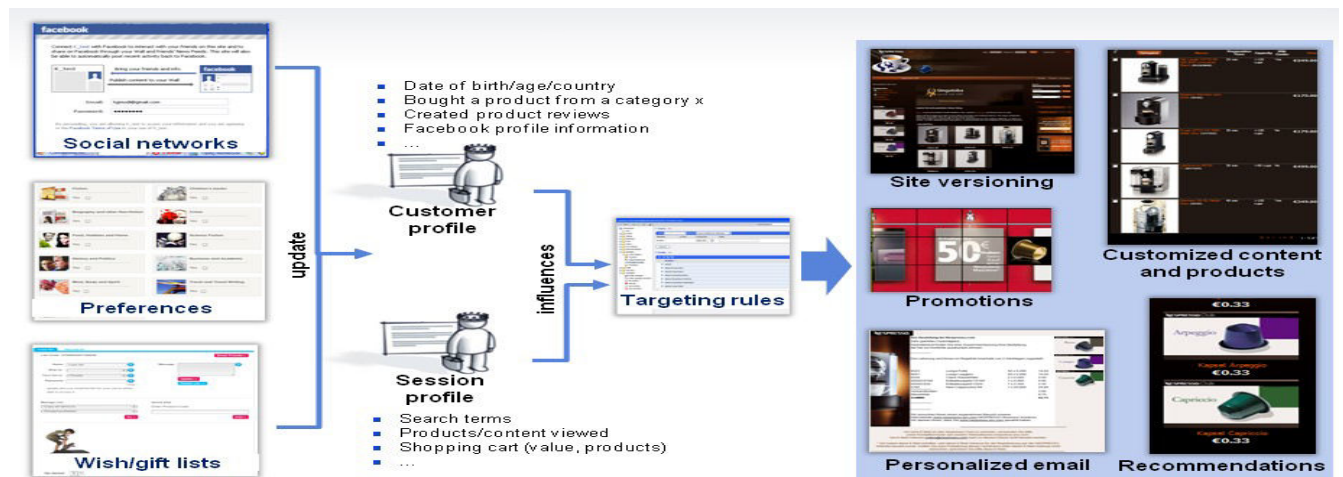
**Output Actions:** Action triggered depending on the success of the segment rules fulfillment

### **Customer Oriented Personalization:**

1. Customers are assigned to segments dynamically based on their online behavior

2. Hybris Advanced Personalization enables you to collect the relevant information from the customer profile or session profile and compare it to the targeting rules

- Products viewed
- Searches performed
- Shopping cart contents
- Date of birth
- Gender
- Region



**AIM:** Use the Advanced Personalization Module to show a merchandising banner on a checkout page for non-hybrid customers (i.e. where the `isInternal` flag evaluates to false).

For all hybrid customers, the banner should not be visible. (A Hybrid is an individual of the species *homo sapiens* who also works for hybris.)

### Implementation:

- We need to compose a valid rule for our story, consisting of a single condition
- For the purpose of this trail, we will create a new operand type that provides the value of the **isInternal** flag.
- This flag determines whether a customer is a 'hybrid' customer or not.
- As an output action that handles CMS component restrictions already exists, we will use it for our story.
- Additionally, we need to create a CMS component for the merchandise banner that will be displayed/hidden depending on the outcome of the advanced personalization evaluation.

**Let's create a banner WCMS component that will display merchandise information.**

**Create a HybridBanner Component:**

Add the following insert statements at the end of the script

merchandiseinitialdata/resources/merchandiseinitialdata/import/sampledatalogs/hybrisContentCatalog/cms-content.impex:

```
##### Hybrids BTG Banner
INSERT_UPDATE SimpleBannerComponent;
$contentCV[unique=true];uid[unique=true];name;&componentRef;urlLink
;;HybridsBanner;Hybrids Banner;HybridsBanner;
```

**Create a content slot for the new banner**

merchandiseinitialdata/resources/merchandiseinitialdata/import/sampledatalogs/hybrisContentCatalog/cms-content.impex

```
INSERT_UPDATE ContentSlot;
$contentCV[unique=true];uid[unique=true];name;active;cmsComponents(&componentRef)
;;SideContent-CheckoutSummary;Side Slot for checkoutSummaryPages;true;HybridsBanner
```

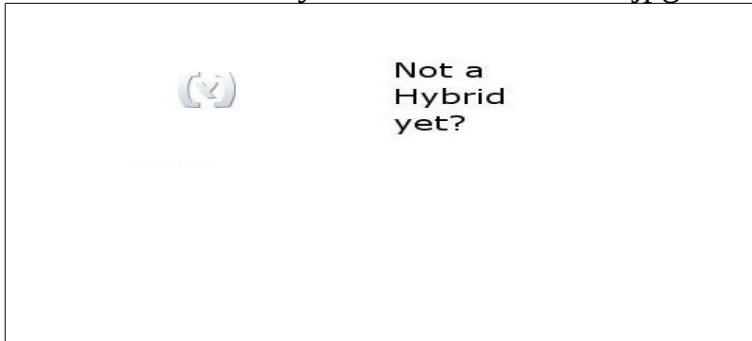
**Assign content slot to page:**

```
INSERT_UPDATE ContentSlotForPage;
$contentCV[unique=true];uid[unique=true];position[unique=true];page(uid,$contentCV)[unique=true]
[default='multiStepCheckoutSummaryPage'];contentSlot(uid,$contentCV)[unique=true]
;;Side-CheckoutSummary;SideContent;;SideContent-CheckoutSummary
```

The banner needs media (image), and to be localized, since the image may contain text. You will find banners in the following

**location:**merchandiseinitialdata/resources/merchandiseinitialdata/import/sampledatalogs/hybrisContentCatalog/images/banners/site/

Merch\_450x290\_HybridsBanner\_EN\_02.jpg



- **Create a Media instance using the English version of the jpg file. Add the following to the end of the script:**

merchandiseinitialdata/resources/merchandiseinitialdata/import/sampledatalog/hybrisContentCatalog/cms-content\_en.impex

```
INSERT_UPDATE Media;  
$contentCV[unique=true];code[unique=true];@media[translator=de.hybris.platform.im  
pex.jalo.media.MediaDataTranslator];mime[default='image/jpg'];altText  
;;Merch_450x290_HybridsBanner_EN_02.jpg;  
$siteResource/images/banners/site/Merch_450x290_HybridsBanner_EN_02.jpg;;"Not  
a hybrid yet?"
```

**Then assign the media instance to the banner by adding the following lines:**

merchandiseinitialdata/resources/merchandiseinitialdata/import/sampledatalog/hybrisContentCatalog/cms-content\_en.impex

```
UPDATE SimpleBannerComponent;$contentCV[unique=true];uid[unique=true];  
$picture[lang=en]
```

```
;;HybridsBanner;Merch_450x290_HybridsBanner_EN_02.jpg
```

Go to the [update page](#) and run the update for project data from merchandiseinitialdata extension (select only **Import Sample Data**)

This will import our banner and synchronize it to the online content catalog version. After this operation your checkout page should look like this:

The screenshot displays the checkout process for 'multichannel accelerator b2c electronic store'. The 'Delivery Address' tab is active, showing a form for address details. The 'ORDER TOTALS' section indicates a subtotal and total of €5.50. The 'ITEMS TO BE DELIVERED' section lists a 'hybris Mug' for €5.50. A large banner at the bottom reads 'Not a Hybrid yet?' with a Hybris logo.

## NOTE: Lets create Dynamic attribute

Before going to this trail Lets create Dynamic attribute “is InternalFlag” attribute in customer model which gives” true” or false ,if customer Login with [\\*\\*\\*\\*@hybris.com](#) or [\\*\\*\\*\\*@hybris.de](#) it will returns true other wise false.

**if u did thin in intial trails(Compose and Model the Merchandise Shop trail)**

Please ignore this bellow steps up to:**Step3**

## STEP:1

```
<typegroup name="merchandise">
<itemtype code="Customer" generate="false" autocreate="false">
  <attributes>
    <attribute qualifier="isInternal" type="java.lang.Boolean">
      <description>Defines if the customer is a hybris internal employee.
Aggregated information</description>
      <persistence type="dynamic">
```

```
attributeHandler="dynamicHybrisCustomerAttributeBean" />
    <modifiers read="true" write="true" optional="true"
        unique="false" />
    </attribute>
</typegroup>
```

## STEP:2

### Adding Code for Dynamic Attribute:

**merchandisecore/src/de/hybris/merchandise/core/model/DynamicHybrisCustomerAttributeBean.java**

```
/**
 *
 */
package de.hybris.merchandise.core.model;

import de.hybris.platform.core.model.user.CustomerModel;
import de.hybris.platform.servicelayer.model.attribute.DynamicAttributeHandler;

/**
 *
 */
public class DynamicHybrisCustomerAttributeBean implements
DynamicAttributeHandler<Boolean, CustomerModel>
{

    @Override
    public Boolean get(final CustomerModel model)
    {
        if (model == null)
        {
            throw new IllegalArgumentException("Item model is required");
        }

        // Accelerator stores the email in the ID (uid) field
        final String email = model.getUid();
        return Boolean.valueOf(email != null && (email.endsWith("hybris.de") ||
```

```
email.endsWith("hybris.com"))));

    }

    @Override
    public void set(final CustomerModel model, final Boolean value)
    {
        //throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

### configure the bean:

#### **merchandisecore/resources/merchandisecore-spring.xml**

```
<bean id="dynamicHybrisCustomerAttributeBean"
class="de.hybris.merchandise.core.model.DynamicHybrisCustomerAttributeBean"/>
```

### STEP:3Localizing the new attributes:

#### **merchandisecore/resources/localization/merchandisecore-locale\_en.properties**

```
type.Customer.isInternal.name=Internal
```

### Testing the extension:

- run ant all (refresh Eclipse workspace if necessary)
  - as we have modified the data model, we need to update the system. Do the update from the hAC and include the type system localization.
1. Update running system
  2. Localize types

#### Note:

Here We have created dynamic attribute isInternal ,it returns true when customer Enter with @hybris.com extension



## **Creating a new Operand Type:**

### **Add Item Type for the Operand**

As mentioned earlier, we need to add a new evaluation operand for our story. We need a marker subtype of BTGAbstractCustomerOperand. Please create one in the **merchandisecore-items.xml**

```
<typegroup name="BTG">
<itemtype code="BTGCustomerInternalFlagOperand"
extends="BTGAbstractCustomerOperand" autocreate="true" generate="true"
jaloclass="de.hybris.merchandise.core.jalo.BTGCustomerInternalFlagOperand">
</itemtype>
</typegroup>
```

As we are extending a BTG extensions type, we need to set the dependencies in **extensioninfo.xml** and in the **Eclipse**.

### **merchandisecore/extensioninfo.xml**

```
<requires-extension name="btg" />
```

### **Every new type needs to be localized:**

Please add localizations for all desired languages, for ]

instance:/**merchandisecore/resources/localization/merchandisecore-  
locales\_en.properties**

### **merchandisecore/resources/localization/merchandisecore-locales\_en.properties**

```
### Localization for type BTGCustomerInternalFlagOperand

type.btgcustomerinternalflagoperand.name=Customer 'internal' flag operand
type.btgcustomerinternalflagoperand.description=Customer 'internal' flag operand
```

We now run a build of the code base (and restart the server), so the equivalent models in the service layer are generated. (**ant all**).

### **Implement and register the operand value provider:**

**OperandValueProvider** is a generic interface to be implemented for a given operand type. Its implementation determines how to fetch the actual operand's value from the target object (User) as well as the type (class) of this value. This allows matching a proper expression evaluator.

We need to implement an **OperandValueProvider** for our new operand type :

**BTGCustomerInternalFlagOperand.**

inside the merchandisecore extension

**merchandisecore/src/de/hybris/merchandise/core/btg/condition/operand/valueprovider/CustomerInternalFlagValueProvider.java**

```
package de.hybris.platform.btg.condition.operand.valueprovider;

import de.hybris.platform.btg.condition.operand.OperandValueProvider;
import de.hybris.platform.btg.enums.BTGConditionEvaluationScope;
import de.hybris.platform.btg.jalo.BTGException;
import de.hybris.platform.core.model.user.CustomerModel;
import de.hybris.platform.core.model.user.UserModel;

/**
 * @author rehaman
 */
public class CustomerInternalFlagValueProvider implements
OperandValueProvider<BTGCustomerInternalFlagOperandModel>
{

    @Override
    public Object getValue(final BTGCustomerInternalFlagOperandModel operand, final
UserModel user,
        final BTGConditionEvaluationScope scope)
    {
        if (user instanceof CustomerModel)
        {
            return ((CustomerModel) user).getIsInternal();
        }
        throw new BTGException("user must be of type [Customer]");
    }

    @Override
```

```

public Class getValueType(final BTGCustomerInternalFlagOperandModel arg0)
{
    return Boolean.class;
}
}

```

registered in the spring context (**merchandisecore/resources/merchandisecore-spring.xml**):

```

<bean id="customerInternalFlagValueProvider"
class="de.hybris.merchandise.core.btg.condition.operand.valueprovider.CustomerIntern
alFlagValueProvider" />

```

### Map the new Operand to the User Rule Types.

Adjust/Create the file with following

content:**merchandiseinitialdata/resources/merchandiseinitialdata/import/sampledatab/stores/hybris/btg.impex**- which will be fired while in theInitialDataSystemSetup- with the following content:

merchandiseinitialdata/resources/merchandiseinitialdata/import/sampledatab/stores/hybris/btg.impex

```

INSERT_UPDATE
BTGConfig[unique=true];defaultTimeUnit(code);usedRuleTypes(code);operandMapping(key(code),value(code))[map-delimiter=|]

BTGConfig;WEEK;ORDER,CART,USER,WCMS;ORDER-
>BTGCategoriesInOrdersOperand,BTGEachOrderTotalSumOperand,BTGOrderTotalSumOperand,BTGProductsInOrdersOperand,BTGNumberOfOrdersOperand,BTGNumberOfOrdersRelativeDateOperand,BTGLastOrderDateOperand|CART-
>BTGCartIsEmptyOperand,BTGCartTotalSumOperand,BTGCategoriesInCartOperand,BTGProductsInCartOperand,BTGQuantityOfProductInCartOperand|WCMS-
>BTGViewedProductsOperand,BTGViewedCategoriesOperand,BTGVisitedContentpagesOperand,BTGReferralUrlOperand,BTGUrlParameterOperand|USER-
>BTGUserAddressPostalCodeOperand,BTGUserCountryOperand,BTGUserGenderOperand,BTGUserMemberOfGroupsOperand,BTGCustomerInternalFlagOperand|SCRIPT->BTGMediaScriptOperand,BTGStringScriptOperand

```

Rebuild the code base (**ant all**) to apply the changes.

A system update is also necessary to initialize and localize the new type in the database.

You may perform the system update from the[update page](#)

Select:

- Update running system (to enable new operand type in the type system)

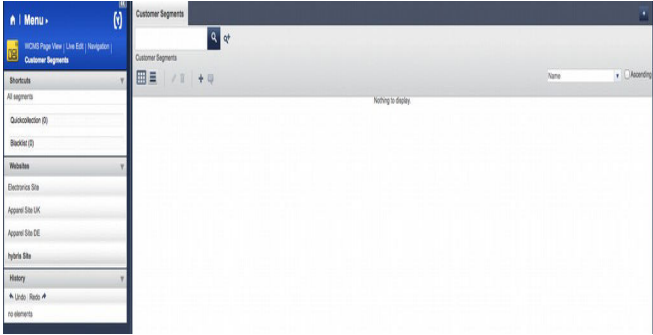
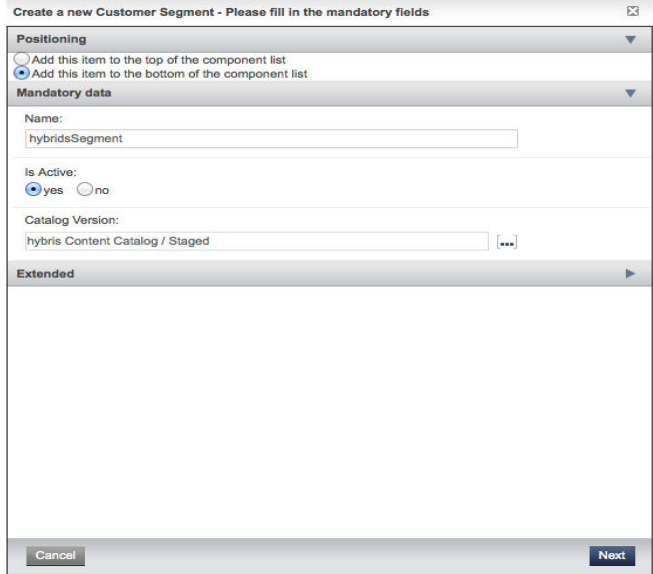
- Localize types (to localize new operand type)
- Project data from merchandiseinitialdata extension (select only **Import Sample Data**)

After the update, you can use your customized personalization rule in the csmcockpit.

## How to apply the personalization to the site:

Let's now apply personalization criteria to our merchandise store. Here is a tutorial that demonstrates how to create concrete instances of BTG segments, rules, conditions and eventually output actions.

First, log in to the [cmscockpit](#) and view the Customer Segments perspective. Select the **hybris Site** in the list of Websites, on the left.

Screen shot	Description
	<p>An empty CustomerSegments perspective. Click the small plus (+) icon to add a segment</p>
	<p>Name your segment, set it to <b>active</b> and assign it the hybris store's <b>staged content catalog</b>.</p>

Create a new Customer Segment - Assigned Websites

Please choose on which Websites this customer segment should be used:

- ☐ Electronics Site
- ☐ Apparel Site UK
- ☐ Apparel Site DE
- ☒ hybris Site

Cancel Back Next

Assign it to our hybris CMSSite

Create a new Customer Segment - (Result Scope)

Please choose the period of time in which the users have to fulfill the rules of this segment.

Session Scope

Persistent Scope

Cancel Back Next

Set the Result evaluation to **Persistent**

Create a new Customer Segment - (Result Scope)

Please choose the period of time in which the users have to fulfill the rules of this segment.

Session Scope

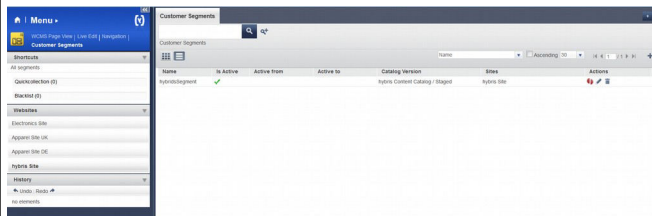
Persistent Scope

Cancel Back Next

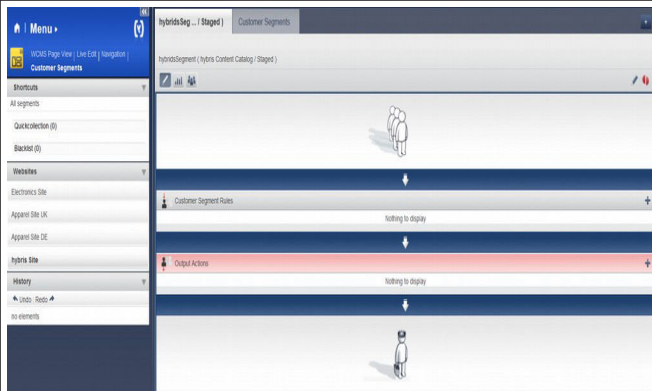
Set the Result evaluation to **Persistent**



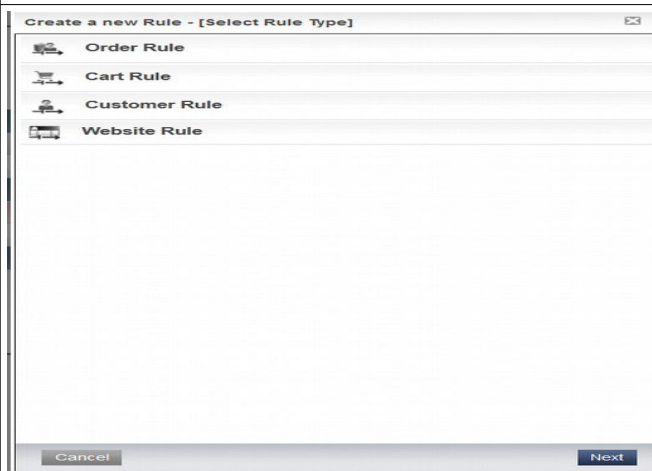
Set the evaluation mode to **OPTIMIZED**



Have a look at the new segment you have created



Double-click on the segment to enter the segment editor's view : Note the two areas for rules and output actions. Click the small plus (+) icon in the Customer Segment Rules area to add a new rule.



Choose a Customer Rule type.

Create a new Rule - [Create new rule]

--- choose left operand type ---

- Postal code of users address
- Country user lives in
- Gender of user
- Member of Usergroups
- Customer 'internal' flag operand

--- choose operator ---

--- choose right ---

Cancel Back Next

As we have updated the mapping, you should be able to choose our new Operand type

Create a new Rule - [Create new rule]

Customer 'internal' flag operand

Customer 'internal' flag operand

==

Boolean

Value

☒ yes ☐ no

Cancel Back Next

Refine the Right operand to true (labelled here as **yes**). You can set an inverse rule just by simply selecting **no** (false).

Done

You are then asked about where to position the rule. As it's the only rule, click on **DONE**.

Menu •

WCMS Page View | Live Edit | Navigation

Customer Segments

hybrisSeg - / Staged | Customer Segments

hybrisSegment (hybris Content Catalog) | Staged

Shortcuts

- All segments
- Quickcollection (0)
- Blacklist (0)

Websites

- Electronics Site
- Apparel Site UK
- Apparel Site DE

Hybris Site

History

- Undo Redo
- No elements

Customer Segment Rules

Customer 'internal' flag operand is equal true


Output Actions


Nothing to display

Note that there is a new rule in our segment. Click on the plus (+) icon to create an output action

**Create Output Action - Choose Type**

Do you want to show or hide a WCMS Component or a WCMS Page?

 **WCMS Page**

 **WCMS Component**

Our action focuses on components only, not on the whole page

**Create Output Action**

Please choose the WCMS Component you want to show or hide. If you are not sure about the name of the WCMS Component you can first select a WCMS Page and afterwards you can select the component from a list of all components of this page.

WCMS Page that contains your WCMS Component (optional):

WCMS Component:

Find our banner component :  
HybridsBanner

**Create Output Action**

Please choose if users which have fulfilled this segment, or only users which have not fulfilled this segment should see this WCMS Component.

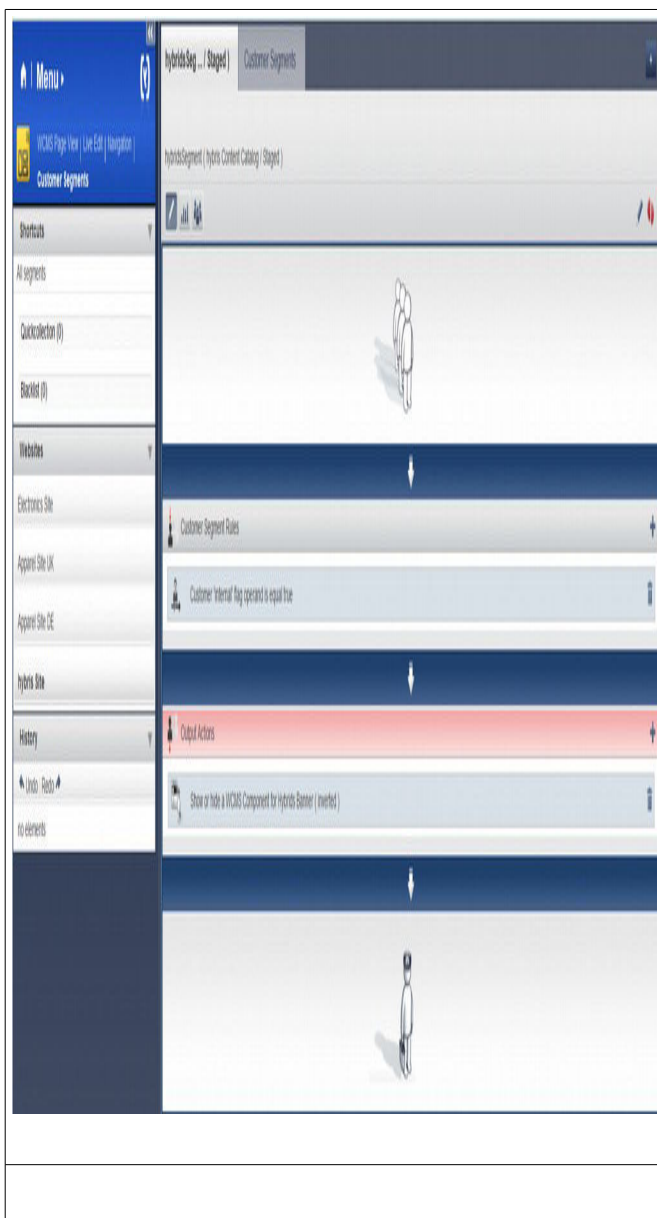
**Users which have fulfilled this segment**

☐ should see this WCMS Component  
☒ should **not** see this WCMS Component

Define if the banner should be visible or not for users who fulfill the segment's rules : in our case we want to hide it

Review the whole segment with its rule and output action. Please note the small **synchronization** button in the right corner. It's red because we haven't yet synchronized it with the **online** catalog





version. Only after we do so will it be usable in our storefront

For more information go through this link

[https://wiki.hybris.com/pages/viewpage.action?pageId=294094330&preview=/252609521/251534835/btg\\_16\\_v50.png](https://wiki.hybris.com/pages/viewpage.action?pageId=294094330&preview=/252609521/251534835/btg_16_v50.png)