BURBERRY STORE

Technical Design documentation

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Updated By | Date | Comments |
| 1.0 | Chaitanya  Chandan  Ankita  Keerthi | 22-06-2016 |  |
|  |  |  |  |
|  |  |  |  |

Document Sign-Off

|  |  |  |
| --- | --- | --- |
| Name | Role | Signature/Date |
| Elangovan | Manager |  |
| Hari | TechLead |  |
|  |  |  |

Table of Contents

[1. Installation Guide 3](#_Toc416258348)

[1.1 Assumptions 3](#_Toc416258349)

[1.2 Document Scope 3](#_Toc416258350)

[1.3 Functional Rules 3](#_Toc416258351)

[1.4 Glossary of Terms 3](#_Toc416258352)

[1.5 Additional Resources 3](#_Toc416258353)

[1.6 Steps to set up hybris 4](#_Toc416258354)

[2. Customizing Burberry Store 5](#_Toc416258356)

2.0 Application Related Sample Diagrams

[2.1 Customizing initialization process 5](#_Toc416258357)

[2.2 Rename project folders 7](#_Toc416258358)

[2.3 Basic Store Data. 8](#_Toc416258359)

[2.4 Creating Products and Variants 10](#_Toc416258360)

2.5 Solr configuration set up............................................................................................10

2.6 Site CMS Content with custom shipping method.....................................................10

2.7 Custom Promotions.................................................................................................10

2.8 Vouchers..................................................................................................................10

**3. Implementation of products addons…………………............................……………………27**

**4. Implementation of coming soon products………………………..................………………30**

**5. Installing Order Management Service(OMS)……………………………...............…………33**

**6. PayPal Integeration……………………………………………………………………................35**

**INTRODUCTION**

The Hybris Commerce Suite is a very lightweight and completely Java-based application that runs on many system combinations. For a quick start, it is important to know the basic software and hardware requirements for running the Hybris Commerce Suite.

**SYSTEM REQUIRMENTS:**

**For Demonstration and Development**

These requirements apply to the use of the Hybris software itself, not to 3rd party applications or databases. These requirements assume a typical scenario based on Hybris Platform, Commerce product package, PIM product package, MySQL as database, Eclipse, and embedded Tomcat web server.

The **Minimum** scenario should be sufficient for demonstrating the software in sales and presale presentations. Our tests have shown that for smooth development you need to fulfil the system requirements as listed under **Recommended**.

| **Demonstration/Development** | **Minimum** | **Recommended** |
| --- | --- | --- |
| **CPU** | Dual Core, for example i5 | Quad Core, for example i7 |
| **RAM [2]** | 8 GB | 16 GB |
| **Hard Disk [3]** | 7200 rpm IDE | SSD [2] |
| **Hard Disk Space** | 10 GB | 20 GB |

## Database

## There is no need to install a separate database if you want to try out and demonstrate the Hybris Commerce Suite. We bundled and preconfigured the lightweight HSQLDB database, which typically is sufficient for primary tests.

When you implement a Hybris solution, we highly recommend to use a professional database system such as SAP Hana, Oracle, MySQL or Microsoft SQL Server.

THIRD PARTY SOFTWARE FOR SAP HYBRIS COMMERCE SUITE

JAVA VERSION

Oracle JRE/JDK : 8.0

SAP JVM : 8.1

APPLICATION SERVER

Apache Tomcat : 7.0

Oracle Weblogic : 12.1.3

Databases:

MySQL : 5.5,5.6

HSQLDB (Single Node): 2.2.9 (for development / evaluation only).

**Build Hybris Commerce**

After all the files have been successfully downloaded and unpacked, you must build and adapt Hybris Commerce before use.

As of Release 5.5.0, you can use the installer to build preconfigured set-ups. Please refer to the [Sample Scenarios Installation](https://wiki.hybris.com/display/release5/Sample+Scenarios+Installation) document. Also see the [Documentation Release Notes](https://wiki.hybris.com/display/release5/Documentation+Release+Notes) for version-specific information.

Why Build Hybris Commerce

The reasons why you need to build Hybris Commerce are as follows:

* Hybris Commerce is an extendable complex solution, and it has a very flexible structure. During the build process, all referenced components are integrated.
* Runtime files and configuration files are created, prepared, and validated.
* Some parts of Hybris Commerce are compiled, such as:
  + Hybris [Service Layer](https://wiki.hybris.com/display/release5/ServiceLayer).
  + Other Hybris Commerce components you have implemented.

To build Hybris Commerce, follow the steps listed in the following sections:

1. Set Up Apache Ant.
2. Build Hybris Commerce.

This document guides you through the setup process of the Hybris Commerce Accelerator on your local machine. The system you install includes sample data for the demonstration storefronts.

To facilitate your installation of SAP Hybris Commerce, Hybris introduces the Installer: an automated script that takes care of creating directories, moving files, updating configuration and properties files, and initializing the system. It allows you to easily install the Hybris Commerce flavour, that is, Hybris Commerce with the desired application, such as B2C Accelerator, Telco Accelerator, or Data hub. You install the Hybris Commerce flavour using an installer recipe; an installer recipe includes Hybris Commerce and the specific application. The installer replaces the manual procedure that you used to install and initialize previous versions of Hybris Commerce.To facilitate your installation of SAP Hybris Commerce, Hybris introduces the Installer: an automated script that takes care of creating directories, moving files, updating configuration and properties files, and initializing the system. It allows you to easily install the Hybris Commerce flavor, that is, Hybris Commerce with the desired application, such as B2C Accelerator, Telco Accelerator, or Data hub. You install the Hybris Commerce flavor using an installer recipe; an installer recipe includes Hybris Commerce and the specific application. The installer replaces the manual procedure that you used to install and initialize previous versions of Hybris Commerce.

# **Installing and Initializing SAP Hybris Commerce Using the Installer for custom store setup**

Since **Hybris Commerce Suite 5.7.**, there exist two installation .zip files:

* **Commerce-suite-x.x.x.zip**: Contains all the files to install Hybris Commerce Suite and run it out-of-the-box.
* **Commerce-suite-x.x.x-repository.zip:** This file contains the Hybris Maven dependencies. It is required to extend or generate archetypes of modules that are built on the Core+ library, such as EMS and SBG. You must download and extract the contents of the **commerce-suite-x.x.x-repository. Zip** file into the same directory to which you extract the Hybris **commerce-suite-x.x.xZip** file.  
  Go to the Hybris [Download](https://wiki.hybris.com/display/release5/Download) page to download the latest version.

To install the Commerce Suite and the desired application (flavour):

On the target machine, create the folder where you will extract the Hybris Commerce installation zip files. The directory must meet the following conditions:

* + It must be close to the system root directory (for example **C:\Hybris**. You may download the file directly to the root directory, because the ZIP files already contain directory **Hybris**). Microsoft Windows requires the directory paths to be shorter than 255 characters. Any part of a directory path that exceeds this limitation is truncated. The Hybris Commerce modules installation creates several subdirectories, therefore use short directory paths.
    1. Do not use directory paths containing spaces (such as **C:\Hybris Platform**). Building Hybris Commerce fails if the directory path contains space.
    2. The Hybris Commerce installation zip file to the newly-created installation folder, for example:**C:\Hybris**.
    3. Navigate to the *{HYBRIS\_HOME}*/installer directory

**install.bat -r b2c\_acc (OR) install.bat -r b2c\_acc\_plus**

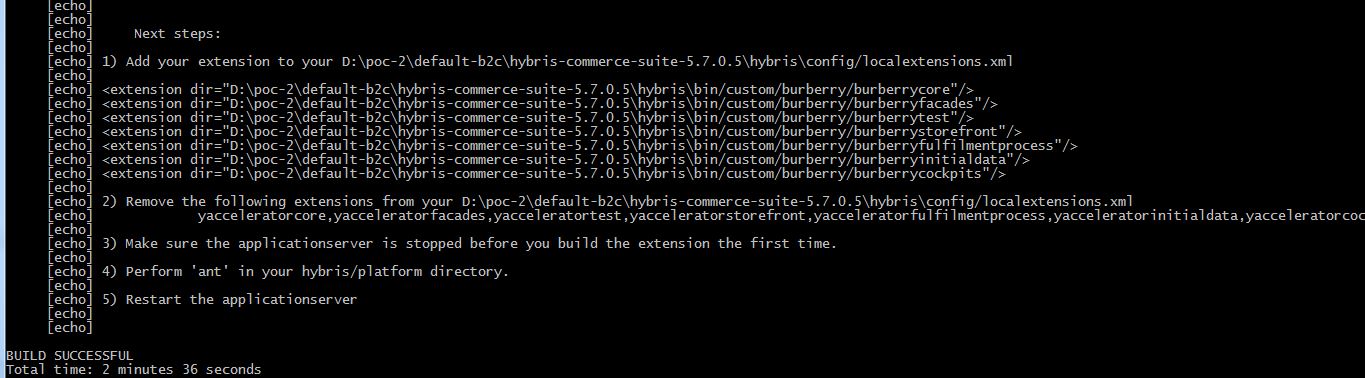
- To generate config, data, log, roles and temp folders

4. Navigate to platform directory:

**setantenv.bat to set path to the ant tool**

**ant modulegen is the command used to create a set of extensions which are depends from the accelerator to build up our own extensions. Give your own project name, package and template as develop.**

**ant modulegen -Dinput.module=accelerator -Dinput.name=burberry -Dinput.package=com.burberry -Dinput.template=develop**

****

**We have to give the above mentioned entry in to localextensions.xml file.**

**For that,**

* + 1. **edit <<Hybris-commerce-suite-5.7.0.5 HOME>>\installer\recipes\b2c\_acc\build.gradle (OR) b2c\_acc\_plus\build.gradle (replace yaccelerator by burberry Extensions and add additional extensions to the respective gradle)**

**extensions {**

**...**

**extName 'burberrycockpits'**

**extName 'burberryinitialdata'**

**extName 'burberryfulfilmentprocess'**

**extName 'burberrystorefront'**

**extName 'burberrycore'**

**extName 'burberryfacades'**

**extName 'burberrytest'**

**...**

**}**

**pl.project.addons {**

**...**

**template "burberrystorefront"**

**storeFronts "burberrystorefront"**

**...**

**}**

* + 1. Navigate to the *{HYBRIS\_HOME}*/installer directory

**install.bat -r b2c\_acc (OR) install.bat -r b2c\_acc\_plus**

**Note:** From 5.7 no additional install the **b2ccheckoutaddon**. The **b2ccheckoutaddon** has been removed because the B2C checkout functionality has been incorporated directly into the **yacceleratorstorefront** extension.

**5. platform:**

**Install all the addons as per the recipe you have choosen: (Sample below)**

**---------------------------------------------------------**

**ant addoninstall -Daddonnames="captchaaddon,liveeditaddon" -DaddonStorefront.burberrystorefront ="burberrystorefront"**

**ant clean all**

* To set up burberry Store

1. To Initialize system execute the below commands
2. platform: **ant** **initialize** command

(OR)

1. platform: if the Hybris server is up (**hybrisserver.bat**), you can select initialization in the platform tab of the Hybris Administration Console (HAC).

(OR)

1. Navigate to the {HYBRIS\_HOME}/installer directory.

**install.bat -r b2c\_acc initialize** **(OR) install.bat -r b2c\_acc\_plus initialize**

1. Start Hybris Server
2. platform**: hybrisserver.bat**

**(**OR)

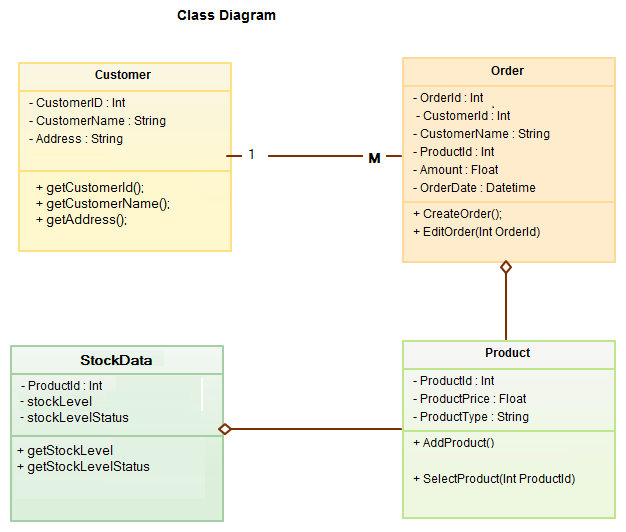
1. installer: **install.bat -r b2c\_acc start (OR) install.bat -r b2c\_acc\_plus start**
2. Check the default electronics store

<https://localhost:9002/burberrystorefront/?site=electronics>

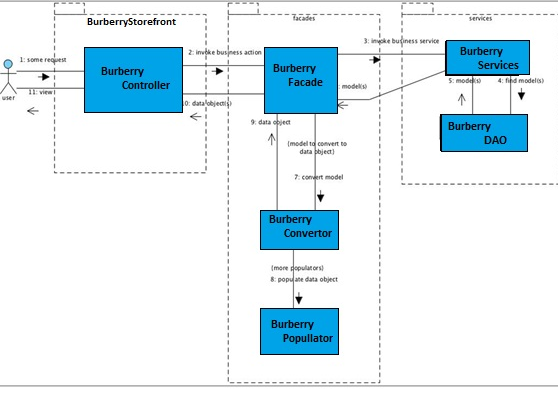
http://localhost:9001/burberrystorefront/?site=electronics

**From above steps we will finish our custom store setup.**

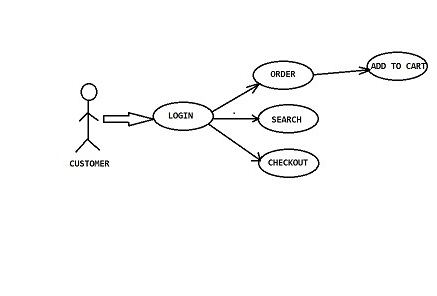
**Application Design Diagrams:-**

****

**Sequence Diagram:-**

****

**Usecase Diagram:-**

****

**Customization of Burberry Store:-**

The main objective of these steps are to create data for Our custom store. The accelerator provides two dedicated initialization hooks that allow you to import your custom data.

1. Core data hook - in our namespace that is the **CoreSystemSetup** class in burberrycore extension.
2. Sample data hook - in our namespace that is the **InitialDataSystemSetup** class in burberryinitialdata extension.

Next, we need to customize the extension's initialization hooks to setup the burberry store data as well.

Both classes are annotated with @SystemSetup and defined as beans in **burberrycore-spring.xml** and **burberryinitialdata-spring.xml**

Data Import across the Accelerator Extensions

Burberry Core initialization hook: By design it is meant to initialize the store's basic data: cockpit definitions, email templates, CMS layout, etc.

We have to review the com.burberry.core.setup.CoreSystemSetup class try to identify the method for importing the project data. With this class we should analyze which impex’s are used to import the stores basic data and non concrete store data.

**Burberry Sample Data initialization hook:**

By design this is meant to be the hook for all the product and concrete cms content initializations. All store-specific data should be defined here: catalogs, Solr configurations, etc. This hook uses import services**CoreDataImportService** and**SampleDataImportService**to import the appropriate data.

You should review the com.burberry.initialdata.setup.InitialDataSystemSetup class and try to identify the method of importing project data. We will need to modify this method to import the data.

#### Burberry Core Data Import Service

By design this is meant to provide the Burberry Sample Data hook an easy way to import all essential core data for a new store. This service is used to import basic structure of a store (i.e.: catalog definition, email templates, site definition)

Review the de.Hybris.platform.commerceservices.dataimport.impl.CoreDataImportService class and try to identify the methods of importing data structure. Note how the store's data is imported: catalogs, solr configuration, cms content, etc. We need to do the same for our Burberry shop.

#### Sample Data Import Service

By design this is meant to provide the Burberry Sample Data hook an easy way to import all the store-specific data and fill the structure defined by Burberry Core Data Import Service with concrete data (i.e.: product data, images)

We should review de.Hybris.platform.commerceservices.dataimport.impl.SampleDataImportService class and try to identify the methods of importing project data. Note how the store's data is imported: products, images, cms content, etc.  
We need to do the same for our Burberry shop.

Modify the **InitialDataSystemSetup** to include our products, cms-content during project data initialization.



## View impex files for the burberrycore extension

To learn more about which data you should import, you should review the createEssentialData() and createProjectData() methods of the CoreSystemSetup class. The impex files in the burberrycore extension define the essential data needed for the cms content and email content. Look through the impex’s and the methods which load them in CoreSystemSetup.

**Review impex files for the burberryinitialdata extension**

The basic Store data is loaded in the b**urberryinitialdata/resources/burberryinitialdata/import/coredata** directory and loaded via the CoreDataImportService class.

To learn more about which data you should import, you should review the importAllData () method of the AbstractDataImportService as well as the overridden importCommonData (), importProductCatalog (), importContentCatalog (), importStore () and importSolrIndex () methods of the CoreDataImportService class. Recall that we set these two services to be called in the createProjectData() of InitialDataSystemSetup class in an earlier step.

For creating **product catalog** related data we need a resource structure containing following file’s :

* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/productCatalogs/**b**urberryProductCatalog/catalog.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/productCatalogs/**b**urberryProductCatalog/catalog\_en.impex**

For creation of a **content catalog** related data we need a resource structure containing following files :

* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/contentCatalogs/**b**urberryContentCatalog/catalog.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/contentCatalogs/**b**urberryContentCatalog/catalog\_en.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/contentCatalogs/**b**urberryContentCatalog/cms-content.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/contentCatalogs/HybrisContentCatalog/cms-content\_en.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/contentCatalogs/**b**urberryContentCatalog/email-content.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/contentCatalogs/**b**urberryContentCatalog/email-content\_en.impex**

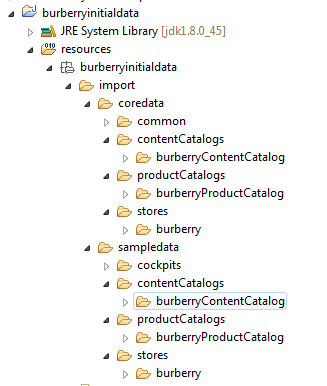
For creation of**store** related data we need a resource structure containing following files :

* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/stores/**b**urberry/store.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/stores/**b**urberry/store\_en.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/stores/**b**urberry/site.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/stores/**b**urberry/site\_en.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/stores/**b**urberry/solr.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/stores/**b**urberry/solr\_en.impex**
* b**urberryinitialdata/resources/**b**urberryinitialdata/import/coredata/stores/**b**urberry/solrtrigger.impex**

## Rename the project folders

Thankfully, starting with the 5.1 accelerator, we don't have to create the above files by hand; all we have to do is rename the catalogName*and* storeName folders in b**urberry**initialdata/resources/b**urberry**initialdata/import/coredatato the catalog and store name we want to use.

For example, Refer to the following screenshot:



Do the same for

**Burberryinitialdata/resources/burberryinitialdata/import/sampledata**

Basic store data:-

The files located in the above folders need to be modified.

In **store.impex** you must add new macro:

$storePrefix=burberry

Because we are not going to use classification system we can remove the following line:

$classificationCatalog=\_\_CLASSIFICATION\_CATALOG\_NAME\_\_

and remove that macro from BaseStore catalogs line.

We should also add some delivery (shipping) methods by adding these lines to the impex file.

For adding custom shipping method add following lines in **store.impex**

INSERT\_UPDATE BaseStore2DeliveryModeRel;source(uid)[unique=true];target(code)[unique=true]

; burberry; premium-net

;burberry; standard-net

;burberry; premium-gross

;burberry; standard-gross

;burberry; pickup

;burberry; free-standard-shipping

;burberry; burberry-shipping

Add same thing in **delivery-modes.impex** with the delivery cost and

**Delivery-modes\_en.impex** according to the zonal things.

**Burberrycore/resources/burberrycore/import/common/delivery-modes.impex**

**Burberrycore/resources/burberrycore/import/common/delivery-modes\_en.impex**

**NOTE:-**

To display the delivery mode on front-end we have to make the **net** as to be **false** to know the Jsp condition (gross=true).

Modify the **store\_en.impex** file as above for the localization of the store.impex

**Store\_en.impex:-**

$lang=en

$storeUid=burberry

# Create Base Store

UPDATE BaseStore;uid[unique=true];name[lang=$lang]

;$storeUid;"burberry Store"

**site.impex :-**

Adds basic site information to the configuration of the Burberry shop - again you need to substitute several macros in the same way as in the above **store.impex** file.

**Burberryinitialdata/resources/burberryinitialdata/import/coredata/stores/burberry/site.impex**

**Burberryinitialdata/resources/burberryinitialdata/import/coredata/stores/burberry/site\_en.impex**

# Language

$lang=en

# Create CMS Site

UPDATE CMSSite;uid[unique=true];name[lang=$lang];locale[lang=$lang]

;burberry;"burberry Site"; en\_US

Update the basic solr configuration, including information for ranges, facets, index names, etc...

**Note:** In the version of Hybris 5.7. the solr server configuration should be in standalone mode.

**burberryinitialdata\resources\burberryinitialdata\import\coredata\stores\burberry\solr.impex**

Add the following lines of code in **solr.impex** file.

# Create the solr server configuration

INSERT\_UPDATE SolrServerConfig;name[unique=true];mode(code);embeddedMaster

;$serverConfigName;standalone;false;

INSERT\_UPDATE SolrEndpointUrl;solrServerConfig(name)[unique=true];url[unique=true];master[unique=true,default=false]

;$serverConfigName;http://localhost:8983/solr;true

Update solr\_en, solrtrigger, site, site\_en impex’s as well.

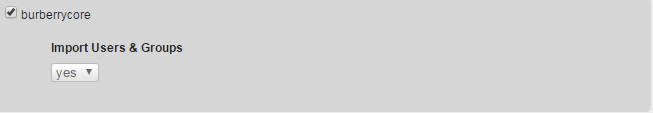
Follow the same with the content catalog data impex’s and product data impex’s in the

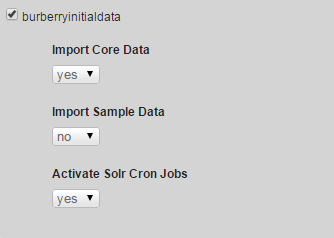
Burberryinitialdata extension.

**CatalogVersion synchronization jobs**

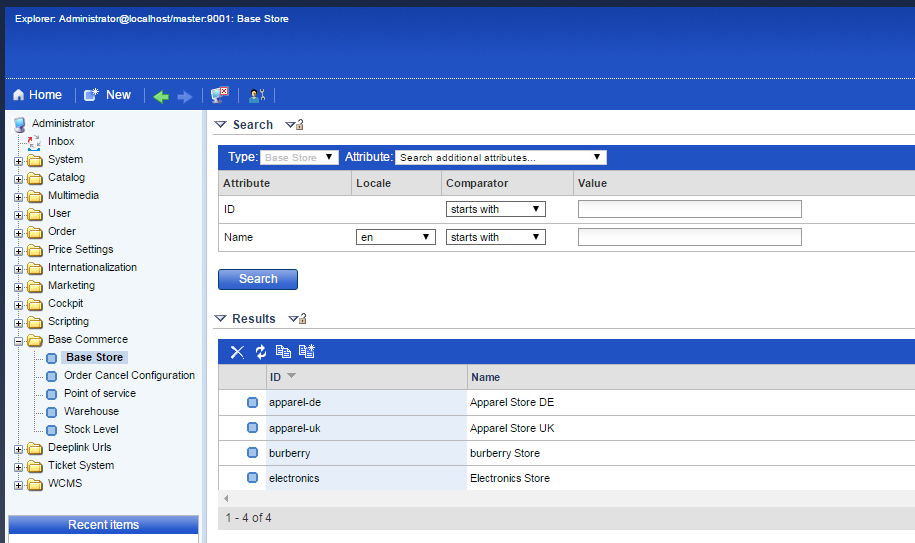
Notice that the **createProjectData ()** method includes code that creates Cronjob’s to synchronize catalog versions. These jobs target our new catalog version’s (content and product) and can be run at any time to synchronize CMS content and products from their staged to online versions.

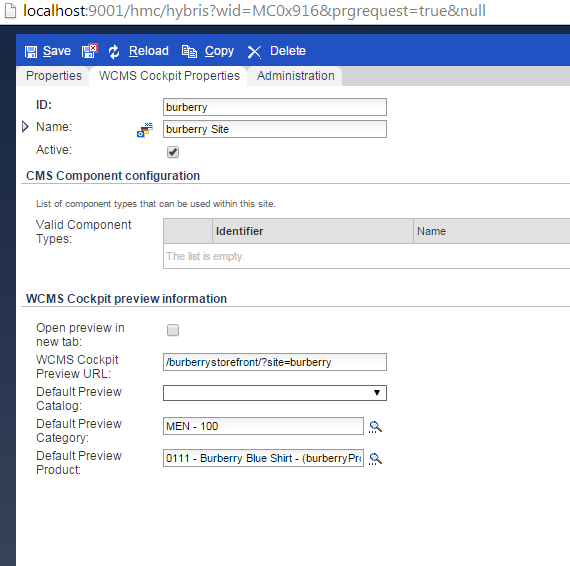
After completing the above modifications execute **ant all** command and **update the project data** with the extension **burberrycore and burberryinitialdata** but allow only **import core data and active solr** Cronjob’s active in burberryinitialdata extension as follows.





Once you have imported the data, you can go to the [hmc](http://localhost:9001/hmc/hybris)and look in the Base Commerce section that we have a new store called **'burberry'** and it has a website assigned with the same name.





For Reference purpose all the impex files under core data have been given below**:**

****

**Create products in Burberry store:**

Create the product categories in the **categories.impex**

The localization is done in the **categories\_en impex**

Import available products to our store with variants using the products .impex, product\_en.impex and media through product-media. Impex and stock levels in product-stocklevels.impex and prices with product-prices.impex.

**Activating Solr index for the product content.**

Accelerator indexes categories names in a way that it requires additional dedicated spring beans for resolving category names, codes per each configured store.

Add the following beans definitions in **burberrycore-spring.xml**

<bean id=*"burberryCategorySource"* parent=*"abstractCategorySource"*>

<property name=*"rootCategory"* value=*"1"*/>

</bean>

<bean id=*"burberryBrandCategorySource"* parent=*"abstractCategorySource"*>

<property name=*"rootCategory"* value=*"brands"*/>

</bean>

<bean id=*"burberryCategoryCodeValueProvider"* parent=*"abstractCategoryCodeValueProvider"*>

<property name=*"categorySource"* ref=*"burberryCategorySource"*/>

</bean>

<bean id=*"burberryBrandCategoryCodeValueProvider"* parent=*"abstractCategoryCodeValueProvider"*>

<property name=*"categorySource"* ref=*"burberryBrandCategorySource"*/>

</bean>

<bean id=*"burberryCategoryNameValueProvider"* parent=*"abstractCategoryNameValueProvider"*>

<property name=*"categorySource"* ref=*"burberryCategorySource"*/>

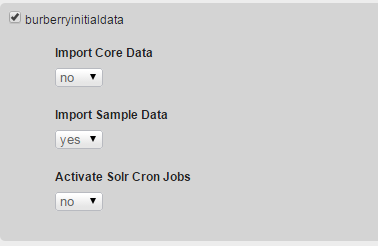
</bean>

<bean id=*"burberryBrandCategoryNameValueProvider"* parent=*"abstractCategoryNameValueProvider"*>

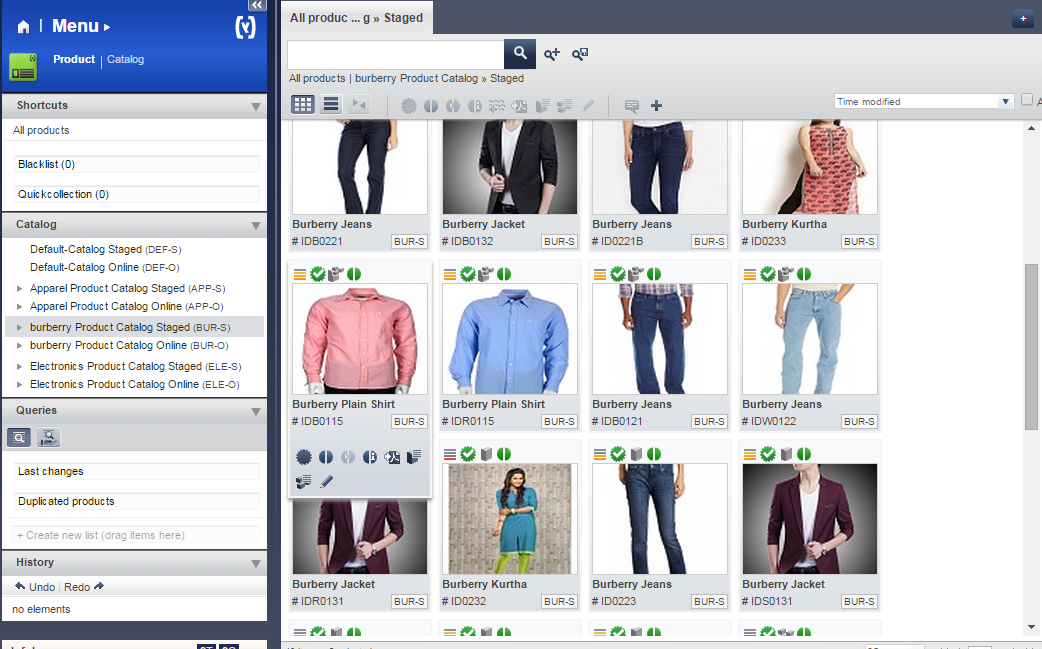
<property name=*"categorySource"* ref=*"burberryBrandCategorySource"*/>

</bean>

After this steps update project data with **burberryinitialdata**



After this check your product cockpit it should be updated with your products.



**Create Site CMS Content using Impex:**

Modify the cms content by changing the catalog name. Create the navigation links and navigation nodes to our store. Add the banners to the home page as follows

**Cms-content.impex**

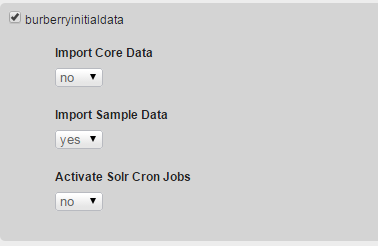
Provide the path to the site logo component to get the media from the provided images folder in the **contentCatalog/burberryContentCatalog/images/**

Add cms link components to create the navigation to products based on the categeorie’s.

Add product carousel to add best selling products into that and banner component to the any of the section.

**Cms-content\_en.impex:-**

Now update your project data with the following settings:



After successful update restart server and navigate to url home page

<https://localhost:9002/burberrystorefront/?site=burberry>

http://localhost:9001/burberrystorefront/?site=burberry



For Reference purpose all the impex files under sample data have been given below**:**

****

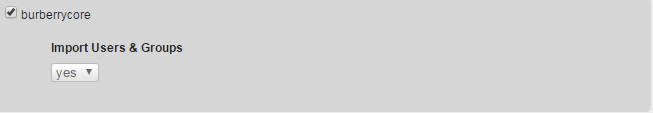
**Setup for variant products:-**

In the **burberrycore-item.xml** add Burberry **typegroup** and declare **BurberryStyleVariant** type which extends **VariantProduct** with attribute style. Declare **BurberrySizeVariant** type which extends **BurberryStyleVariant** with the attribute size.

Stop the server & Run the command in Hybris platform path: **ant all**, Generates models, DAO, DTO and Jalo classes are created for above specified item types.

Refresh the workspace and start the server using command **hybrisserver.bat** in Hybris platform path.

Update the system using HAC with the project data with burberrycore. It adds the newly created attributes to the database.



Now add following code to the **productVariantSelector.tag** file to show the variants in frontend as follows

<%-- Determine if product is one of Burberry style or size variant --%>

<c:if test=*"*${product.variantType **eq** 'BurberryStyleVariantProduct'}*"*>

<c:set var=*"variantStyles"* value=*"*${product.variantOptions}*"* />

</c:if>

<c:if test=*"*${(**not empty** product.baseOptions[0].options) **and** (product.baseOptions[0].variantType **eq** 'BurberryStyleVariantProduct')}*"*>

<c:set var=*"variantStyles"* value=*"*${product.baseOptions[0].options}*"* />

<c:set var=*"variantSizes"* value=*"*${product.variantOptions}*"* />

<c:set var=*"currentStyleUrl"* value=*"*${product.url}*"* />

</c:if>

<c:if test=*"*${(**not empty** product.baseOptions[1].options) **and** (product.baseOptions[0].variantType **eq** 'BurberrySizeVariantProduct')}*"*>

<c:set var=*"variantStyles"* value=*"*${product.baseOptions[1].options}*"* />

<c:set var=*"variantSizes"* value=*"*${product.baseOptions[0].options}*"* />

<c:set var=*"currentStyleUrl"* value=*"*${product.baseOptions[1].selected.url}*"* />

</c:if>

<c:url value=*"*${currentStyleUrl}*"* var=*"currentStyledProductUrl"*/>

And add the map entry in **burberryfacades-spring.xml** to fetch the media to frontend using map key and value pair as follows

**burberryfacades-spring.xml:-**

<alias name=*"defaultAcceleratorVariantOptionDataPopulator"* alias=*"acceleratorVariantOptionDataPopulator"*/>

<bean id=*"defaultAcceleratorVariantOptionDataPopulator"* class=*"com.burberry.facades.populators.AcceleratorVariantOptionDataPopulator"*

parent=*"variantOptionDataPopulator"* >

<property name=*"mediaService"* ref=*"mediaService"*/>

<property name=*"mediaContainerService"* ref=*"mediaContainerService"*/>

<property name=*"typeService"* ref=*"typeService"*/>

<property name=*"imageFormatMapping"* ref=*"imageFormatMapping"*/>

<property name=*"variantAttributeMapping"*>

<map>

<entry key=*"ApparelStyleVariantProduct.style"* value=*"styleSwatch"*/>

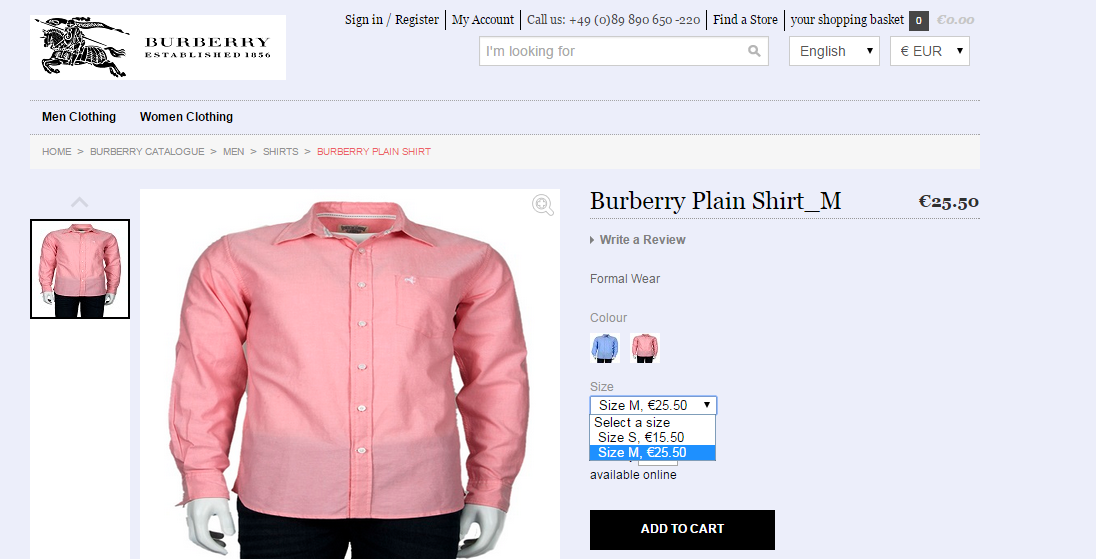
<entry key=*"BurberryStyleVariantProduct.style"* value=*"styleSwatch"*/>

</map>

</property>

</bean>

With this change we can achieve both size and colour variants in the frontend.

****

**Adding promotions to our products:-**

**Defining Customized Promotion type and enable it**

To avail offer for customers, the promotions are used. Define the new type of promotion:

**Buy X items of product A, get product B for free!**

We need to create the burberrycommerceservices extension using following steps:

1. In platform, **setantenv.bat**
2. Run the following command **ant extgen**
3. Choose the **yaddon**  template
4. Give extension as **burberrycommerceservices**
5. Package name **com.burberry.burberrycommerceservices**

After extension created successfully, add following extension in your **config/localextentions.xml**

**<extension name='burberrycommerceservices' />**

And run ant **all.**

A promotion type is an entity we need to define in **burberrycommerceservices-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**

**burberrycommerceservices-items.xml :-**

****

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).

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

**burberrycommerceservices/src/de/Hybris/burberry/jalo/ProductQuantityThresholdFreeGiftPromotion.java**



package  com.burberry.jalo;

public class ProductQuantityThresholdFreeGiftPromotion extends GeneratedProductQuantityThresholdFreeGiftPromotion

{

    @Override

public List<PromotionResult> evaluate (final SessionContext ctx, final PromotionEvaluationContext promoContext)

    {

// Here need to write the Business logic and return the List of PromtionResult object

 }

}

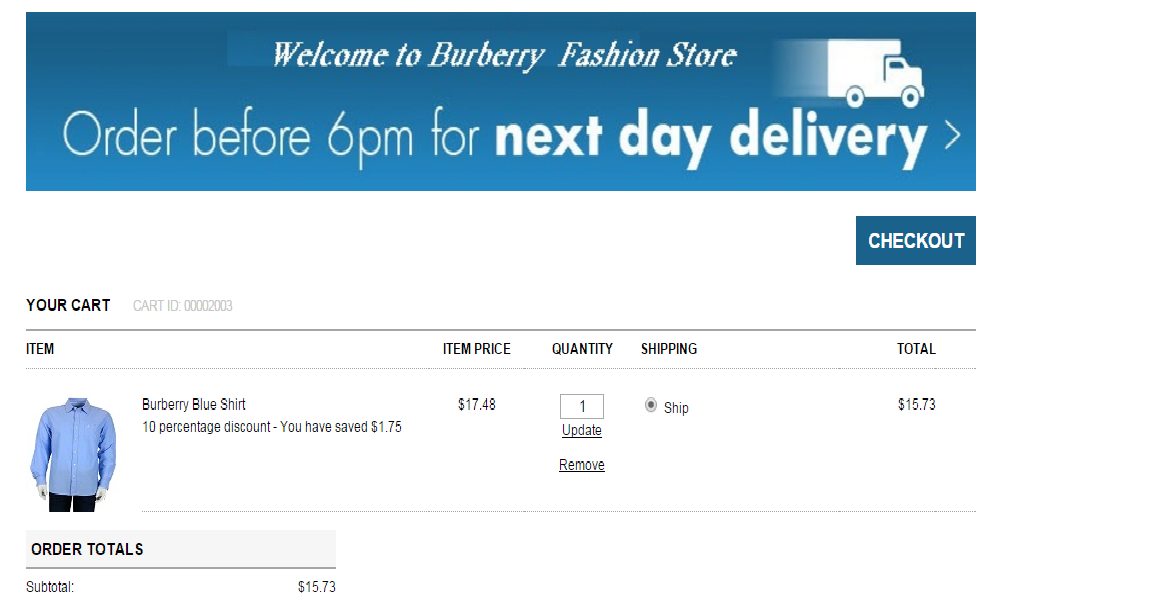
**/burberryinitialdata/resources/burberryinitialdata/import/sampledata/stores/burberry/promotions.impex**

****

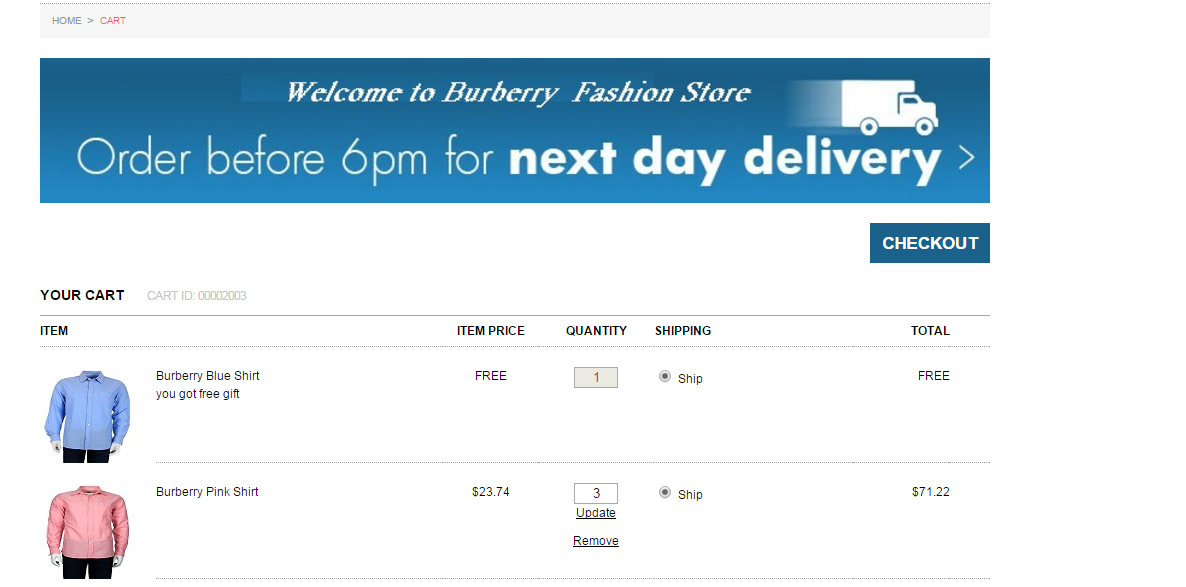
#### Rebuild and Verify

Rebuild the code base and start the server and update from HAC.

**10% percentage discount promotion for a particular product**

****

**Buy 3 Burberry Pink Shirt will get 1 Burberry blue shirt free:**



**Vouchers:-**

Voucher’s are a special form of discounts that can be applied to an order.  vouchers have to be actively redeemed by the customer.

There are 2 types of Vouchers available in HMC

1. Promotional Voucher
2. Serial Voucher

Promotional Voucher:-

With Promotional Vouchers Voucher code can be redeemed several times.

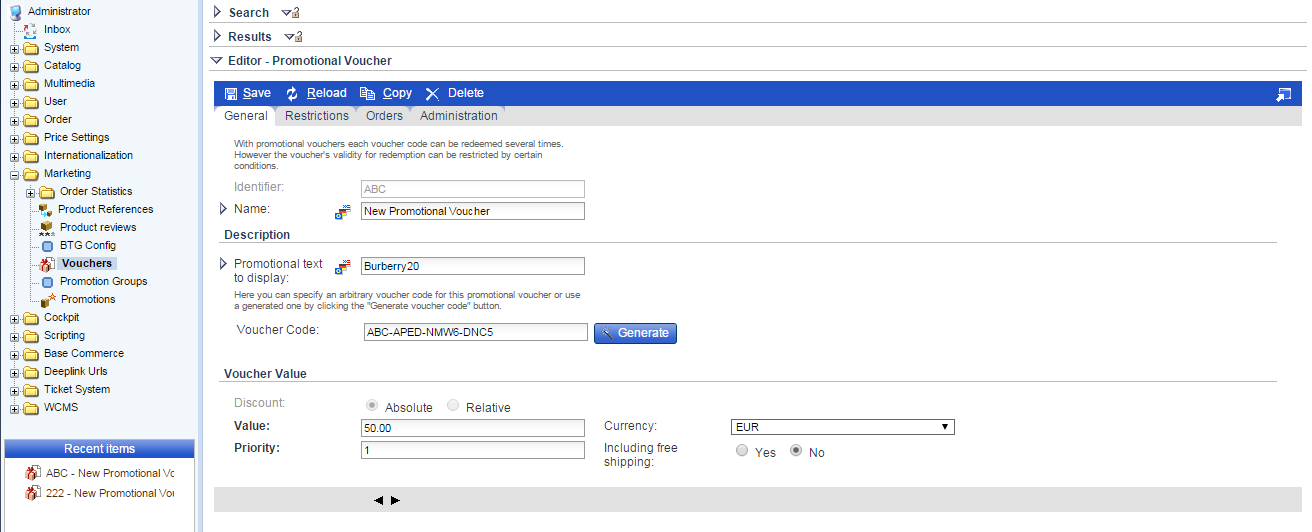
To Create the Promotional Voucher through HMC.

Go to HMC-🡪Marketing-🡪Vouchers-🡪Right-click on Vouchers-🡪click on create-🡪select Promotional Voucher

It opens new Promotional Voucher window, in that give the identifier name as like ‘ABC’ , Promotional text to display as like ‘Burberry20’ , value as like ‘50’ ,Currency as like ‘EUR’ and click on Create button. After that click on Generate button it’s generate Voucher code. We have to redeem this voucher code in our Burberry Site.

For this we have created one textbox & apply button in silentOrderPostPage.jsp, and create one controller like DefaultVoucherController.java for redeem this voucher code.

Creating Promotional Voucher through HMC.



After creating this New Promotional voucher & it’s code .We have to redeem this Voucher code in our Burberry Site.

For this we have created one textbox & apply button in silentOrderPostPage.jsp, and create one controller like DefaultVoucherController.java for redeem this voucher code.

Add the following lines of code in the given Jsp.

Burberrystorefront/web/webroot/WEB-INF/views/desktop/pages/checkout/multi/silentorderpostPage.jsp

**silentOrderPostPage.jsp:-**

………………………..

<c:url value=*"/checkout/voucher/redeem"* var=*"applyVoucherURl"*/>

<form action=*"*${applyVoucherURl}*"* method=*"GET"*

<p>Voucher Code :<input type=*"text"* class=*"text"* name=*"voucherNumber"* /> </p>

<input type=*"submit"* value=*"Apply"* />

</form>

......................

Create the controller class like DefaultVoucherController, to add the given lines of code in your controller.

**Burberrystorefront/web/src**

**DefaultVoucherController.java**

****

package com.burberry.storefront.controllers.pages;

@Controller

@Scope("tenant")

@RequestMapping(value = "/checkout/voucher")

public class DefaultVoucherController extends AbstractPageController

{

// voucher code redeem related code to check in store

@RequestMapping(value = "/redeem", method = RequestMethod.GET)

public String redeemVoucher(@RequestParam ("voucherNumber") final String voucherCode, final Model model, final HttpServletRequest request)

{

………….

…………….

}

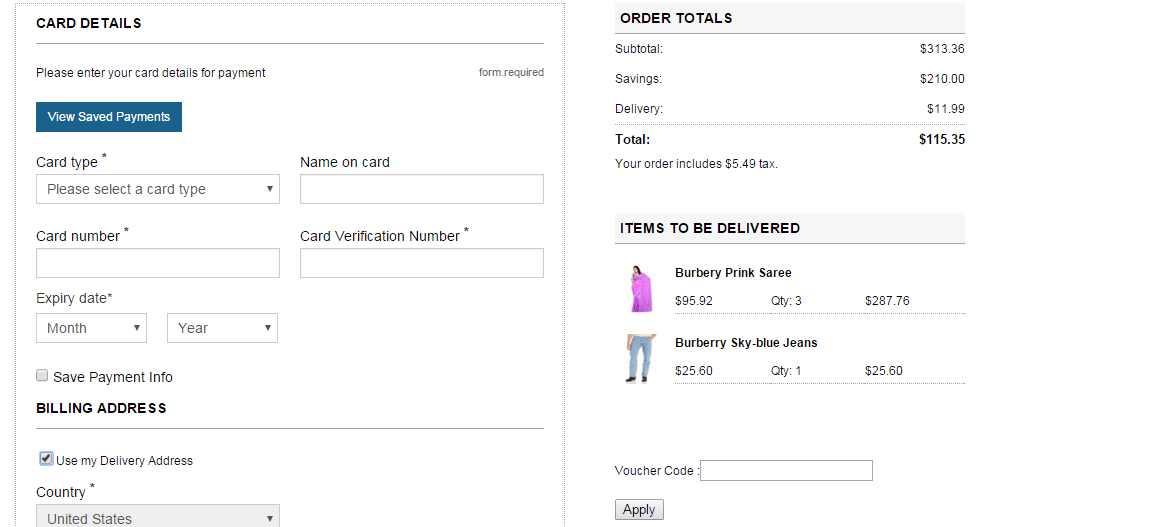
}

After that you run following command & refresh the workspace. Start the Hybris server & check the Burberry Site.

<Hybris\_home>\Hybris\bin\platform>ant all

D:\HYBRISCOMM57\Hybris5.7.0.5\Hybris\bin\platform>Hybrisserver.bat

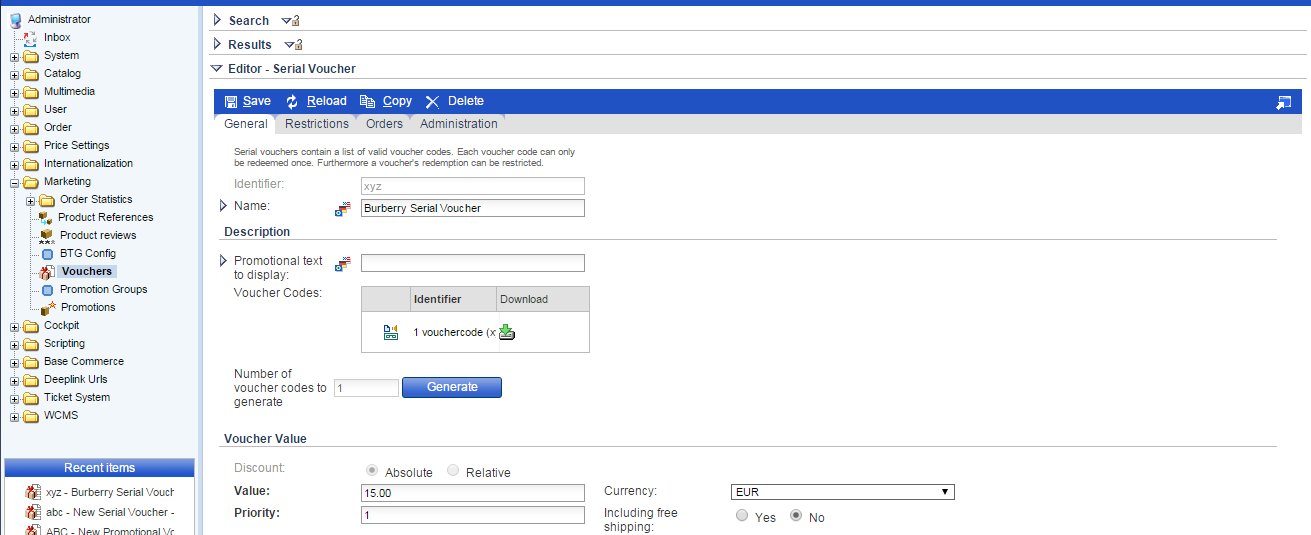
Go to Burberry site check the Voucher code textbox & apply button. There you can apply your voucher code.



**Serial Voucher :-**

With Serial Vouchers Voucher code can be redeemed once. Serial vouchers contain a list of valid voucher codes.

Creating Serial Voucher through HMC

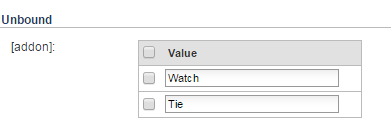


Implementation steps for the adding product addons

1. Create the attribute of Map type with key, value as String, Boolean in your itemtype in burberry-items.xml



1. Localize the attribute in the burberry-locales\_en.properties file.
2. Open the hMC --> Catalog --> Product-->Administration



Add the data as shown above.

1. ProductPageController handles request for the addons. Add all the addons values created from hMC in the product model.



Put all the addons’ data in map.

1. Display the data exists in the map on PDP through addtocartaction.jsp file.

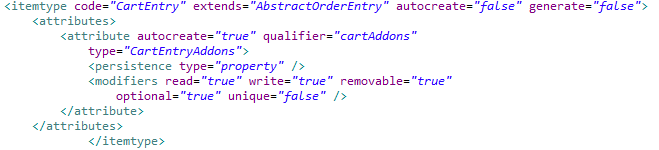


Now user will select some of the addons. Iterate the data and pass the selected addons keys to the BurberryAddToCartForm through hidden input values.

1. AddToCartController handles request for adding the product to the cart. Pass the selected values through the addtocart method of the AddToCartController.



1. Now create one attribute in the CartEntry itemtype, the attribute must be of collection type.





Localize the attribute.

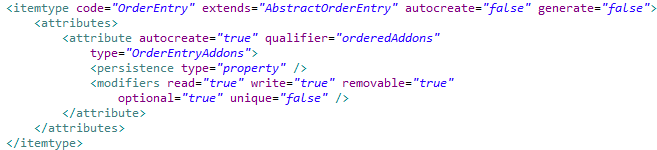
1. Now create a cartModel and get the cartEntries through it. Add the selected addons list in the CartEntryModel for the matching product codes and save the modelService.
2. Now open hMC-->Marketing --> Order Statistics-->cart-->cart entries-->Administration



You can see the selected addons which was added with the product here for current cart session.

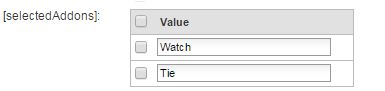
Now addons are here in the cart entry and after placing the order it should get removed from here. Once the order is placed it must be seen in the order entries of hMC.

1. When user places the order, the **placeOrder** method of SummaryCheckoutStepController is invoked. Get all the selected addons and add the list in the OrderEntryModel and save the modelService.
2. Create an attribute in the itemtype orderEntry to show the addons in order entries of hMC and localize it.





1. Now open hMC-->Order --> Order Entries -->Administration



You can see the selected addons which was added with the product after placing the order successfully.

1. Create a property for bean OrderEntryData to set the selected addons in the orderEntryAddons attribute for cart entries and order entries.



1. Create a populator for populating the data through OrderEntryModel.



Now we have the selected addons list in the OrderEntryModel.

And also create bean for the populator.



1. Show the addons on corresponding pages with the help of the following jsp and tag files.

|  |  |
| --- | --- |
| File Name | Description |
|  | Display on PDP addtocartaction.jsp |
|  | To display on cart page. |
|  | On cart popup |
|  | On Address details, Payment method, Shipment and payment fulfillment |
|  | On order confirmation page |
|  | In my account on order history view page |

Implementation for coming soon products

1. Create new attribute “**isComingSoon**” in product itemtype in **burberrycore-items.xml** file.

|  |
| --- |
| <itemtype code=*"Product"* extends=*"GenericItem"* autocreate=*"false"* generate=*"false"*>  <attributes>  <attribute qualifier=*"isCommingSoon"* type=*"java.lang.Boolean"*>  <description>Coming Soon Products</description>  <modifiers />  <persistence type=*"property"* />  </attribute>  </attributes>  </itemtype> |

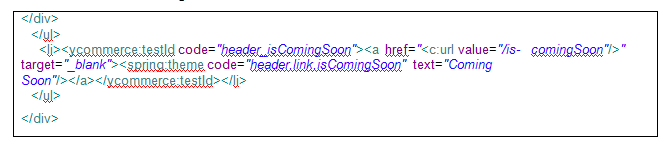
1. Navigate to platform directory in command prompt and run following commands.

**ant clean all**

**hybrisserver.bat**

1. Go to hybris admin console <http://localhost:9001/> and update system by selecting Update Running System.
2. Create coming soon link to display coming soon products. You can create any where you want that link.

I have created in **header.tag** file.



1. Create new template for coming soon product page in **coredata/cms-content.impex**



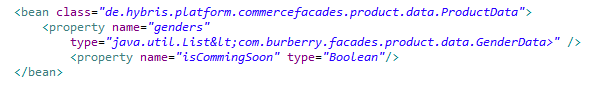
1. Create new **comingSoonLayout.jsp** for coming soon product page in **coredata/cms-content.impex**

(burberrystorefront\web\webroot\WEB-INF\views\desktop\pages\layout\comingSoonLayout.jsp)

1. Create controller **ComingSoonController.java** to redirect to that page.



1. Add **isCommingSoon** attribute in **burberryfacade-beans.xml** file inside ProductData bean



1. Now we are going to create **ComingSoonDao.java** to retrieve coming soon products from product table.



1. Write bean entry of **ComingSoonDao** in **burberrycore-spring.xml** file & build the system (**ant clean all**).



1. Create **ComingSoonService.java**



1. Write bean entry of **ComingSoonService** in **burberrycore-spring.xml** file. Give reference of above Dao & build the system (**ant clean all**)

|  |
| --- |
| <bean id=*"ComingSoonService"* class=*"com.burberry.product.impl.ComingSoonService"*>  <property name=*"ComingSoonDao"* ref=*"ComingSoonDao"*/>  </bean> |

1. Change the controller to get coming soon products from service & set in model.
2. Change **comingSoonLayout.jsp** to show your coming soon products on page.

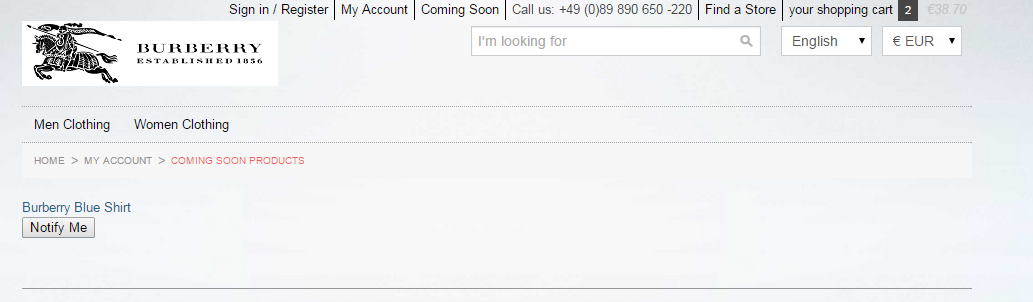


1. Build the System (**ant all**) and start the hybris server (**hybrisserver.bat**).
2. Go to Hybris Management Console & change isComingSoon status **Yes** for few products in online catalog & save.
3. Go to Your Site and click on **Coming Soon** link.

Home page



Coming soon page



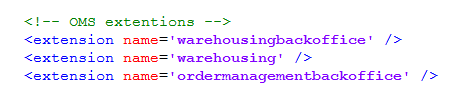
INSTALLING ORDER MANAGEMENT SERVICE (OMS)

Order Management System allows admin to customize the management system as needed. In burberry OMS, we have added warehousing, warehousingbackoffice and yacceleratorordermanagement sub project to manage order effectively.

|  |  |
| --- | --- |
| **Yacceleratorordermanagement (default)** | * Template that defines workflows for OMS orders, consignments, returns |
| **warehousingbackoffice** | * Extension that integrates OMS UI into the Backoffice * Manage consignments, stock levels, POS |
| **warehousing** | * Extension that contains the warehousing business logic * Items, beans and Spring configuration that are used by the warehousing services * Services and DAOs |
| **ordermanagementbackoffice** | * Extension that integrates OMS UI into the Backoffice * Order cancellation, returns, fraud management, payment transactions |

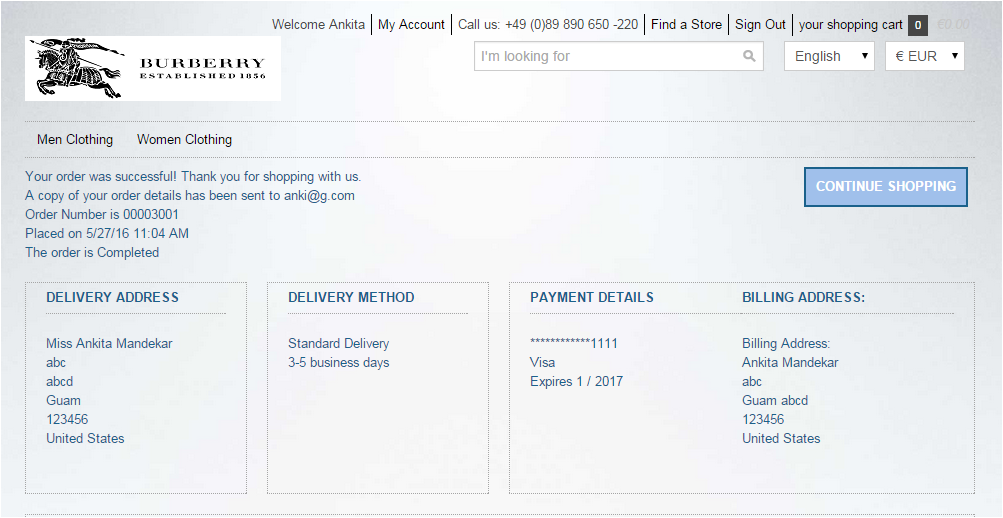
For Burberry OMS Setup, we have used latest Hybris 5.7.5 commerce suite. For installation we need to run:

1. **SETUP: install.bat -r b2c\_acc\_oms setup**  in installer Hybris folder
2. **INITIALIZE: install.bat -r b2c\_acc\_oms initialize**
3. **START HYBRIS SERVER:  install.bat -r b2c\_acc\_oms start**
4. **ECLIPSE :** Importall the project which are present in localextention.xml in workspace



1. Now, test the OMS Installation:

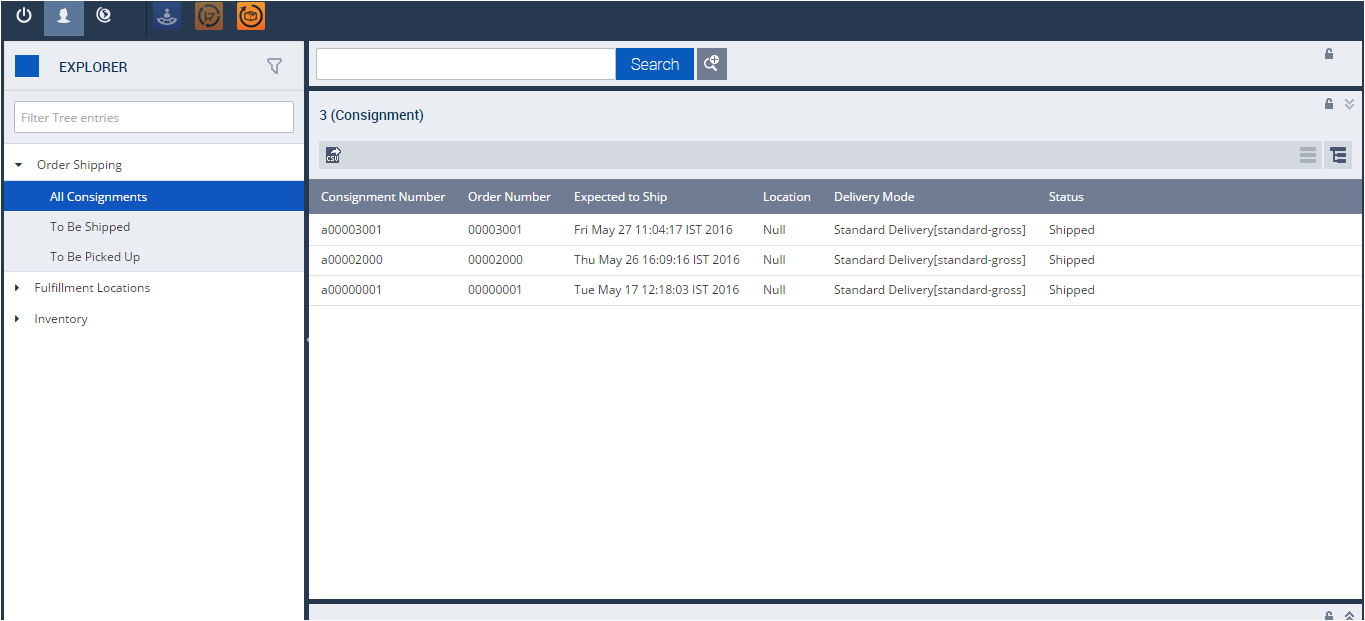
After placing Order on burberry Store:



In above image shows the order placed and its number on burberry online store.

Browse: <http://localhost:9001/backoffice/> then click on the Order Fulfilment icon at the top of the window. Then go to "Order Shipping" and click on "All Consignments". For version 5.7, although the model allows multiple warehouses per point of service, you should make sure your data only has ***one*** warehouse per point of service.

You should see a screen similar to the following screen with the appropriate consignments created for your order(s):



Above image shows the consignment a00003001, order number 00003001 and its status as shipped. Using OMS recipe, admin can track the order, it’s consignment number and shipping details.

**PAYPAL INTEGERATION WITH BURBERRY**

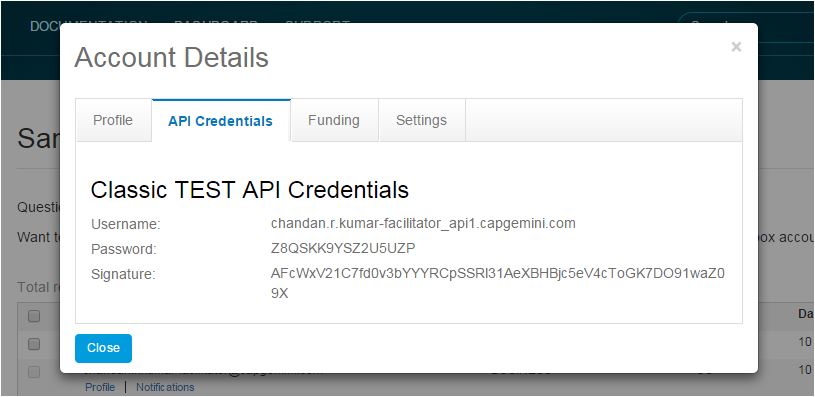
**PayPal** is the faster, safer way to send money, make an online payment, receive money or set up a merchant account. We have integerated paypal with burberry store to avail customers with faster transactions to purchase online.

1. **Configuring with Paypal**
2. For development and testing purposes you firstly need to register PayPal Sandbox business account at <https://www.developer.paypal.com/>
3. Go to http://www.developer.paypal.com and log in (Production PayPal credentials are required to access the developer portal)
4. Click **Dashboard** link at the top of the page
5. Click **Accounts** in the Sandbox section
6. You can now create a Sandbox account to receive test API credentials. Click **Create Account** at the top of the page and populate the following fields:

|  |  |
| --- | --- |
| **Account Type** | Tick either the **Personal** or **Business** radio button. |
| **Email address** | The email address doesn't need to be a real email address; the Sandbox doesn't send any email outside of the Sandbox environment. Email to Sandbox accounts, generated as a result of your test API requests, are listed on both the Notifications tab on the Developer site, as well as on the Sandbox test site.  Use the assigned email value to reference this test account in your API calls, and to log in to the Sandbox site when you want to review the details associated with the account. |
| **Password** | The password must be 8-20 alpha-numeric characters in length. Use the password to log in to the Sandbox site as the test account. |
| **First and Last names** | The optional name fields accept alpha-numeric characters. |
| **PayPal balance** | While this field is optional, it's a good idea to create test accounts with positive bank balances. Enter an integer value between 1 and 5000. |
| **Bank Verified Account** | You should create both *Verified* and *Unverified* test accounts so you can fully test your application. |
| **Select Payment Card** | Test payments made with different payment cards by selecting either **Discover** or **PayPal**. |
| **Credit card type** | You must select a single credit card type for each test account. The Sandbox associates a mock credit card number with the account. |

1.6 Once the follow details have been entered click the **Create Account** button

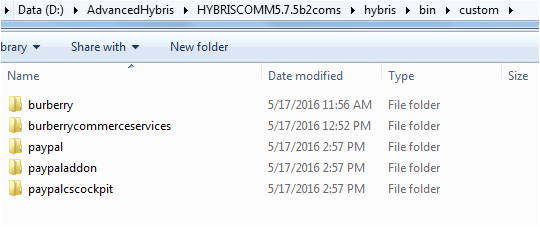
1.7 To locate the API credentials for your Sandbox PayPal account, click on the account you would like to view credentials for then select the **Profile** link.



1. **Installing Add-on**

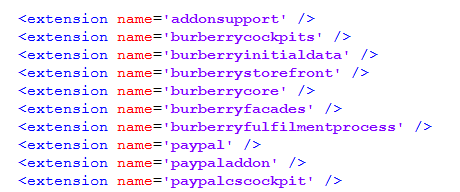
**Step1: Downloading Add-on**

1. Download adapter archive from the hybris marketplace
2. Unpack archive and place folders **paypal**, **paypaladdon** and **paypalcscockpit** into **..\bin\custom**

****

**Step2: Update localextensions.xml**

1. Check the presence of **<extension name="addonsupport" />**
2. Add extensions **<extension name="paypal"/>** and **<extension name="paypaladdon"/>** and **<extension name="paypalcscockpit"/>**
3. Check the presence of target storefront extension



**Step3: Run installation command**

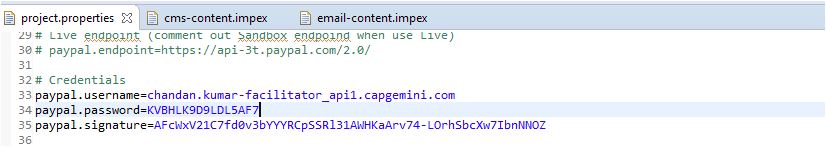
1. Stop hybris if it’s running with Ctrl+C
2. Run **setantenv.bat** if it was not applied in this terminal
3. Go to **..\bin\platform** and run installation command:

**ant addoninstall -Daddonnames=paypaladdon –DaddonStorefront.burberrystorefront= burberrystorefront**

The output should end with BUILD SUCCESSFUL

**Step4: Main configurations**

1. Go to **.\bin\custom\paypal\project.properties**



1. Find and update the following configurations: **paypal.username**, **paypal.password**, **paypal.signature** and **paypal.seller.email** with your API credentials from PayPal business account.

**Step 5: Rebuild the system**

**Step 6: Update**

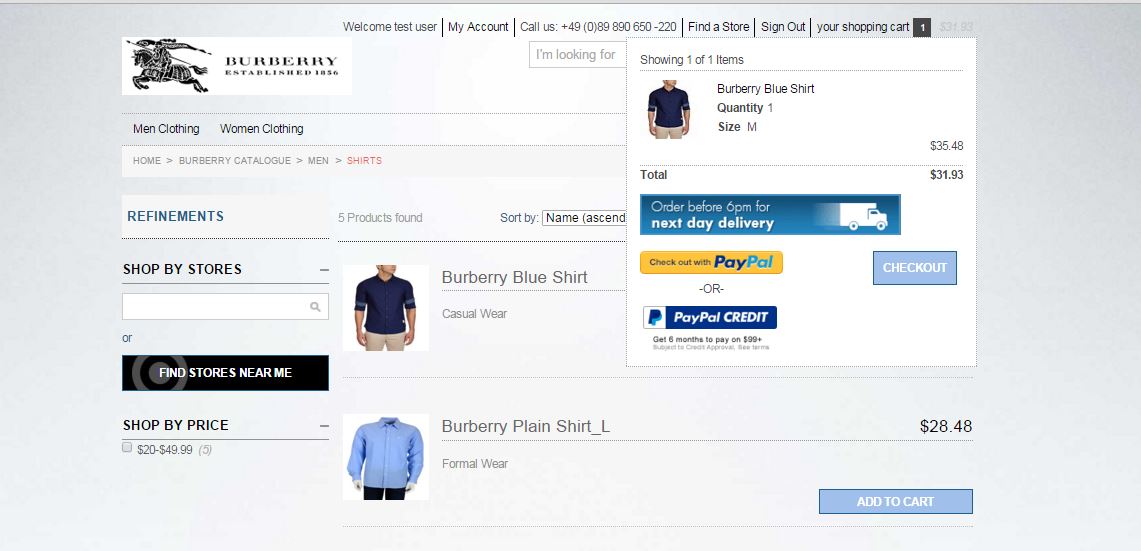
At this step you may make **full initialization** through hAC if you use clean hybris for testing purposes. And that is 1 sufficient step to finish add-on installation.

If you can’t perform full initialization then you need to make more complicated update:

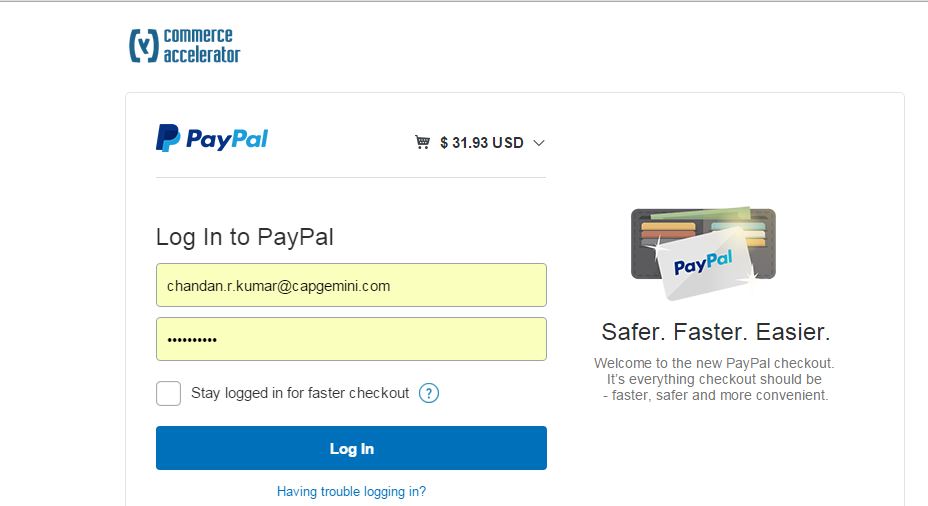
1. Go to hAC->Platform->Update
2. Choose 4 update options **Update running system**, **Clear the hMC configuration from the database**, **Create essential data**, **Localize types**
3. Choose add-on components to update: **paypal**, **paypaladdon**, **paypalcscockpit**
4. Then make update
5. After update finishes go to **Console**->**Impex Import**.
6. You must insert script here. You should use one of ready scripts (if it is accelerator based installation) or customize one for your content catalog.
7. You can find ready scripts in add-on files: **..\bin\custom\paypaladdon\resources\paypaladdon\import\contentCatalogs\**
8. Import content
9. Go to hMC->Catalog->Catalog Management Tools->Synchronization
10. Choose Source Catalog Version and execute Synchronization

**Step 7: Placing Order on Burberry Online Store using Paypal**

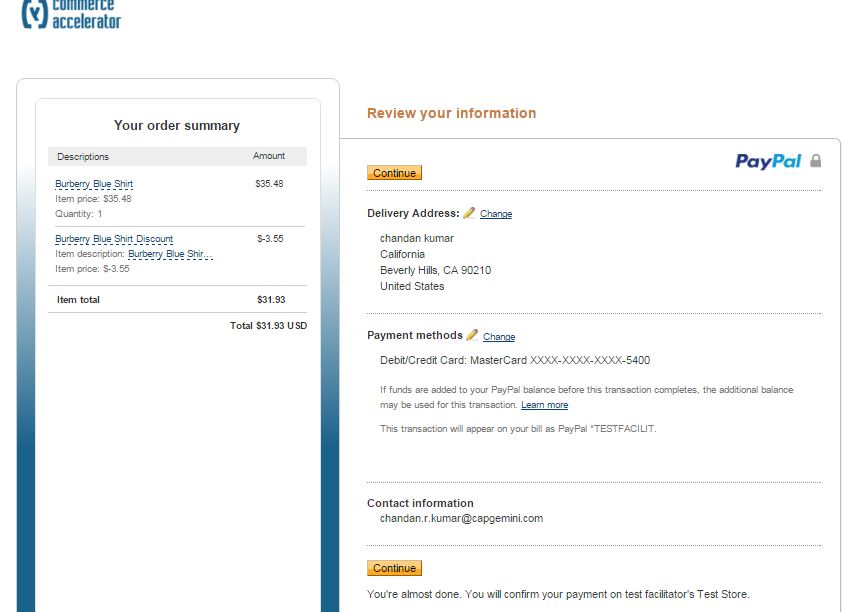
After selecting the order, use PayPal payment to checkout from popup cart on Burberry site:

****

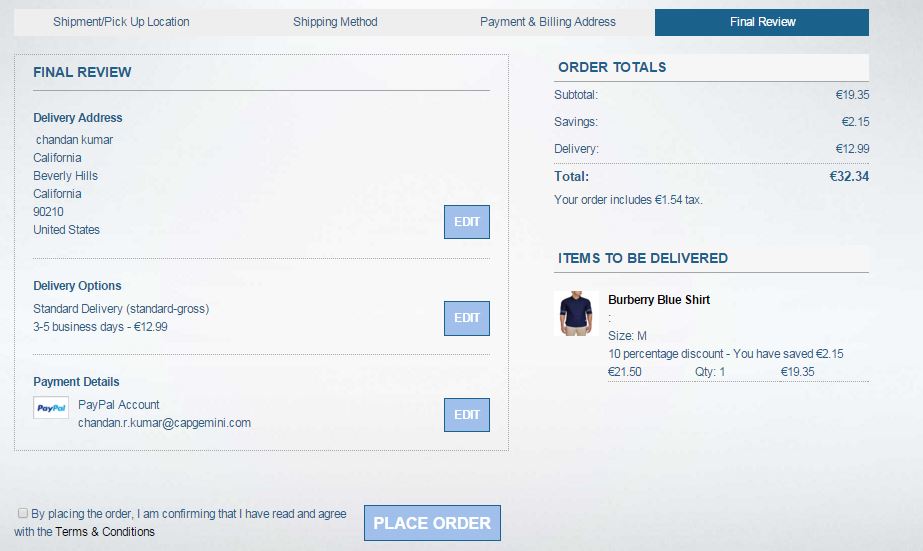
Click on checkout with PayPal , It redirects to the login page of Paypal. With valid credentials, you should login:



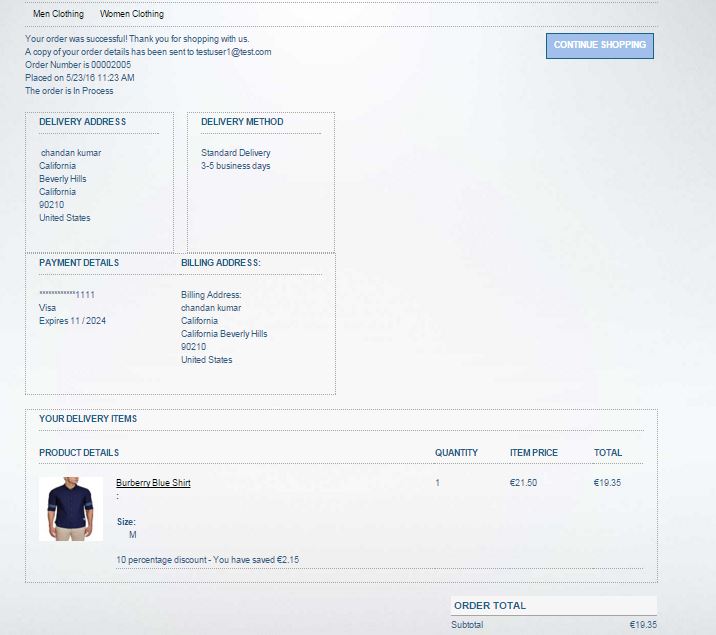
Continue the order placing process from PayPal with verifying your information:



It will redirect to merchant burberry page with payment details as PayPal, continue to add PayPal card details by click on Edit tab for Payment Details:

****

Select pay with card and enter the needed details to proceeds and place the order, It will place order successfully as show in below image:

****