Hybris - Create components via IMPEX files

Author: Joni Halabi, Senior UX Developer, Optaros
Published: July 2013

Introduction

Adding components to content slots in Hybris pages is relatively straightforward in the WCMS Cockpit; however exporting that data from the WCMS Cockpit to impex files is relatively difficult and can result in exporting more information than necessary. If you are the kind of developer who likes clean impex files and would just rather create them by hand, this document will hopefully help.

The purpose of this document is to outline how to create basic but commonly used components in Hybris by manually adding them to your impex file. This document will only demonstrate the code used to create components.

Organizing the code in terms of best practices – i.e. core vs. initial data or separating your impex file data by page or category – is a good idea, but is out of scope for this document.

General notes:

Each of the commands below are INSERT_UPDATE commands. An INSERT_UPDATE will do exactly that. If the specified UID already exists in that particular table of the database, this command will update the rest of that UID's entry. If the UID does not exist in that table, this command will add a new entry for the specified UID. This command will not delete entries from the database.

Each INSERT_UPDATE command starts with the name of the database table to be modified. This table name relates directly to the type of component you are adding.

Variables in the impex file:

There are several variables used in the examples below. These variables are used to simplify the INSERT_UPDATE header. All variables should be declared at the top of your impex file. (Otherwise, you will get an error.)

Here is a list of variables used in the examples below:

- **$contentCV** – This is the content catalog version. For example:

```
catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),C
atalogVersion.version[default=Staged])[default=$contentCatalog:Staged]
```

- **$lang** – This is the default language for headers, labels, names, etc. The 2-letter code for the default language is used here. For example: English is 'en'; German is 'de'.

- **$siteResource** – This is a constant used to point to the path of the content catalog directory in the code repository. For example:

```
$siteResource=jar:de.hybris.platform.acceleratorsampledata.constants.Acceler
atorSampleDataConstants&/SITENAMEcore/import/contentCatalogs/$contentCatalog
```

Creating a new content page

Impex files can be used to define new content pages in the site. Content pages can be defined either in core or initial data; however it feels a little more "correct" to define pages in core.

Code:

```
INSERT_UPDATE
ContentPage;$contentCV[unique=true];uid[unique=true];name;masterTemplate(uid,$cont
entCV);label;defaultPage[default='true'];approvalStatus(code)[default='approved'];
homepage[default='false'];previewImage(code,
$contentCV)[default='ContentPageModel__function_preview']

;;about;About Us;ContentPage2Template;/about
```

Explanation:

The code above will create a new content page, called "About Us", for the content catalog specified in $contentCV.

- **uid:** The unique system identifier for the content page.
- **name:** The friendly name for the content page.
- **masterTemplate:** The JSP template that will be used for this page; these are usually defined in cms-content.impex.
- **label:** The URL of the content page. The forward slash preceding the label is important here; this makes it a link relative to the site root. (Otherwise, the link will be relative to the currently displayed page and break for other pages.)

Creating components

A wide variety of component types can be defined in the IMPEX files. The components below are among some of the most common components that are defined in the code. There are certainly other components that can be defined here.

INTERNAL LINK

Code:

```
INSERT_UPDATE
CMSLinkComponent;$contentCV[unique=true];uid[unique=true];name;contentPage(uid,$co
ntentCV);linkName[lang=$lang];&linkRef;&componentRef;target(code)[default='sameWin
dow'];restrictions(uid,$contentCV)[default='']

;;linkAboutUs;About Us Link;about;About Us;linkAboutUs;linkAboutUs
```

Explanation:

The code above will create a link to an internal page, in this case the "About Us" content page.

- **uid:** The unique system identifier for the internal link.
- **name:** The friendly name of the internal link.
- **contentPage:** The label of the content page being linked to.
- **linkName:** The displayed link text (i.e. what the user sees). The language specified is value of the $lang variable. If no language is specified, the system defaults to German.
- **linkRef:** Reference to the link; it is simplest if this is identical to the UID.
- **componentRef:** Reference to the link's component; again, it is simplest if this is identical to the UID.

EXTERNAL LINK

Code:

```
INSERT_UPDATE
CMSLinkComponent;$contentCV[unique=true];uid[unique=true];name;url;linkName[lang=$
lang];&linkRef;&componentRef;target(code)[default='sameWindow'];restrictions(uid,$
contentCV)[default='']

;;linkAccount;Account Link;/my-account;My Account;linkAccount;linkAccount
```

Explanation:

The code above will create a link to any URL. The example above links to a page within the site by relative URL, in this case the "My Account" page; however this link type can be used to create links to external websites as well.

- **uid:** The unique system identifier for the external link.
- **name:** The friendly name of the external link.
- **url:** The URL of the page being linked to. This can be either a relative link *(as in the example above)* or an external site.
- **linkName:** The displayed link text (i.e. what the user sees). The language specified here is value of the $lang variable. If no language is specified, the system defaults to German.
- **linkRef:** Reference to the link; it is simplest if this is identical to the UID.
- **componentRef:** Reference to the link's component; again, it is simplest if this is identical to the UID.

CATEGORY LINK

Code:

```
INSERT_UPDATE
CMSLinkComponent;$contentCV[unique=true];uid[unique=true];name;&linkRef;&component
Ref;category(code,
$productCV);linkName[lang=$lang];target(code)[default='sameWindow']

;;catAdhesives;Adhesives Category
Link;catAdhesives;catAdhesives;pc_child_Adhesives;Adhesives
```

Explanation:

The code above will create a link to a product category page, in this case the "Adhesives" category page.

- **uid:** The unique system identifier for the category link.
- **name:** The friendly name of the category link.
- **linkRef:** Reference to the link; it is simplest if this is identical to the UID.
- **componentRef:** Reference to the link component; it is simplest if this is identical to the UID.
- **category:** The UID of the category that is being linked too.
- **linkName:** The displayed link text (i.e. what the user sees). The language specified is value of the $lang variable. If no language is specified, the system defaults to German.

NAVIGATION NODE

Code:

```
INSERT_UPDATE CMSNavigationNode;uid[unique=true];name;parent(uid,
catalogVersion(CatalogVersion.catalog(Catalog.id[default=]),CatalogVersion.version
[default=Staged])[default=:Staged]);links(&linkRef);&nodeRef;title[lang=$lang]

;navnodeAccount;Account Navigation
Node;;linkResetPassword,linkOrderHistory,linkAddressBook,linkProfile;navnodeAccoun
t;My Account
```

Explanation:

The code above will create a navigation node. A navigation node is a collection of links that are grouped together and are usually used in top navigation bars and footer components.

- **uid:** The unique system identifier for the navigation node.
- **name:** The friendly name of the navigation node.
- **parent:** The parent node for this navigation node. This can usually be left blank.
- **links:** A comma-delimited list of link components to be included in the navigation node. The link components should be identified by their UIDs.
- **nodeRef:** Reference to the node; it is simplest if this is identical to the UID.
- **title:** The (optional) title of the navigation node, displayed on the page directly above the list of links. The language specified here is value of the $lang variable. If no language is specified, the system defaults to German.

FOOTER COMPONENT

Code:

```
INSERT_UPDATE
FooterComponent;$contentCV[unique=true];uid[unique=true];wrapAfter;&componentRef;n
avigationNodes(&nodeRef)

;;siteFooterComponent;6;siteFooterComponent;navnodeResources,navnodeAccount
```

Explanation:

The code above will create a footer component, which contains a navigation node. The footer component will display the title and all links in the navigation node. It will also automatically display the links in columns, wrapping to a new column after a specified number of links.

- **uid:** The unique system identifier for the footer component.
- **wrapAfter:** An integer representing the maximum number of links per column in the component. Extra links will wrap to a new column.
- **componentRef:** Reference to the link's component. It is simplest if this is identical to the UID.
- **navigationNodes:** A comma-delimited list of the UIDs of all navigation nodes to be included in the footer component.

IMAGE COMPONENT

Code:

```
INSERT_UPDATE
Media;$contentCV[unique=true];code[unique=true];realfilename;@media[translator=de.
hybris.platform.impex.jalo.media.MediaDataTranslator];mime[default='image/jpg'];al
tText

;;/images/logo.gif;logo.gif;$siteResource/images/logo.gif;image/gif;Site Logo
```

Explanation:

The code above will create an image component. An image component is generally used within other components that require images; for example site logo or banner components.

- **code:** The relative path and file name of the image, relative to the content catalog directory.
- **realfilename:** The image filename.
- **@media:** The full path of the image. You'll notice that the data includes a variable called $siteResource, which contains the path of the content catalog directory in the code repository.
- **mime:** The mime type of the image.
- **altText:** Alternate text for the image, used just as you would expect alternate image text to be used in the HTML <img> tag.

BANNER COMPONENT

Code:

```
# Banner Component
INSERT_UPDATE
BannerComponent;$contentCV[unique=true];uid[unique=true];name;&componentRef;urlLin
k

;;myBanner;My Banner;myBanner;"/"

# Banner Components/Image
UPDATE
BannerComponent;$contentCV[unique=true];uid[unique=true];headline[lang=$lang];$pic
ture[lang=$lang]

;;myBanner;My Banner Headline;my-banner.jpg
```

Explanation:

The code to create a banner component is actually in 2 parts, mostly for simplicity. The first INSERT_UPDATE section defines the component.

- **uid:** The unique system identifier for the site logo component.
- **name:** The friendly name of the logo component.
- **componentRef:** Reference to the link's component. It is simplest if this is identical to the UID.
- **urlLink:** The URL that the banner links to when clicked.

The next UPDATE section adds an image component to the banner component.

- **uid:** The unique system identifier for the site logo component to which you are adding an image component.
- **headline:** A text headline for the banner.
- **picture:** The filename of the image component being added; this should match the *realfilename* of the image component, as specified in the Image Component section of this document.

Adding components to a content slot

Once a component has been created, it can be added to any defined content slot on any page.

Code:

```
INSERT_UPDATE
ContentSlot;$contentCV[unique=true];uid[unique=true];cmsComponents(&componentRef)

;;FooterContentSlot;linkHomepage,linkAboutUs,linkPrivacy,linkHelp
```

Explanation:

The example above adds several link components to a pre-defined content slot in the site footer.

- **uid:** The unique system identifier for the content slot to which components are being added.
- **cmsComponents:** A comma-delimited list of the components being added to the specified content slot. Components are added to the content slot in the order in which they are listed.

Component restrictions

Restrictions can be added to components to restrict display of a particular component for a set of users. The restrictions defined below are all based on user groups defined in the system.

If such a restriction is applied to a component, all users who are a member of the user group specified in the restriction will be able to see the content of that component. The component will not be rendered for all other users.

USER RESTRICTION

Code:

```
INSERT_UPDATE
CMSUserRestriction;$contentCV[unique=true];uid[unique=true];name;users(uid);

;;anonymousUser_restr;Anonymous User Restriction;anonymous
```

Explanation:

The example above defines a component restriction for any anonymous user on the site. Components that include this restriction will only be displayed to users who are anonymous, or not logged in. This example can be extended to any user group defined in the system.

- **uid:** The unique system identifier for the restriction.
- **name:** The friendly name of the restriction.
- **users:** The ID of the user group to which the restriction is being applied.

INVERSE RESTRICTION

Code:

```
INSERT_UPDATE
CMSInverseRestriction;$contentCV[unique=true];uid[unique=true];name;originalRestri
ction(uid);
;;authenticatedUser_restr;Authenticated User Restriction;anonymousUser_restr
```

Explanation:

Inverse restrictions are defined as the opposite of another defined restriction. These restrictions can be useful for displaying content to all users but those in one particular user group. The example above defines a restriction for all users who are authenticated, or all users who are *not* anonymous.

- **uid:** The unique system identifier for the inverse restriction.
- **name:** The friendly name of the restriction.
- **originalRestriction:** The UID of the restriction that is being inversed.

ADDING A RESTRICTION TO A COMPONENT

Code:

```
UPDATE
FooterComponent;$contentCV[unique=true];uid[unique=true];restrictions(uid,$cont
entCV)[default='']

;;siteFooterComponent;anonymousUser_restr
```

Explanation:

You can add visibility restrictions to any component by specifying restrictions at the end of any command. The example above adds a restriction to our footer component, so that the component only displays for anonymous users.

- **uid:** The unique system identifier for the component to which the restriction is being added.
- **restrictions:** The unique system identifier for the restriction being added. Technically, more than one restriction can be added to a component; however this can be awkward if they conflict with each other. No more than one restriction is added to a component in normal usage.