

## Explain ImpEx?

Integrated text-based Import / Export extension is called impex.

ImpEx engine allows creating, updating, removing and exporting platform items such as customer, product or order data **to & from** comma-separated value (CSV) data files, both during run time and during the initialization or update process.

Hybris provides extension called "**Impex**" inside **platform/ext** folder.

**Impex import** = Always data flows into Hybris system. **Impex export** = Always data flows out of Hybris system.

### Key Features:

- ✓ Import & Export hybris Commerce Suite data from and into CSV files.
- ✓ To feed data into DB we go for impex.
- ✓ Impex file is developed based on the "**BeanShell Script**".
- ✓ **ImpEx extension** is a **converter** between items in hybris & CSV file, and other way round as show: -



By using Impex extension it is easy to:

- ✓ Update data at run time
- ✓ Creating initial data for project / migrating existing data from one hybris into another.

**Q:** Why do we need to Import & Export?

- ✓ Any ecommerce **site** will have a **data** like products, customer, regions, countries, promotions etc...
- ✓ All this data should be inserted into **Hybris system** to display it on the **site**.
- ✓ Import data **multiple times** into Hybris System.
- ✓ To **insert** this kind of data into hybris system, we go for **Impex Import**.
- ✓ We can **insert data** either at the **run time** or **during initialization** or **during update** process.
- ✓ Whenever we need to take the backup of data, we have to **export** data into csv files so that later it can be imported into Hybris system.
- ✓ If we are doing initialization, it is recommended to **take data backup** using **Impex Export**.
- ✓ Once the initialization is done, we can import the data back to hybris system.

**Q:** What are different types of Data Languages?

DML (Data Manipulation Language) = select, insert, update, delete etc.

DDL (Data Definition Language) = create, alter, drop, truncate etc.

DCL (Data Control Language) = grant, revoke

TCL (Transaction Control Language) = commit, rollback, savepoint etc.

**Note:** - Impex is for Load data / Take a backup.

**items.xml** -- Used for DDL (Table Creation). **impex** --- Load data (DML). **Flexible Search** – Select.

**Q:** How to insert 100 Records?

```
insert into tablename(Col1, Col2...) values (va1, va2...);  
Insert into tablename(Col1, Col2...) values (va1, va2...);  
....100 times....
```

**In Hybris =**

```
insert_update itemtype;att1;att2;...;10;20;...; (This is Syntax if att is automatic type)  
insert_update itemtype;att1;att2;...;1,2,3;4,5,6;...; (This is Syntax if att is collection type)  
insert_update itemtype;att1[lang=en];att1[lang=hin];att2;...;product;hello;abc...;(This is the  
syntax if att1 is multiple language support)
```

**Note:** - Hybris allows us to change the delimiters (; / , / # / ...)

**Q:** What are the Terminologies (or) Jargons used in ImpEx?

✓ **Headers** in header section =

```
mode type[modifier=value];attribute[modifier=value];attribute[modifier=value];attribute[modifier=value]  
[;...];attribute[modifier=value]  
INSERT_UPDATE category;code[unique=true];name[lang=de];name[lang=en];$superCategories;$thumbnail;description[lang=de];order
```

MODE      Item Type      Attributes      Modifier

- ✓ **Mode** = Tells what is to be done with the following value lines (insert, update, and so on)
  - **INSERT** = Used to insert record into DB. If record already exists it throws an exception.
  - **UPDATE** = Used to update existing record. It throws exception if record does not exist.
  - **INSERT\_UPDATE** = Most preferred 1 for insertion or update. There won't be exception.
  - **REMOVE** = Used to remove existing records. We need to specify 1 unique attribute to remove the record.
- ✓ **Type / ItemType** = Defines type of item to be processed (Category, Product, Media and so on)
- ✓ **Attribute** = Defines attributes of the **item type** which are defined in **items.xml** file. Each attribute **specified** will get the **values affected** on corresponding columns in DB.
- ✓ **Modifier** = Used to specify **additional info** for the attributes like specifying attribute uniqueness using unique=true.

**Q:** How can we specify multiple modifiers?

**1<sup>st</sup> way** (Comma Separated) = attribute[modifier1=value1,modifier2=value2,modifier3=value3];

**2<sup>nd</sup> way** (Separate bracket []) = ;attribute[modifier1=value1][modifier2=value2];

**Note:** - Header is applicable for all the value lines till next header appears in the file. 1 file can have any number of headers.

✓ **Lines of values** value line section = There could be **1/ more** value lines in the file.

Each value line represents **one row of data** in table.

**Each field** in the value line is separated by **semi colon** and represents the data into **one column**.

**Note:** - Semi colon should not be there for last field. Space is also considered as value of the field.

Each line should have same number of fields. If field contains any line breaks or quotes, then enclose such fields in double quotes.

- ✓ **Comments** = A comment starts with hash symbol #.

```
#Inserting product details This will be ignored  
INSERT_UPDATE Product;code[unique=true];varianttype(code);name[lang=en]
```

- ✓ **Macro** definitions = Used to define the value at one place and use the macro in all the places.  
How to define max size for stack in C? = #define MAX 10000000. Macro is more like constant.  
It's preprocessor. Used to distinguish lines which need preprocessing before actual compilation begins.  
It helps to eliminate writing same lines again & again.  
Advantage with macro? == If you change 1, it replaces at run time all the times during runtime.  
Macros starts with "\$" and they are referenced by \$macroName.  
Macro must be first statement in impex file. We can have any number of macros.

\$catalog=catalog(id) □ Defining macro called Catalog.

\$catalogVersion=catalogVersion(\$catalog,version) □ Defining macro called "catalogVersion".

INSERT Product; code; \$catalogVersion --- Header

Wherever term \$catalogVersion appears, it is replaced by: catalogVersion(catalog(id),version)

**Q:** Explain how data feeding can be done?

- ✓ **1<sup>st</sup> way = Through HAC**
  - hMC (<http://localhost:9001>) □ Console □ ImpEx Import □ Copy the ImpEx Statement □ Validate □ Impex Import
- ✓ **2<sup>nd</sup> way = Through HMC**
  - HMC (<http://localhost:9001/hmc/hybris>) -- System -- Extract System -- Tools -- Import -- New Window is opened -- Import File -- New window opened -- File -- Select Impex file -- Upload File -- Start -- Your record is successfully uploaded.
- ✓ **3<sup>rd</sup> way = Programmatic Way / ImpEx approach**
  - We can configure the impex's to load during Initialization and update process through code.

**Q:** Explain Convention over Configuration

The **Convention over Configuration** principal is adopted in SAP Hybris Commerce to simplify and reduce the need for writing configuration files. As a result, **ImpEx** files for **essential & project** data can be prepared without the need for additional data configuration.

During the **initialization & update** = Platform looks for **ImpEx files** in the <extension\_name> /resources/impex folder. **In particular:**

**For Essential Data** = platform scans <extension\_name> /resources/impex folders for files with names that match pattern **essentialdata\*.impex** & imports files during essential data creation.

**For Project Data** = Platform scans <extension\_name> /resources/impex folders for files with names that match pattern **projectdata\*.impex** & imports files during the project data creation.

The **ImpEx directory** does not exist by default. You must create it and copy files to it.

**Example:** - Let's say we have the following folder structure:

resources/impex/essentialdataOne.impex

resources/impex/DataOne.csv

resources/impex/essDataOne.impex

resources/impex/projectdataOne.impex

Files located in **essentialdataOne.impex** & **projectdataOne.impex** are imported during the essential and project data imports respectively.

**Q:** What is Type & Process?

TYPE	DESCRIPTION
ESSENTIAL	Configured methods will be executed only during essential data creation process
PROJECT	Configured methods will be executed only during project data creation process
ALL	This is the default type which means Configured methods will be executed during both essential data and project data creation process

PROCESS	DESCRIPTION
INIT	Configured methods will be executed only during initialization process
UPDATE	Configured methods will be executed only during Update process
ALL	This is the default process which means Configured methods will be executed during both Initialization and Update process

**Note:** - Hybris Provides a class "**CoreSystemSetup.java** & **InitialDataSystemSetup.java**" to achieve this.

**Q:** What are Steps to configure an impex to **import** during initialization / Update process (or how control what an extension is doing during the initialization and update process)?

**Step 1** = Create a class & **annotate with @SystemSetup** & Create public methods & **annotate with @SystemSetup** by specifying "**Type**" and "**Process**".

```
@SystemSetup(extension = ChennaTrainingConstants.EXTENSIONNAME)
public class CoreSystemSetup extends AbstractSystemSetup
{
    public static final String IMPORT_ACCESS_RIGHTS = "accessRights";

    @SystemSetup(type = Type.ESSENTIAL, process = Process.ALL)
    public void createEssentialData(final SystemSetupContext context)
    {
        importImpexFile(context, "/ChennaTraining/import/common/essential-data.impex");
        importImpexFile(context, "/ChennaTraining/import/common/countries.impex");
        importImpexFile(context, "/ChennaTraining/import/common/delivery-modes.impex");
        importImpexFile(context, "/ChennaTraining/import/common/themes.impex");
        importImpexFile(context, "/ChennaTraining/import/common/user-groups.impex");
    }

    @SystemSetup(type = Type.PROJECT, process = Process.ALL)
    public void createProjectData(final SystemSetupContext context)
    {
        final boolean importAccessRights = getBooleanSystemSetupParameter(context, IMPORT_ACCESS_RIGHTS);
        final List<String> extensionNames = getExtensionNames();
        if (importAccessRights && extensionNames.contains("cmscockpit"))
        {
            importImpexFile(context, "/ChennaTraining/import/cockpits/cmscockpit/cmscockpit-users.impex");
            importImpexFile(context, "/ChennaTraining/import/cockpits/cmscockpit/cmscockpit-access-rights.impex");
        }
    }
}
```

**Note:** - In above **both** methods will be executed during **Initialization** process but 1 will be executed during **essential data** creation process. Other will be executed during **project data** creation process. We passed a parameter to a method of type **SystemSetupContext** which can be used to get some additional details like **user selected** parameter value.

**Step 2** = **Register** the class which is annotated with **@SystemSetup** as a spring bean  
hybris\bin\custom\training\chennatraining\resources\chennatraining-spring.xml

```
<bean id="acceleratorCoreSystemSetup" class="org.training.core.setup.CoreSystemSetup" parent="abstractCoreSystemSetup"/>
```

**Q:** How to remove all data (all records) from the itemtype?

```
$item=TestItem
REMOVE $item[batchmode=true]; itemtype(code)[unique=true]
;$item
```

```
REMOVE Employee[batchmode=true]; itemtype(code)[unique=true]
;Employee
```

**Q:** Explain conditional Impex?

```
$var=value
-----
INSERT_UPDATE TestItem;string[unique=true];integer
"## if: ""$var"".equals("""value""")
"key";5
"## endif:
```

**Note:** - In ImpEx every line is **auto commit**. Let's say an ImpEx is having 10000 lines of code. 7000 lines executed. after that 1 line is having error. By default, after that statements will not be executed.

**Q:** Where to create ImpEx? = **(1)** In Core – chennatrainingcore **(2)** In Initialdata – chennatraininginitialdata

**Q:** How to take backup?. We created 1000 products & want to take backup?

Go to ImpEx -- Select the Header (Select macro if any) -- hAC -- ImpEx Export, paste it -- Validate Content -- Export Content, Download the Zip file (It will have all). CVS File Contain the data. \*.impex file will have the header.

**Note:** - We can write all the ImpEx (Categories, Products, Images...) in single file. But that is not best practice. For Readability use separate ImpEx files.

**Q:** Impex object methods?

- ✓ impex.setCurrentHeader(HeaderDescriptor).
- impex.setCurrentHeader(String)
- Used to replace current header.
- Eg-** U can conditionally use INSERT/UPDATE modes depending on data
- ✓ impex.setValueLinesToSkip(int) = How many value lines should be skipped.
- ✓ impex.getLastImportedItem() = Returns the last imported item (of Item type)
- ✓ impex.isSecondPass() = If csv data arsed again for resolving previously unresolvable item references.
- ✓ impex.discardNextLine() = Next line won't be processed.
- ✓ impex.includeSQLData(url,user,password,JDBCdriver,sql) = Initialize parameters & fires an SQL with DB.
- ✓ impex.includeSQLData(sql) = The alternative way of firing a statement. To use with initDatabase(..).
- ✓ impex.includeExternalData(file, encoding, lineToSkip) = Reads values from the file.
- ✓ impex.getCurrentLocation() =Returns current line number.
- ✓ impex.insertLine(csvLine) = Inserts a line into a parsing stream (it will be a next processing line)
- ✓ impex.addDefinition(definition) = Creates/modifies a macro. Definition may look like "\$a=33".
- ✓ impex.setRelaxedMode() = The same as setValidationMode("import\_relaxed")
- ✓ impex.info(msg) = Shows a debug message in the console (log.info)
- ✓ impex.warn(msg) = Shows a debug message in the console (log.warn)
- ✓ impex.error(msg) = Shows a debug message in the console (log.error)

**Q:** What are the main ways in hybris to update data from the external sources?.

- ✓ **IMPEX. HotFolder & Import Cockpit** are also in this category, because Spring Integration and Import Cockpit convert CSVs to IMPEXes.
- ✓ hybris API (modelService). This approach is to create your own importers with the custom logic.

## Q: Explain Hot folders =

SAP Hybris provides different ways to import data, one of them is a hot folder.

A hot folder means that when you transfer a file into a specific folder with matching name it will automatically be processed by SAP Hybris. The hot folder functionality is located inside the accelerator services extension, it's based on the Spring Integration Framework.

## Q: What are the actions executed when you transfer a file into a hot folder? =

- ✓ An inbound channel detects a new file matching a pattern (file:inboundchannel-adapter).
- ✓ An outbound channel moves the file to a new folder (file:outbound-gateway).
- ✓ activators setup the headers (int:service-activator).
- ✓ Resolve mapping between the file name and the type to import.
- ✓ Generate the impex.
- ✓ Run the impex.
- ✓ Cleanup

## Q: Explain different modes of ImpEx

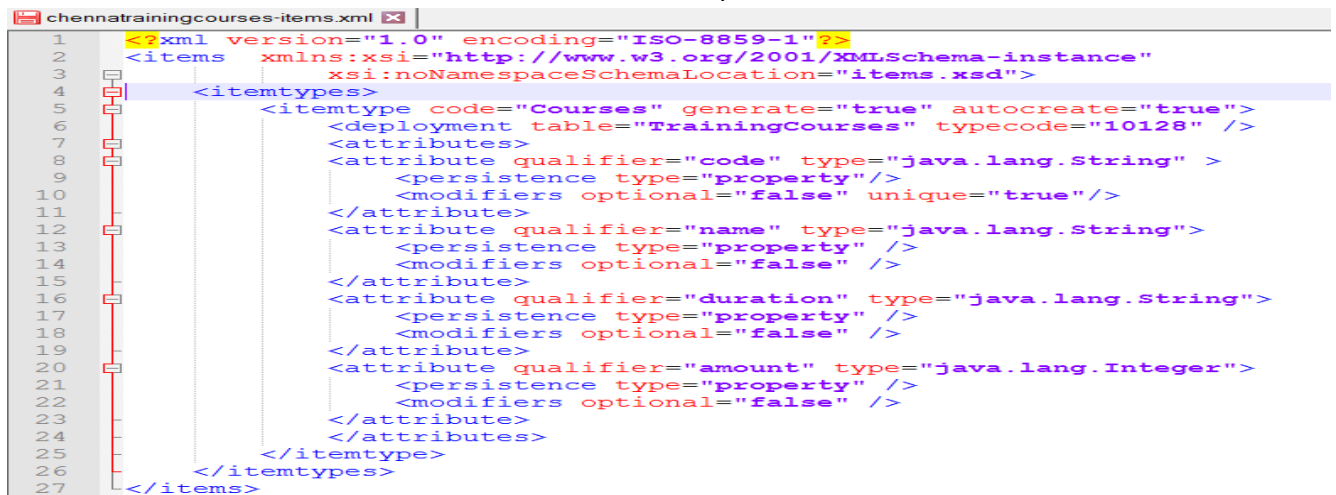
ImpEx import has 2 options –

**Import Content** = Copy the script & and import the data

**Import Script**= Select the impex file & import the data

Let's insert records into a table called "TrainingCourses" which has itemtype ="Courses"

Here is the **items.xml** file and for these attributes write the impex with different modes: -



```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <items xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:noNamespaceSchemaLocation="items.xsd">
4     <itemtypes>
5         <itemtype code="Courses" generate="true" autocreate="true">
6             <deployment table="TrainingCourses" typecode="10128" />
7             <attributes>
8                 <attribute qualifier="code" type="java.lang.String" >
9                     <persistence type="property" />
10                    <modifiers optional="false" unique="true" />
11                </attribute>
12                <attribute qualifier="name" type="java.lang.String">
13                    <persistence type="property" />
14                    <modifiers optional="false" />
15                </attribute>
16                <attribute qualifier="duration" type="java.lang.String">
17                    <persistence type="property" />
18                    <modifiers optional="false" />
19                </attribute>
20                <attribute qualifier="amount" type="java.lang.Integer">
21                    <persistence type="property" />
22                    <modifiers optional="false" />
23                </attribute>
24            </attributes>
25        </itemtype>
26    </itemtypes>
27 </items>
```

- ✓ **INSERT** = Used to insert the records. If record exists, it will throw an error & you need to correct the error.

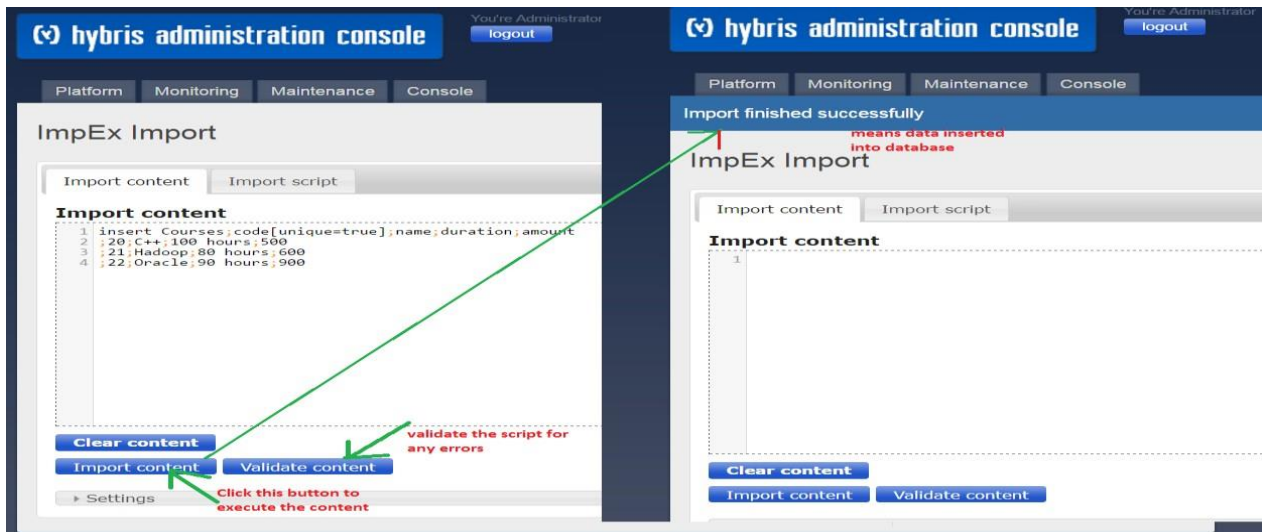
**Option1** = Insert the data into Courses Item Type using Import Content

**Step 1** = Goto <https://localhost:9002> ☐ Console ☐ ImpEx Import ☐ Import Content tab & past the script

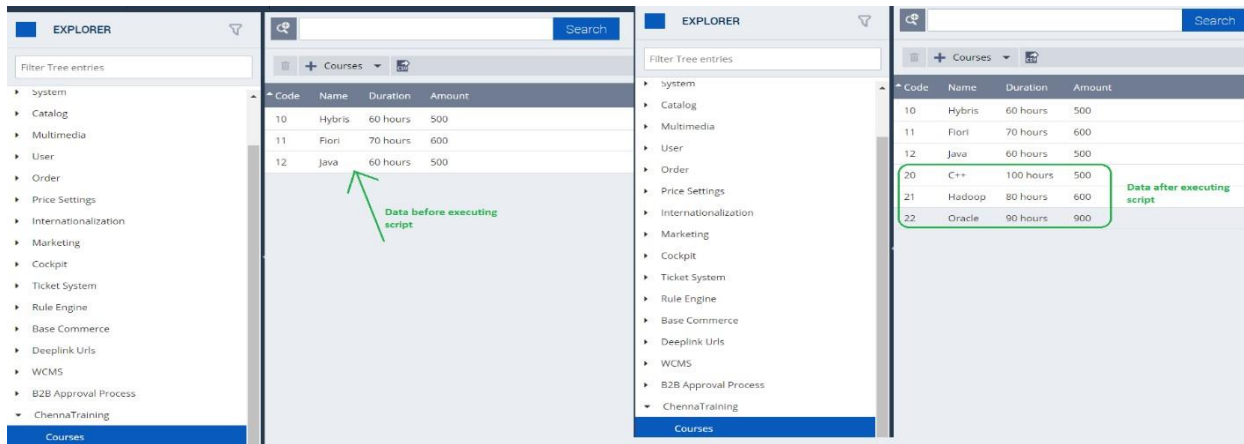
### Script-

```
insert Courses;code[unique=true];name;duration;amount
;20;C++;100 hours;500
;21;Hadoop;80 hours;600
;22;Oracle;90 hours;900
```

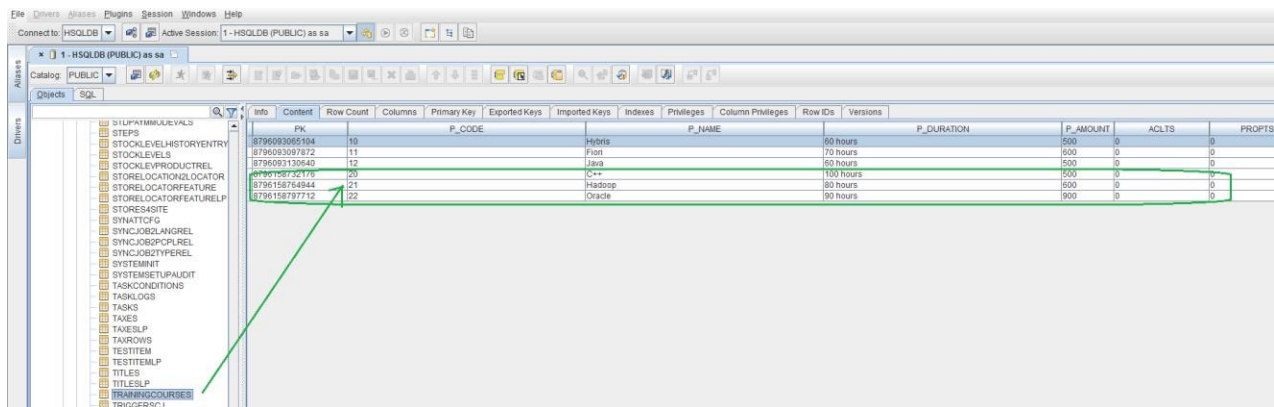




**Step 2 = Goto back office and verify the data**



**Step 3 = Goto DB & Verify data (If you already opened Squirrel then close & reopen to see data changes).**



**Option 2 = Insert the data into Courses Item Type using Import Script**

**Step 1 = Goto <https://localhost:9002> ☐ Console ☐ ImpEx Import ☐ Import Script tab & select impex file**





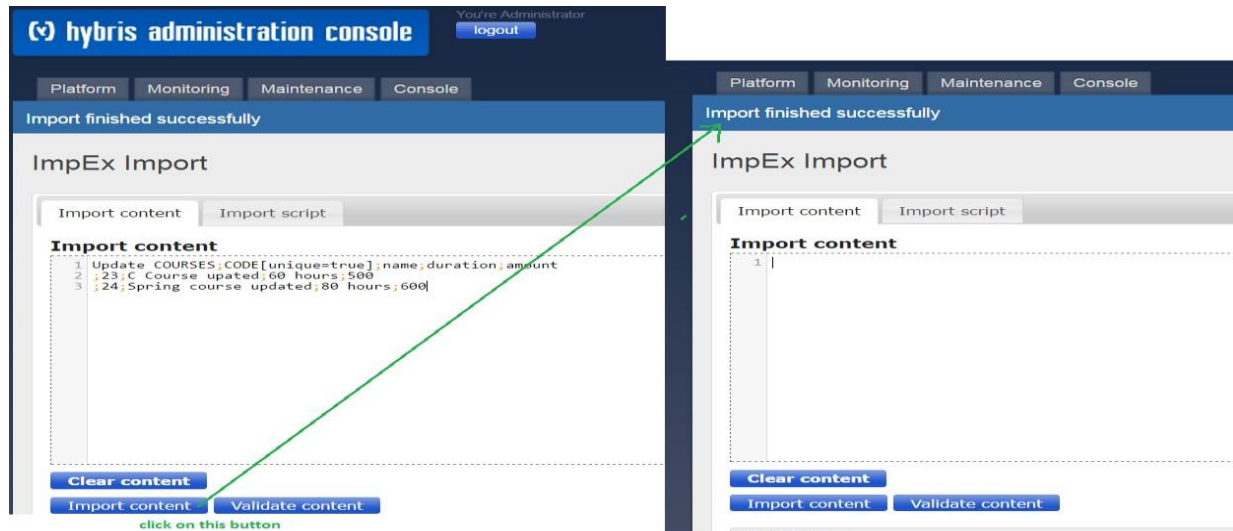
✓ **UPDATE** = Used to update records. If record not exists, it will throw an error & you need to correct error

**Step 1** = Goto <https://localhost:9002> □ Console □ ImpEx Import □ Import Script tab & select impex file  
**Script –**

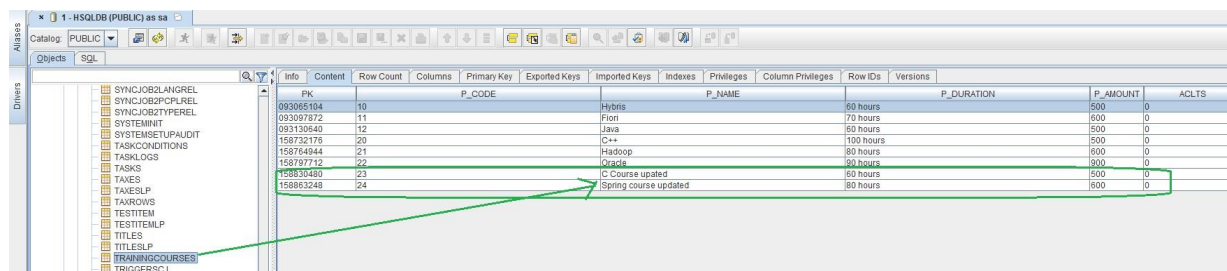
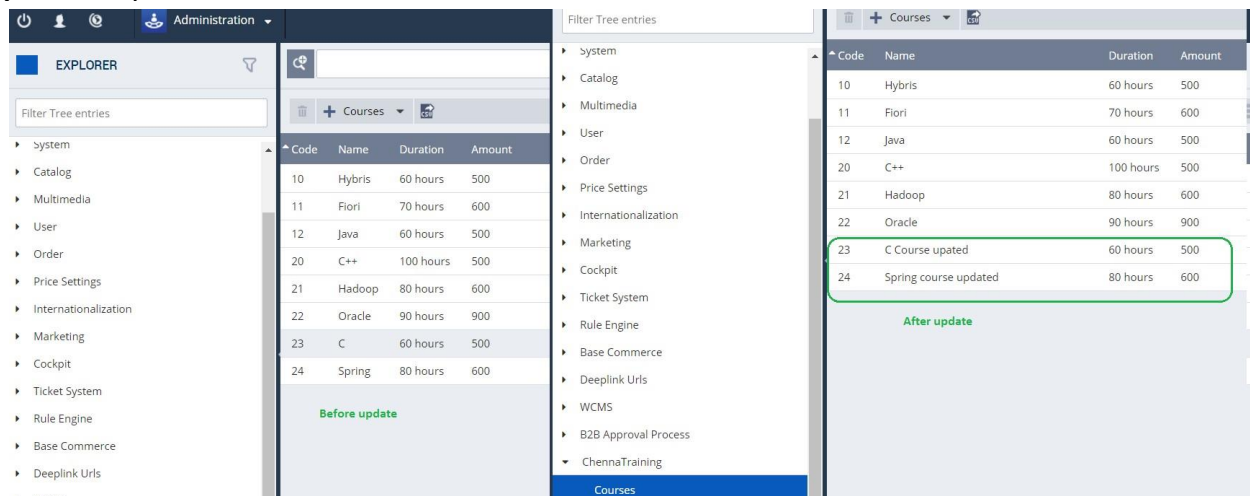
Update COURSES;CODE[unique=true];name;duration;amount

;23;C Course updated;60 hours;500

;24;Spring course updated;80 hours;600



**Step 2** = Verify the data in back office & in DB.



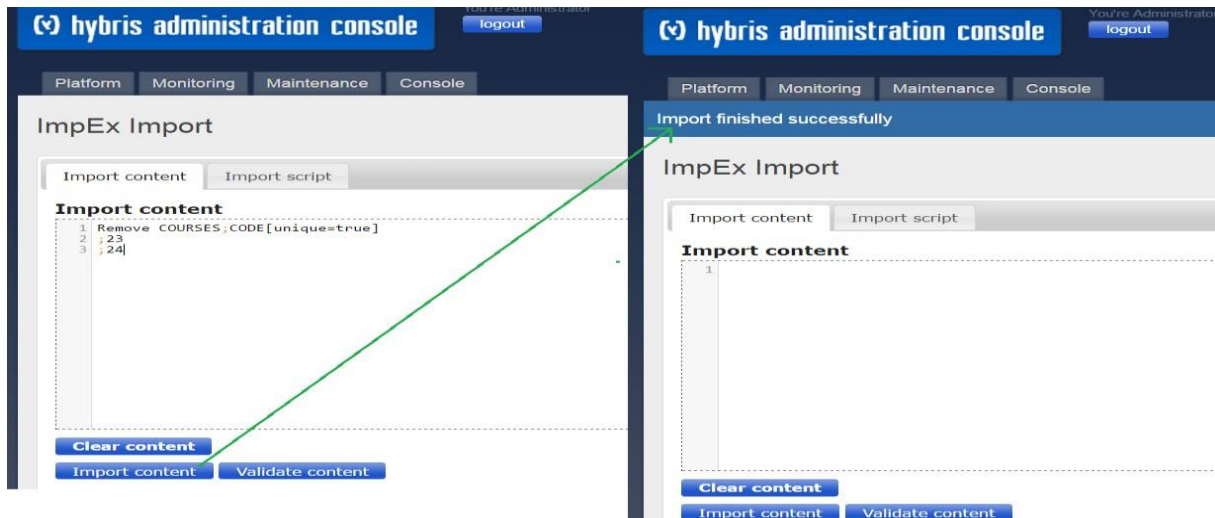
- ✓ **REMOVE** = Used to delete records. If record exists it will delete, otherwise it won't delete any records & no error message.

**Step 1** = Goto <https://localhost:9002> □ Console □ ImpEx Import □ Import Script tab & select impex file  
Script –

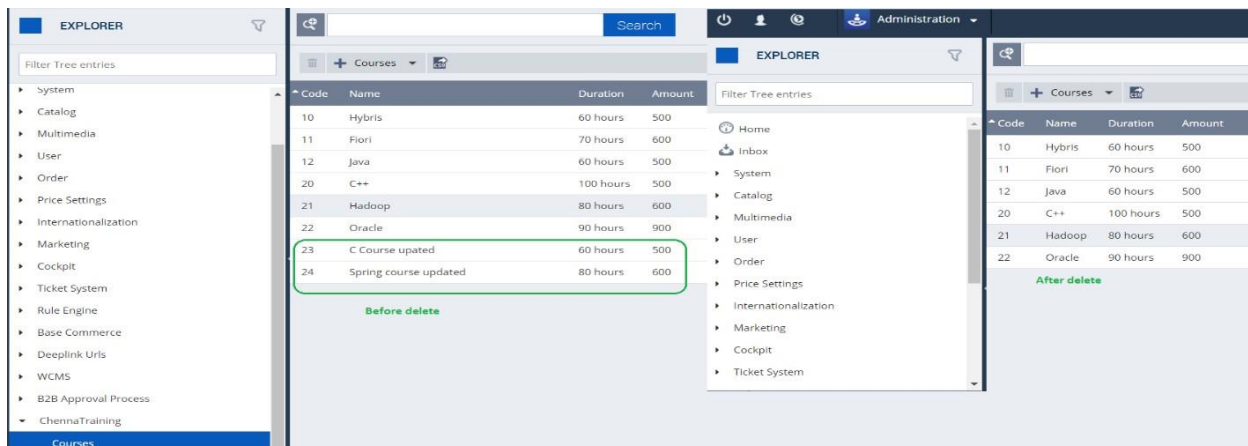
Remove Courses;code[unique=true]

;23

;24



**Step 2** = Verify the data in back office and in database



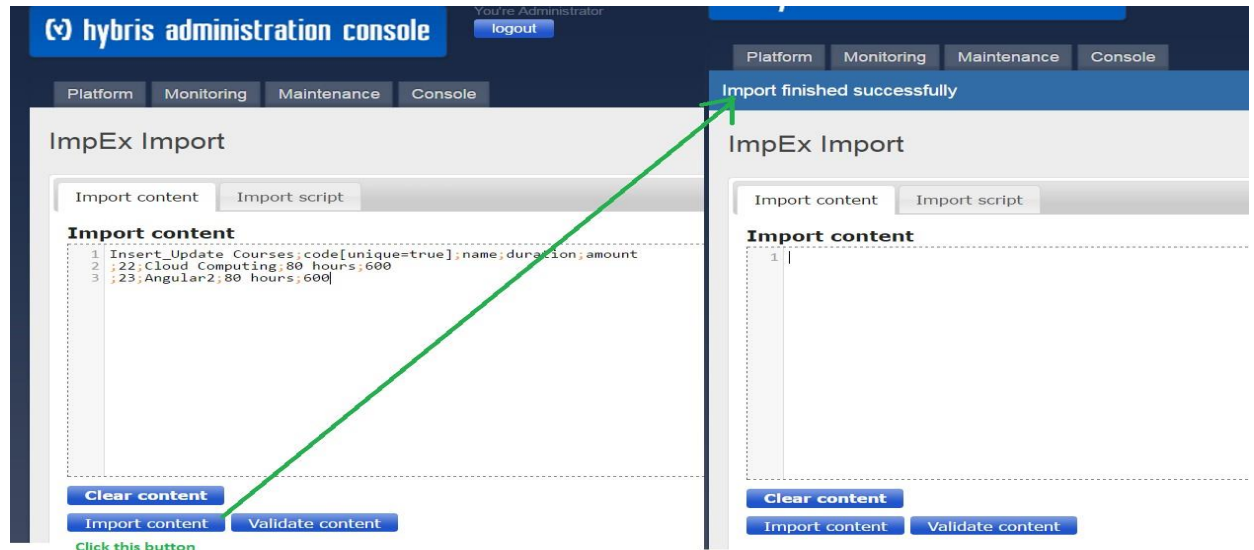
Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Versions
TYPEPKSTRING		OWNERPKSTRING	PK		P_CODE				P_NAME	
8796129394770	<null>	8796093095104	10		Hybris				60 hours	500
8796129394770	<null>	8796093097872	11		Flori				70 hours	600
8796129394770	<null>	8796093130540	12		Java				60 hours	500
8796129394770	<null>	8796158732176	20		C++				100 hours	500
8796129394770	<null>	8796158797712	22		Oracle				90 hours	900
8796129394770	<null>	8796158895016	21		Hadoop				80 hours	600

- ✓ **INSERT\_UPDATE** = Used to insert/update records. If record already exists, it will update otherwise it will insert the record

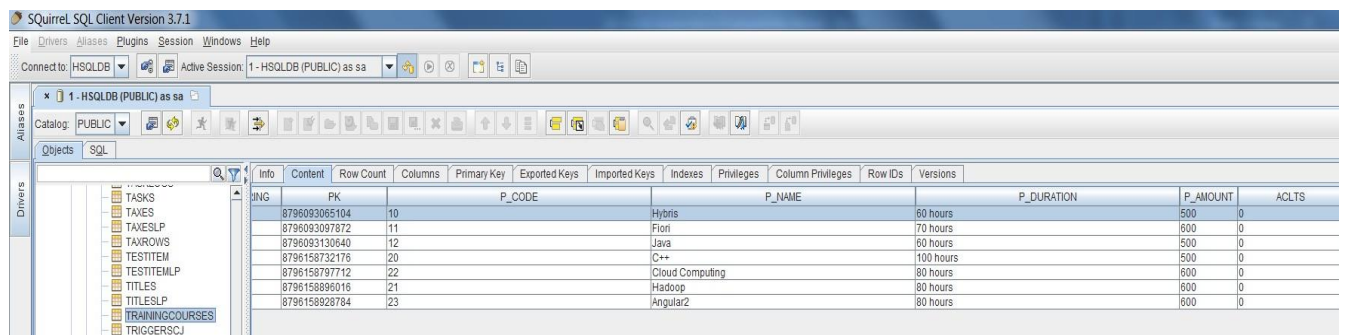
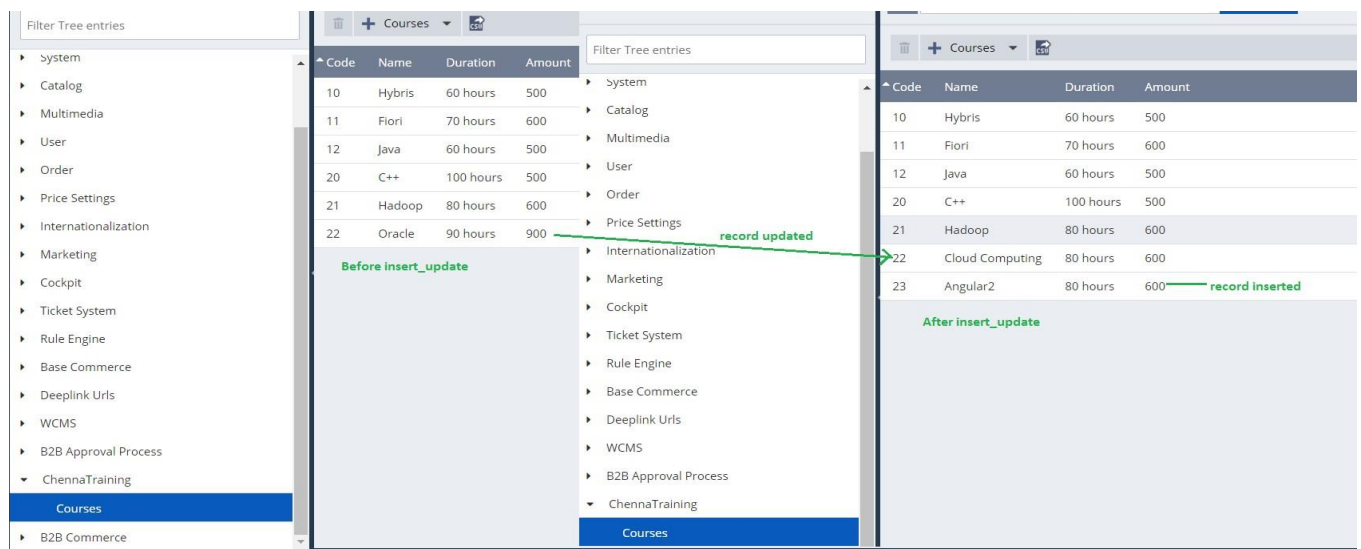
**Step 1** = Goto <https://localhost:9002> □ Console □ ImpEx Import □ Import Script tab & select impex file  
Script –

Insert\_Update Courses;code[unique=true];name;duration;amount

;22;Cloud Computing;80 hours;600  
;23;Angular2;80 hours;600



Step 2 = Verify the data in back office & in DB



## Q = Explain catalog impex?

**Step 1** = Copy the below script into a text file and save it as “**catalog.impex**” in some folder

```
# Import the Product Catalog and Classification Catalog
$productCatalog=ChennaTrainingProductCatalog
$classificationCatalog=ChennaTrainingClassificationCatalog
$languages=ja,en,de,zh

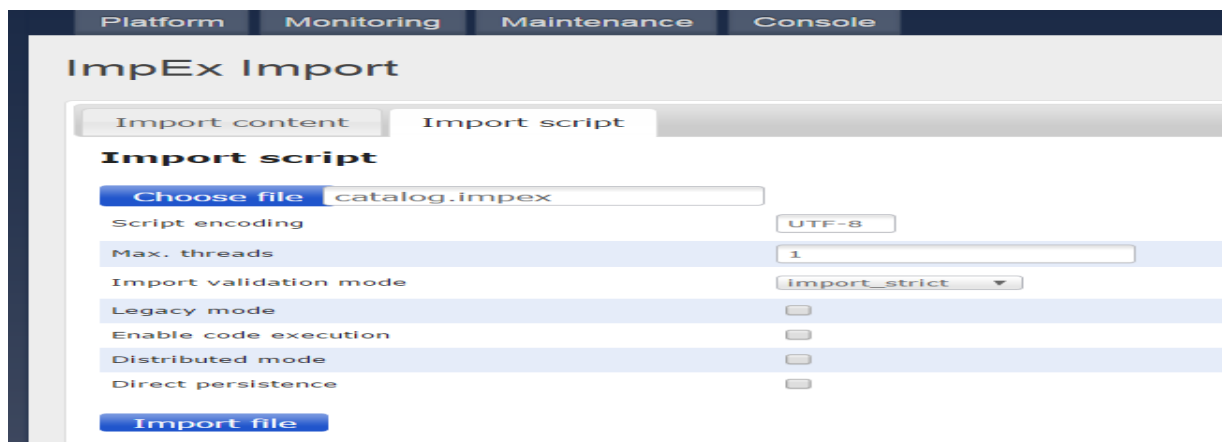
# Product catalog
INSERT_UPDATE Catalog;id[unique=true]
;$productCatalog

# Classification catalog
INSERT_UPDATE ClassificationSystem;id[unique=true]
;$classificationCatalog

# Product versions for product catalogs
INSERT_UPDATE
CatalogVersion;catalog(id)[unique=true];version[unique=true];active;languages(isoCode);readPrincipal
s(uid)
;$productCatalog;Staged;false;$languages;employeeegroup
;$productCatalog;Online>true;$languages;employeeegroup

# Insert Classifications System Version
INSERT_UPDATE
ClassificationSystemVersion;catalog(id)[unique=true];version[unique=true];active;inclPacking[virtual=tr
ue,default=true];inclDuty[virtual=true,default=true];inclFreight[virtual=true,default=true];inclAssuranc
e[virtual=true,default=true]
;$classificationCatalog;1.0>true
```

**Step 2** = Goto <https://localhost:9002> ☐ Console ☐ ImpEx Import ☐ Import Script tab & select the catalog.impex file & click on import file: -



The screenshot shows the 'ImpEx Import' interface with the 'Import script' tab selected. The 'Choose file' field contains 'catalog.impex'. The 'Script encoding' is set to 'UTF-8'. The 'Max. threads' is set to '1'. The 'Import validation mode' is set to 'import\_strict'. The 'Legacy mode', 'Enable code execution', 'Distributed mode', and 'Direct persistence' checkboxes are all unchecked. An 'Import file' button is at the bottom.

Field	Value
Choose file	catalog.impex
Script encoding	UTF-8
Max. threads	1
Import validation mode	import_strict
Legacy mode	<input type="checkbox"/>
Enable code execution	<input type="checkbox"/>
Distributed mode	<input type="checkbox"/>
Direct persistence	<input type="checkbox"/>

### Step 3 = See the the data changes in back office and database

Administration Explorer - Catalogs

ID	Name
ChennaTrainingClassificationCatalog	Null
ChennaTrainingProductCatalog	Null
apparel-deContentCatalog	Apparel DE Content Catalog
apparel-ukContentCatalog	Apparel UK Content Catalog
apparelProductCatalog	Apparel Product Catalog
electronicsContentCatalog	Electronics Content Catalog
ElectronicsClassification	Null
electronicsProductCatalog	Electronics Product Catalog
powertoolsContentCatalog	Powertools Content Catalog
PowertoolsClassification	Null
powertoolsProductCatalog	Powertools Product Catalog
Default	Default-Catalog

SQL Server Enterprise Manager - Catalogs Table

P_ID	P_ACTIVECATALOGVER	P_DEFAULTCAT	P_PREVIEWURLTEMPLATE	P_URLPATTERNS	P_SUPPLIER
Default	879609305577	0	/storefoundation/index.jsf	<null>	<null>
powertoolsProductCatalog	8796093153881	0	/storefoundation/index.jsf	<null>	<null>
PowertoolsClassification	8796093186649	0	/storefoundation/index.jsf	<null>	<null>
powertoolsContentCatalog	8796093252185	0	/storefoundation/index.jsf	<null>	<null>
electronicsProductCatalog	8796093317721	0	/storefoundation/index.jsf	<null>	<null>
ElectronicsClassification	8796093350489	0	/storefoundation/index.jsf	<null>	<null>
electronicsContentCatalog	8796093416025	0	/storefoundation/index.jsf	<null>	<null>
apparel-ukContentCatalog	8796093547097	0	/storefoundation/index.jsf	<null>	<null>
apparelProductCatalog	8796093481551	0	/storefoundation/index.jsf	<null>	<null>
apparel-deContentCatalog	8796093612633	0	/storefoundation/index.jsf	<null>	<null>
ChennaTrainingProductCatalog	8796158621445	0	/storefoundation/index.jsf	<null>	<null>
ChennaTrainingClassificationCatalog	8796158689881	0	/storefoundation/index.jsf	<null>	<null>

populated because of the default value defined in the catalog-items.xml file

```
<attribute qualifier="previewURLTemplate" type="java.lang.String">  
  <modifiers read="true" write="true" search="true" optional="true"/>  
  <persistence type="property"/>  
  <defaultvalue>/storefoundation/index.jsf</defaultvalue>  
</attribute>
```

Administration Explorer - Catalog Versions

Catalog	Catalog Version	is active catalog version
ChennaTrainingClassificationCatalog	1.0	true
ChennaTrainingProductCatalog	Online	true
ChennaTrainingProductCatalog	Staged	false
Apparel DE Content Catalog	Online	true
Apparel DE Content Catalog	Staged	false
Apparel UK Content Catalog	Online	true
Apparel UK Content Catalog	Staged	false
Apparel Product Catalog	Online	true
Apparel Product Catalog	Staged	false
Electronics Content Catalog	Online	true
Electronics Content Catalog	Staged	false
ElectronicsClassification	1.0	true
Electronics Product Catalog	Online	true
Electronics Product Catalog	Staged	false



## Note – Observe the default values

Catalogs

Catalog Versions

Categories

Products

Product Variant Types

Units

Keywords

Classification Systems

Classifying Categories

Feature List

Feature Values

Classification Units

Multimedia

User

Order

Price Settings

ChennaTrainingClassificationCatalog	5.0	false
ChennaTrainingProductCatalog	Online	true
ChennaTrainingProductCatalog	Staged	false
Apparel DE Content Catalog	Online	true
Apparel DE Content Catalog	Staged	false
Apparel UK Content Catalog	Online	true

ChennaTrainingProductCatalog : Online

Agreements

Incl. freight: ☐ True ☒ False

Incl. packing: ☐ True ☒ False

Incl. assurance: ☐ True ☒ False

Incl. duty: ☐ True ☒ False

Catalogs

Catalog Versions

Categories

Products

Product Variant Types

Units

Keywords

Classification Systems

Classifying Categories

Feature List

Feature Values

Classification Units

Multimedia

User

Order

Price Settings

ChennaTrainingClassificationCatalog	5.0	false
ChennaTrainingProductCatalog	Online	true
ChennaTrainingProductCatalog	Staged	false
Apparel DE Content Catalog	Online	true
Apparel DE Content Catalog	Staged	false
Apparel UK Content Catalog	Online	true

ChennaTrainingClassificationCatalog : 5.0

Agreements

Note - The values set to true because the script defined in the impex file

Incl. freight: ☒ True ☐ False

Incl. packing: ☒ True ☐ False

Incl. assurance: ☒ True ☐ False

Incl. duty: ☒ True ☐ False

1 - HSQldb (PUBLIC) as sa

Objects

SQL

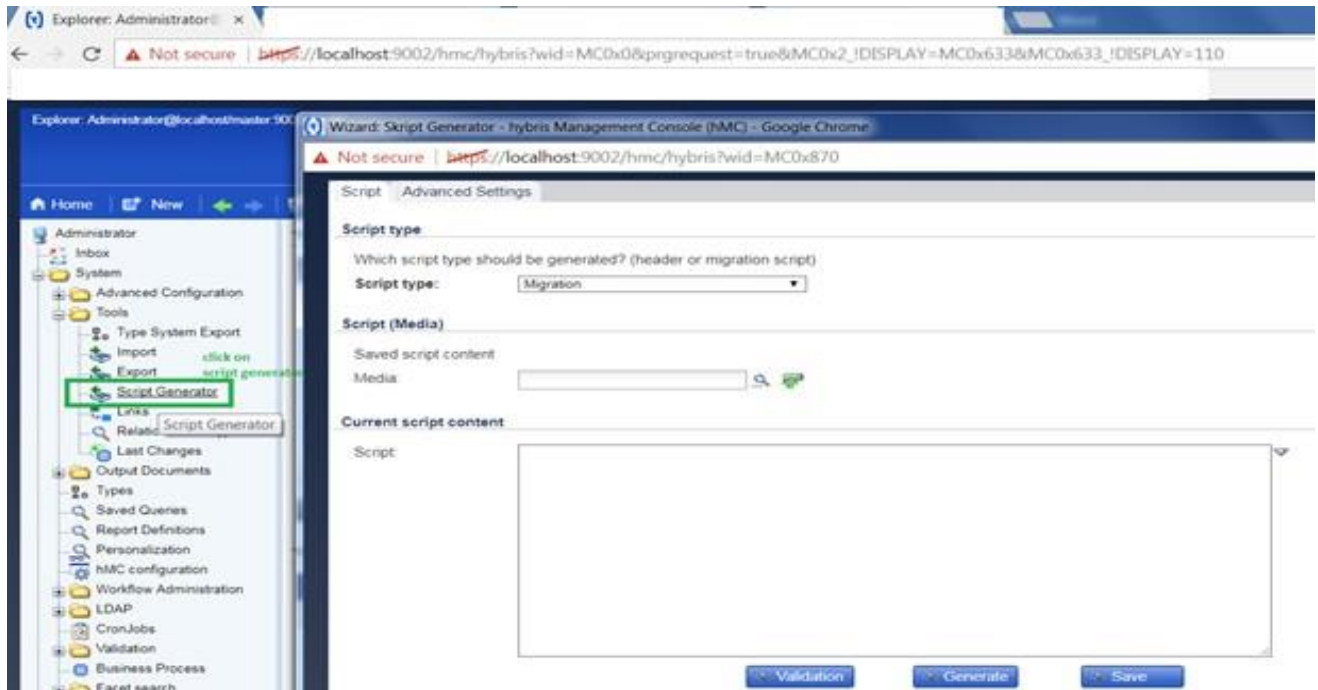
Info	Content	Row Count	Columns	Primary Key	Exported Keys	Imported Keys	Indexes	Privileges	Column Privileges	Row IDs	Versions
	P_PRODUCT		P_PG	P_PRODUCTMATCHQUALI...	P_STARTTIME	P_ENDTIME	P_USER	P_UG	P_USERMATCHQUALI...		
		8796120023131	8796120023131					8796119859291	8796119859291		
		8796120055899	8796120055899					8796119892059	8796119892059		
		8796120023131	8796120023131					8796119892059	8796119892059		
		8796120055899	8796120055899					8796119892059	8796119892059		
		8796119924827	8796119924827					8796119793755	8796119793755		
		8796119924827	8796119924827					8796119793755	8796119793755		
		8796119957595	8796119957595					8796119760987	8796119760987		
		8796119990363	8796119990363					8796119760987	8796119760987		
		8796119957595	8796119957595					8796119826523	8796119826523		
		8796119990363	8796119990363					8796119826523	8796119826523		
		8796119957595	8796119957595					8796119859291	8796119859291		
		8796119957595	8796119957595					8796119892059	8796119892059		
		8796119957595	8796119957595					8796119760987	8796119760987		
		8796119990363	8796119990363					8796119760987	8796119760987		
		8796119957595	8796119957595					8796119826523	8796119826523		
		8796119990363	8796119990363					8796119859291	8796119859291		
		8796119957595	8796119957595					8796119826523	8796119826523		
		8796119957595	8796119957595					8796119859291	8796119859291		
		8796119957595	8796119957595					8796119892059	8796119892059		
		8796119924827	8796119924827					8796119793755	8796119793755		



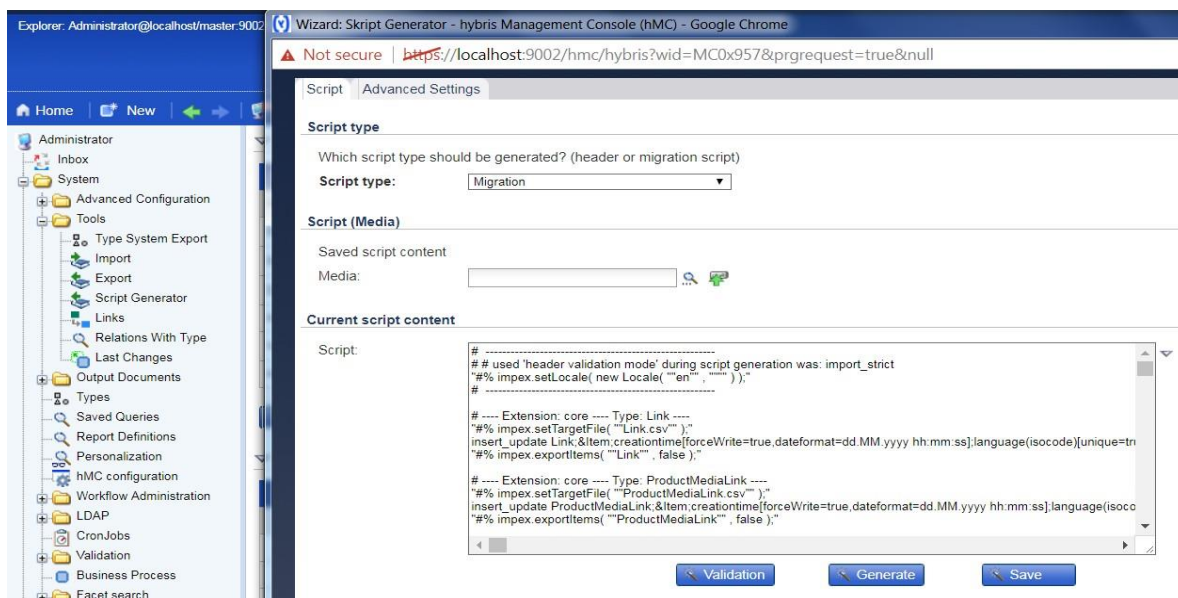
**Q:** How to generate impex script and corresponding data for one table?

**Scenario** – Let say there is new table created and now need to create the impex script for that table. Instead of creating manually (which is hard), you can generate the script using hmc.

**Step 1** = Goto hmc □ System □ tools □ Script Generator & open



**Step 2** = Click Generate button to generate the script. This will generate the script for all the tables.



**Step 3** = Copy the script for the item type you are looking into textpad / notepad

Example – I want copy the script for Courses item type code (here is the generated script)

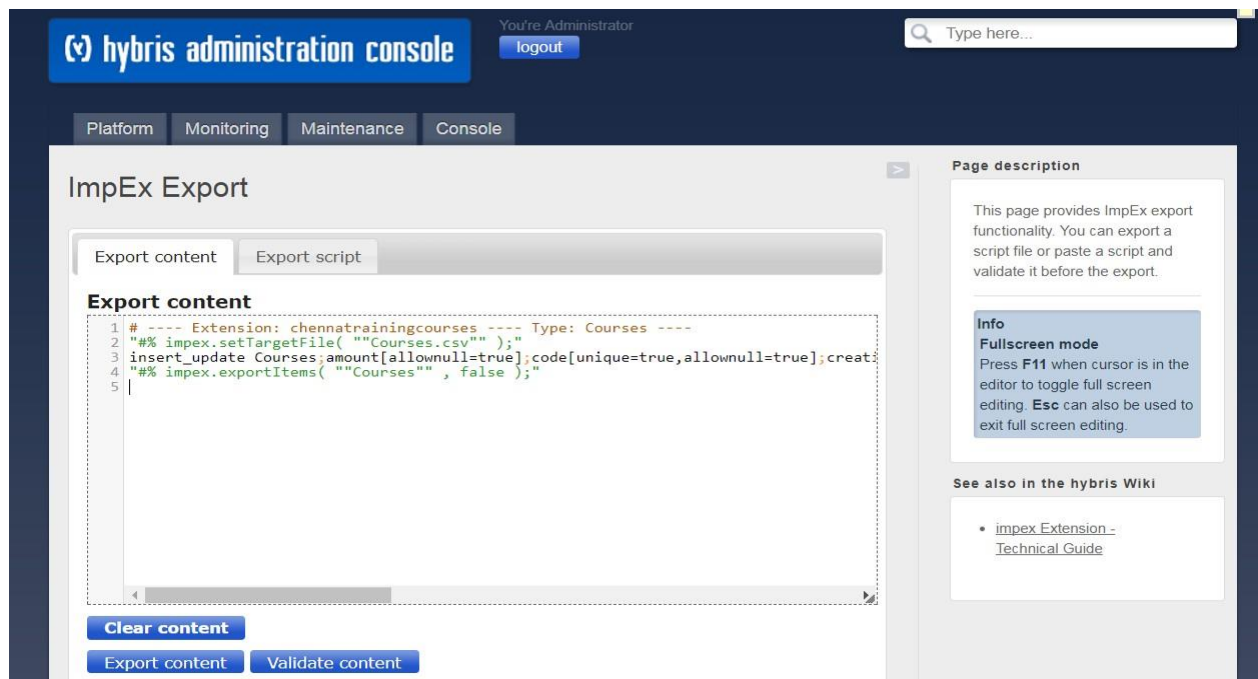
**Note** = Remove `forceWrite=true` in the `creationtime` because creation time can't be changed for jalo classes

```
# ---- Extension: chennatrainingcourses ---- Type: Courses ----
"##% impex.setTargetFile( ""Courses.csv"" );" □ where to export
insert_update
Courses;&Item;amount[allownull=true];code[unique=true,allownull=true];creationtime[forceWrite=true,datef
ormat=dd.MM.yyyy          hh:mm:ss];duration[allownull=true];modifiedtime[dateformat=dd.MM.yyyy
hh:mm:ss];name[allownull=true];owner(&Item) □ How to export
"##% impex.exportItems( ""Courses"" , false );" □ what to export
```

Script after removing `forceWrite=true`, Copy this script into the Console □ ImpEx Export □ Export Content tab

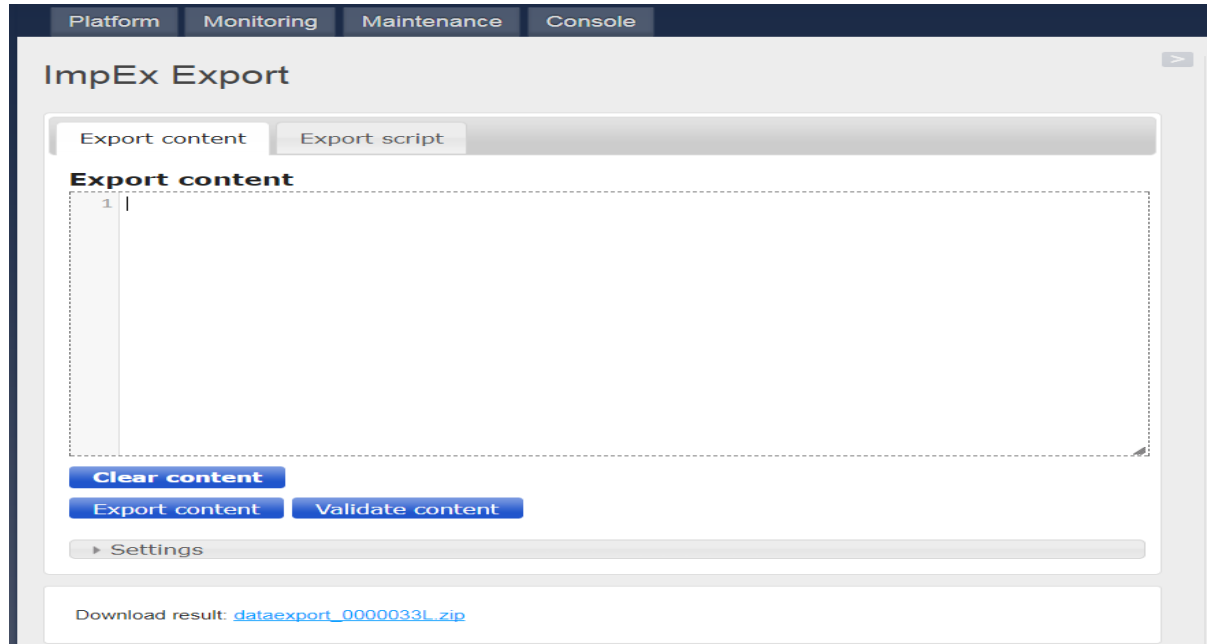
```
#-----
insert_update
Courses;&Item;amount[allownull=true];code[unique=true,allownull=true];creationtime[dateformat=dd.MM.y
yyy          hh:mm:ss];duration[allownull=true];modifiedtime[dateformat=dd.MM.yyyy
hh:mm:ss];name[allownull=true];owner(&Item)
"##% impex.includeExternalDataMedia( ""Courses.csv"" , ""UTF-8"" , ';', 1 , -1 );"
#-----
```

**Step 4** = Goto hac □ Console □ ImpEx Export □ Export Content & paste the modified item type script

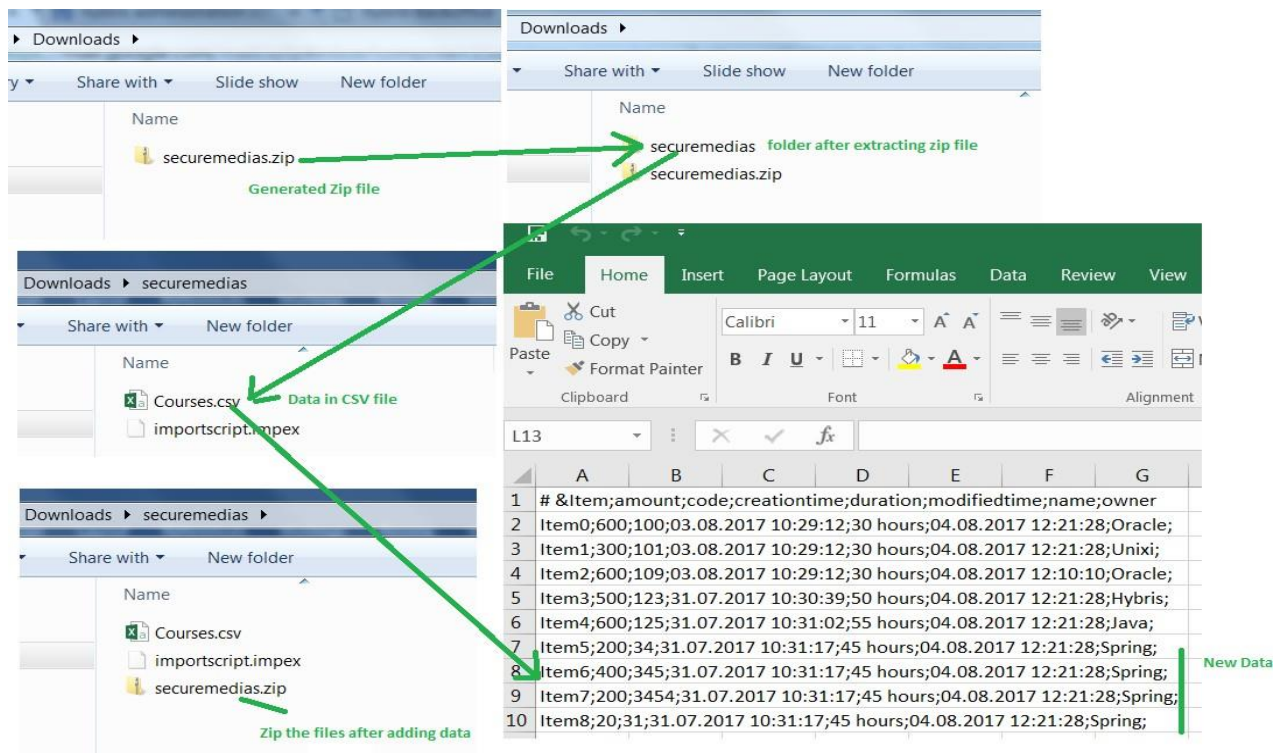


**Step 5** = Click on Export Content and save the file. The file is saved as zip file.

**Note** = You can use the zip file to import the data or you can copy the script and the data into Export content tab and run.



**Step 6** = Extract the zip file and add the new data in the "Courses.csv" file and then zip the file



**Step 7** = Goto hac ☐ Console ☐ Import Script ☐ Choose the zip file and then click on import ( this will import the new data into the database)

**Step 8** = You can validate data in back office / in the database

Course Code	Course Name	Course Duration	Course Amount
31	Spring	45 hours	20
34	Spring	45 hours	200
109	Oracle	30 hours	600
101	Unixi	30 hours	300
100	Oracle	30 hours	600
3454	Spring	45 hours	200
345	Spring	45 hours	400
125	Java	55 hours	600
123	Hybris	50 hours	500

**Q:** Explain Header Level Attribute modifiers?

- alias for export ([alias=theAlias])
- allownull ([allownull=true])
- collection-delimiter, ([collection-delimiter=;])
- dateformat ([dateformat=dd-MM-yyyy])
- forceWrite ([forceWrite=true]), not compatible with the service layer
- ignoreKeyCase ([ignoreKeyCase=true])
- ignorenull ([ignorenull=true]), ignore null for collection
- lang ([lang=en])
- numberformat ([numberformat=#.###,##])
- path-delimiter ([path-delimiter=:])
- unique ([unique=true]) used to mark this attribute
- virtual ([virtual=true,default=value]) needs to have a default modifier as well
- cacheUnique ([cacheUnique=true])
- impex.legacy.mode ([impex.legacy.mode=true])
- batchmode ([batchmode=true]), used with update / remove. Allows to modify >1 item that matches query