Setting up Syndication for Agility to Magento Connection

Full Export

You may want to do an initial full extract of data from Agility into Magento, in which case you will need to configure syndication to perform the full export extract. Note that the full exporter also performs an update on products that have already been added to Magento via the connector. Incremental updates from Agility to Magento will also required syndication to be set up in the same way.

Note that all label names should be used exactly as stated in this article.

Syndicate.xml file

Magento Core

The core attribute values on a product in Magento are as follows:

sku	This is a required field and must have a unique value
name	This is a required field and must have a unique value
price	If no price is given, it will be set to zero
weight	If no weight is given, it will be set to zero
attribute set name	This is a text field (such as 'Top') and defines the attribute set the product belongs to. If none is given, the default Magento attribute set will be allocated. The plugin creates an attribute set with the name passed in the xml, if no attribute set is found that matches on the name. All the attributes that are on the product will be added to the newly created attribute set, which is modeled on the Magento default attribute set. Note that any new attributes will be added to the default set, if a name is not specified.
stock status	This is an attribute that uses a yes/no choice list, where yes indicates that the product is in stock, and no that it is out of stock. If the quantity on a product is greater than 0 the connector sets it to in stock by default, although if the quantity is 0 and the stock status attribute is set to in stock (yes), this will over ride the out of stock setting.
quantity	A numeric value. Configurable products can not have a quantity value.
status	This is a text attribute that uses a yes/no choice list, where yes indicates it is enabled and no that it is disabled. A product's status is set to enabled by default. However you can override this by passing the status value using this attribute. If it is not set to enabled (yes) then it will be set to disabled.
type	This is a text attribute that uses a single value choice list, of Magento product types. The choice list values should be one of simple, grouped, configurable, virtual, bundle, or downloadable. The value is optional, as product type is derived in the code, except in the case of virtual products where this is a required field. You can also use the type field to change a product type when doing an update of Magento.

By default, visibility is set to 'both' for configurable products and to 'not visible individually' for all other product types. Note that there is no setting for visibility in the magento core.

Product type is derived, apart from for virtual products, where the value of 'product type' must be set to 'virtual'. If the product has variants in agility, it will be a configurable type (with product options) in Magento. A product that has downloadable links in Agility will be a downloadable product type in Magento. A product that has group product relations will be a type of 'grouped'. A product that has bundle product relations will be a type of 'bundled'. Any other product is a simple product, including variants. You can set the product type in the magento core (see table above) - although it is only a required field for virtual products. If you want to change a product type you can set the value here and the Magento plugin will update the product type for you.

The syndicate.xml file should be set to have a ‘magentoCore’ which includes these attributes - as per the example below.

```
<attribsAsXmlRule label="magentoCore" includeRootElement="true" includeDeclaration="true" templateId="8">
```

```
    <attribRule id="219" label="name" cdata="true"/>
    <attribRule id="112" label="sku" cdata="true"/>
    <attribRule id="1" label="price" cdata="true"/>
    <attribRule id="445" label="attributesetname" cdata="true"/>
    <attribRule id="306" label="weight" cdata="true"/>
    <attribRule id="489" label="quantity" cdata="true"/>
    <attribRule id="490" label="instock" cdata="false"/>
    <attribRule id="492" label="status" cdata="true"/>
    <attribRule id="1269" label="type" cdata="true"/>
</attribsAsXmlRule>
```

The id values shown above are the attribute id values in Agility. You will of course need to amend these values to match those in your attribute definitions.

In the example, the magento core uses a template with an id of 8 - which looks as below:

```
<template templateId="8" name="magentoCore">
   <startTemplate><![CDATA[<magentoCore>\n]]></startTemplate>
   <endTemplate><![CDATA[<\magentoCore>\n]]></endTemplate>
   <attribTemplate><![CDATA[<%ATTRIB_LABEL%>%ATTRIB_VALUE%</%ATTRIB_LABEL%>\n]]></attribTemplate>
</template>
```

General attributes should include the attribute type, as the type is used to make decisions on the kind of attribute to write to the Magento eav attribute table. Define the attribute with the type in the template as follows

```
<attribTemplate><![CDATA[<%ATTRIB_LABEL%>%ATTRIB_VALUE%</%ATTRIB_LABEL%>\n<type>%ATTRIB_TYPE%</type>\n]]></attribTemplate>
```

Note that if you want to map an attribute to an existing Magento attribute, then the label value in the syndicate.xml file must match the Magento attribute code value. If it is an attribute in Agility that you want to create in Magento, then it will be created using the label value you have given it. If you camel case the attribute label, for example 'newAttributeName', this will appear in magento with an attribute code of 'newattributename' and a front end label value of 'New Attribute Name'.

Note that Magento won't accept an attribute with a name longer than 30 characters, and so all attribute names must be 30 characters or less.

In this example, the Agility attribute with an id of 375 is mapped to the existing Magento attribute code for products 'description'.

```
      <!--Description-->
       <attribRule id="375" label="description" cdata="true"/>
```

In this example, the Agility attribute with an id of 600 will be used to create a new Magento attribute for products of 'newvalue'. Note that any new attributes will be added to the attribute set as defined on the product it is added to.

```
      <!--New Value-->
       <attribRule id="600" label="newvalue" cdata="true"/>
```

Choice Lists

We need to identify that an attribute is a choice list and how it is separated on the syndicate.xml as per the example below, where 'availablesizes' is the attribute name, '!CL' indicates that it is a choice list attribute, and ':/' indicates how the choice list has been separated. The value '!CL:x' must follow the choice list attribute name, to indicate to the magento connector code that it is a choice list attribute, where x is the choice list separator value. In the example, availablesizes is a choice list that has been separated by a forward slash.

```
<attribRule id="224" label="availablesizes!CL:/" cdata="true"/>
```

Variants - in Magento, variants are created as simple products, and the parent product is a configurable product. Configurable products can be set to be related to the variant products by setting attributes to match. For example, the match could be an attribute of size which contains the values ‘S,M,L’, and/or an attribute of color that contains the values ‘Red,Green,Black’. The Magento plugin caters for up to 2 attributes to match on from Agility, the attributes can be anything (but are typically size and color). The attributes set on a variant in syndication will be used to create the match. A variant with the following defined would have both size and color attributes written to the simple product in Magento. Also, size and color (and their values) would be used as the matching attribute for the parent product. The variants would then be displayed as product options in Magento.

Note that any attributes defined as linking attributes on a variant, must first be defined in Magento as super attributes.

```
<attribRule id="223" label="size" cdata="true"/>
<attribRule id="444" label="color" cdata="true"/>
```

Related Product linking - in Magento, there are 3 types of related product links (that is, where a product is linked to another product) which are Cross-sell, Up-sell and Related. The syndicate.xml will need to search for linked products - assess the linking attribute to determine which label to give the product searched for, as it is this value that is used to define the link type in Magento. In the example below, the label ‘related’ has been used. Note that the names to use are one of ‘crosssell’, ‘upsell’ and ‘related’ and the label should exactly match one of these values.

The 'find link-to distinct any via #472' statement means that it is searching for a product relation that is linked to the product using the relation attribute 472. The attribute id="112" statement will include the name of the related product.

```
<!--Related Products -->
    <searchAsXmlRule label="related_data"  query="FIND LINK-TO DISTINCT ANY VIA #472" includeRootElement="false"
includeDeclaration="false" templateId="11">
    <attribRule id="112" label="related" cdata="true"/>
</searchAsXmlRule>
```

The associated template is defined as

```
    <template templateId="11" name="relatedSku">
         <startTemplate><![CDATA[<relatedSku>\n]]></startTemplate>
            <attribTemplate><![CDATA[<attribute>\n<name>%ATTRIB_LABEL%</name>\n<value>%ATTRIB_VALUE%</value>\n<type>%ATTRIB_TYPE%
</type>\n</attribute>\n]]></attribTemplate>
        <endTemplate><![CDATA[</relatedSku>\n]]></endTemplate>
```

```
    </template>
</template>
```

Grouped Product

A grouped product is made up of multiple, standalone products that are presented as a group. You can offer variations of a single product, or group them by season or theme to create a coordinated set.

The set up in Agility for a grouped product is similar to adding related products (see section above). You will need a product relation attribute, and use this to relate products that you want to group together on the grouped product. Syndication should be set as:

```
 <!-- Grouped Products -->
<searchAsXmlRule label="group"  query="FIND LINK-TO DISTINCT ANY VIA #1268"includeRootElement="false" includeDeclaration="false"
templateId="16">
        <attribRule id="32" label="associated" cdata="true"/>
</searchAsXmlRule>
```

with a corresponding template as (note the value 'associated' in the name is required).

```
<template templateId="16" name="group">
        <startTemplate><![CDATA[<group>\n]]></startTemplate>
<attribTemplate><![CDATA[<attribute>\n<name>associated</name>\n<value>%ATTRIB_VALUE%</value>\n
<type>%ATTRIB_TYPE%</type>\n</attribute>\n]]></attribTemplate>
        <endTemplate><![CDATA[</group>\n]]></endTemplate>
</template>
```

Bundle Product

See the article Bundle Product Setup for details on how to set up a product in Agility that will be added as a bundle product to Magento by the connector. The bundle product should have the items in the bundle linked to the product via a product relation. The syndicate.xml file should look like:

```
<attribsAsXmlRule label="bundleOption" includeRootElement="true" includeDeclaration="true" templateId="14">
        <attribRule id="493" label="title" cdata="true"/>
        <attribRule id="494" label="type" cdata="true"/>
        <attribRule id="495" label="isRequired" cdata="true"/>
        <attribRule id="496" label="priceType" cdata="true"/>
        <attribRule id="497" label="shipmentType" cdata="true"/>
</attribsAsXmlRule>
<searchAsXmlRule label="bundleLink" query="FIND LINK-TO DISTINCT ANY VIA #501" includeRootElement="false"
includeDeclaration="false" templateId="15">
        <attribRule id="40" label="sku" cdata="true"/>
        <attribRule id="498" label="quantity" cdata="true"/>
        <attribRule id="499" label="canChangeQuantity" cdata="true"/>
        <attribRule id="500" label="isDefault" cdata="true"/>
        <attribRule id="1" label="price" cdata="true"/>
</searchAsXmlRule>
```

with 2 corresponding templates (one for bundleOption, the other for bundleLink) -

```
<template templateId="14" name="bundleOption">
    <startTemplate><![CDATA[<bundleOption>\n]]></startTemplate>
    <endTemplate><![CDATA[</bundleOption>\n]]></endTemplate>
    <attribTemplate><![CDATA[<%ATTRIB_LABEL%>%ATTRIB_VALUE%</%ATTRIB_LABEL%>\n]]></attribTemplate>
</template>

<template templateId="15" name="bundleLink">
    <startTemplate><![CDATA[<bundleLink>\n]]></startTemplate>
    <endTemplate><![CDATA[</bundleLink>\n]]></endTemplate>
    <attribTemplate><![CDATA[<%ATTRIB_LABEL%>%ATTRIB_VALUE%</%ATTRIB_LABEL%>\n]]></attribTemplate>
</template>
```

Images

There are three attributes required for an image on the syndication extract, which are path, label and filetype. The path should contain a path to the image, which can either be a URL path or a path on a machine that the magento app code will run on. The label value is used to create both the asset name and label, and the filetype corresponds to the mime type of the image. If no filetype is supplied from syndication, the connector will derive the filetype from the asset url ending, for example a 'jpg' file will become a 'jpeg' mime type. The label 'thumbnail' means that a thumbnail image will be created in Magento. Separate extracts are required for each image type, with corresponding templates. See below for sample syndicate.xml -

```
<!--Thumbnail Image -->
    <searchAsXmlRule label="thumbnailimage" query="FIND LINK-TO DISTINCT ANY VIA #464" includeRootElement="false"
includeDeclaration="false" templateId="5">
    <attribRule id="427" label="path" cdata="true"/>
    <!--Image Name -->
    <attribRule id="470" label="label" cdata="true"/>
    <!--Image Type -->
    <attribRule id="488" label="filetype" cdata="true"/>
</searchAsXmlRule>
```

Other image types are 'image' (for main image), 'swatchimage' (for swatch images) and smallimage (for small image). Each of these types needs a corresponding template as below. This example shows the template for a thumbnail image, so substitute the name 'thumbnailimage' for 'image', 'swatchimage', 'smallimage' as required.

```
    <template templateId="5" name="thumbnailimage">

        <startTemplate><![CDATA[<image>\n<type>thumbnail</type>\n]]></startTemplate>
        <endTemplate><![CDATA[</image>\n]]></endTemplate>
        <attribTemplate><![CDATA[<%ATTRIB_LABEL%>%ATTRIB_VALUE%</%ATTRIB_LABEL%>\n]]></attribTemplate>

    </template>
```

Once your syndicate.xml file and your templates are set up correctly, you are ready to run syndication using the 'FullExporter.cmd' function, which will generate an xml output file from Agility.

Downloadable Products

Refer to the article on downloadable product setup for details on how to configure your data for downloadable products. In syndication, you will need two search as xml rules for download links and download samples, which will return any related download links and samples on the product. Note that the product relation id is used in the 'find link-to distinct

any' query.

```
<searchAsXmlRule label="downloadlink"  query="FIND LINK-TO DISTINCT ANY VIA #1265" includeRootElement="false"
includeDeclaration="false" templateId="14">
        <!--Shareable -->
        <attribRule id="1259" label="isshareable" cdata="true"/>
        <!--Link Title -->
        <attribRule id="1251" label="title" cdata="true"/>
        <!--Links Purchased Separately-->
        <attribRule id="1253" label="purchasedseparately" cdata="true"/>
        <!--Link Price-->
        <attribRule id="1254" label="price" cdata="true"/>
        <!--Link Filepath-->
        <attribRule id="1255" label="imagepath" cdata="true"/>
        <!--Link Image Name-->
        <attribRule id="1256" label="imagename" cdata="true"/>
        <!--Link sample filepath -->
        <attribRule id="1257" label="linksamplepath" cdata="true"/>
        <!--Link sample image name -->
        <attribRule id="1258" label="linksamplename" cdata="true"/>
        <!--Maximum number of downloads -->
        <attribRule id="1260" label="numberofdownloads" cdata="true"/>
</searchAsXmlRule>



<searchAsXmlRule label="downloadsample"  query="FIND LINK-TO DISTINCT ANY VIA #1266" includeRootElement="false"
includeDeclaration="false" templateId="15">
        <!--Sample title -->
        <attribRule id="1261" label="title" cdata="true"/>
        <!--Sample filepath -->
        <attribRule id="1262" label="samplepath" cdata="true"/>
        <!--Sample image name -->
        <attribRule id="1263" label="samplename" cdata="true"/>
</searchAsXmlRule>
```

The corresponding templates for the download links and samples are:

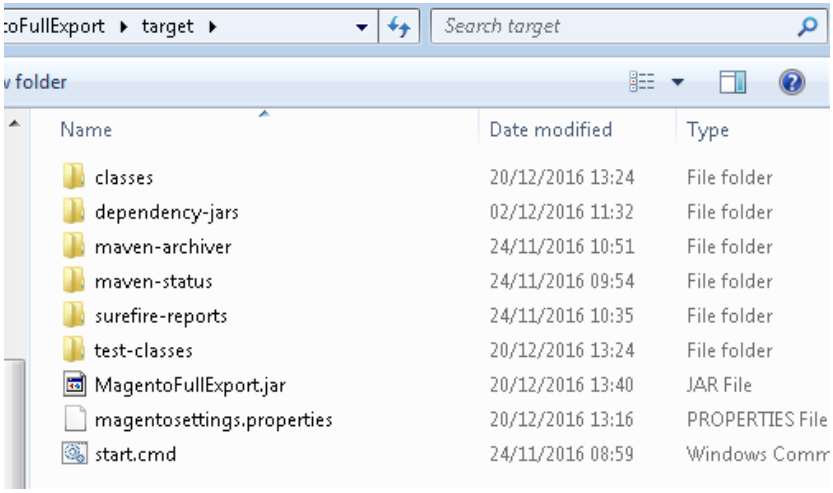
```
        <template templateId="14" name="downloadlink">
        <startTemplate><![CDATA[<downloadlink>\n]]></startTemplate>
        <endTemplate><![CDATA[</downloadlink>\n]]></endTemplate>
        <attribTemplate><![CDATA[<%ATTRIB_LABEL%>%ATTRIB_VALUE%</%ATTRIB_LABEL%>\n]]></attribTemplate>
    </template>

        <template templateId="15" name="downloadsample">
        <startTemplate><![CDATA[<downloadsample>\n]]></startTemplate>
        <endTemplate><![CDATA[</downloadsample>\n]]></endTemplate>
        <attribTemplate><![CDATA[<%ATTRIB_LABEL%>%ATTRIB_VALUE%</%ATTRIB_LABEL%>\n]]></attribTemplate>
    </template>
```

Running a Full Export Transform

Once you have the full export output file from syndication, you can run the full export process which will take the syndication file, transform it and create the corresponding categories, products and product variations in Magento.

MagentoFullExport

The code to run the full export file is in the 'MagentoFullExport' folder which should have been given to you by a support contact. The full export folders look as below:



Within the folder, locate the file 'magentosettings.properties'. Edit this file as follows:

MagentoInstallName	the name of the magento installation.
MagentoServer	the IP address of the magento server location.
MagentoUser	user name credential for magento.
MagentoPsw	password credential for magento.

InputFile	the full export file that has been generated from syndication.
MagentoFeedFile	this is an optional field. If you have a large file of data, use the feedfile. This is an internal file used in the processing of a full extract to Magento, it should exist before running the full export code, and you must have read/write access to it.

magentosettings.properties

```
# Magento Install Name - put the name of your magento setup in here
MagentoInstallName=magento
# MagentoServer - put the magento server location here : default=localhost (127.0.0.1)
MagentoServer=127.0.0.1
# MagentoUser - put the magento user here
MagentoUser=CatherineL
# MagentoServer - put the magento psw here (needs to be hashed)
MagentoPsw=August16
# Input file - put your file input for the full export in here
InputFile=C:\\Agility_Syndication_HEAD.34\\FullExportTest.xml
# Magento feed file - needs to have r+w access -
MagentoFeedFile=C:\\Users\\catherine learmonth\\Documents\\Magento\\FullExport.xml

```

Run the full export by running the start.cmd file from a run command window. Once this has finished, check that your products and categories have been written to Magento as expected. Once the products have been written to Magento via the connector, the full export can be run again to update the product data from Agility to Magento.

Any error messages are written to the **system.log** file in xampp\htdocs\magento\var\log.

Incremental updates from Agility to Magento

Magento Adapter

Once you have exported Agility data into Magento, using the full syndication export process, then you may wish any changes that you make to the data in Agility (including update, delete, create, link, unlink and move) to be reflected straight away in Magento.

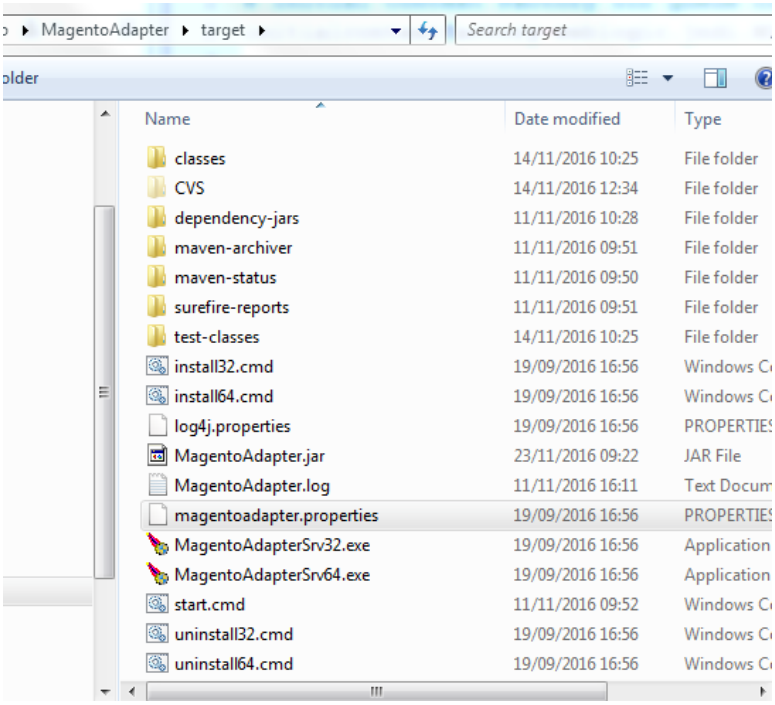
Note that link, unlink and move actions can only be reflected in Magento by using the incremental updater, because the connector will be passed the parent id to work out the new position in Magento. The full exporter would not include the parent id and so these actions would not be reflected. A delete action likewise would only be picked up by the Magento adapter.

The full exporter would pick up any product updates and creates. You could run the full exporter once to extract all the data, and again to pick up updates and creates made in bulk. The connector would then update and create Magento products/categories and options accordingly.

Incremental changes can be made by running syndication server in conjunction with the Magento Adapter, which should be supplied to you by your Agility support contact. If you want to include move and link events, ensure you have defined this in the publishFilterRule and messageFilterRule of the syndicate.xml file, for example:

```
<publishFilterRule export="true" add="true" link="true" update="true" move="true"/>
<messageFilterRule export="true" add="true" link="true" update="true" move="true"/>
```

The contents of the Magento Adapter folder look as follows:



Edit the file **magentoadapter.properties** within the target folder.

The initialcontextfactory should be set to point to your application server. You can find this value in your syndication.properties file within the syndication folder.

The providerurl is the url instance and port where your Agility instance is running.

The connectionfactory and syndicationqueue settings can again be found within the syndication.properties file.

new 1 | magentoadapter.properties | magentosettings.properties | magentoadapter.properties

```
# Initial Context Factory for Queue creation
#initialcontextfactory=org.jboss.naming.remote.client.InitialContextFactory
initialcontextfactory=weblogic.jndi.WLInitialContextFactory

# Provider URL - put your appserver address in here
#providerurl=remote://10.11.2.9:4447
providerurl=t3://172.24.4.119:7001

# Queue Security Principal
#securityprincipal=jboss

#Queue Security Credentials
#securitycredentials=jb0ss

# Queue URL Package Prefixes
#urlpkgprefixes=org.jboss.ejb.client.naming

# Connection Factory
#connectionfactory=jms/RemoteConnectionFactory
connectionfactory=jndi/ConnectionFactory

# Syndication Server Queue Name
#syndicationqueue=jms/queue/SyndicationQueue
syndicationqueue=jndi/SyndicationQueue

```

It is possible to map Agility Workflow states to actions in the Magento Adapter. Staying in the magentoadapter.properties file you can edit this section:

```

#workflow state ids that should be treated as an 'add' message (comma separated list, no spaces, e.g., workflowAddIds=1,2,3)
workflowAddIds=
#workflow state ids that should be treated as an 'update' message (comma separated list, no spaces, e.g., workflowUpdateIds=4,5,6)
workflowUpdateIds=
#workflow state ids that should be treated as a 'delete' message (comma separated list, no spaces, e.g., workflowDeleteIds=7,8,9)
workflowDeleteIds=
```

Here, you can list all the Agility Workflow State IDs that map to an Add, Update or Delete action. These should be entered as a comma separated list with no spaces, e.g.,

workflowAddIds=1,2,3

When Agility Syndication publishes a Workflow message, the State ID is checked against these lists to determine the action that should be taken. If the State ID in the message is not present in any of the lists then the message will not be sent to Magento.

Update the **magentosettings.properties** file within the target folder.

new 1 | magentoadapter.properties | magentosettings.properties | magentoadapter.properties

```
# Magento Install Name - put the name of your magento setup in here
MagentoInstallName=magento
# MagentoServer - put the magento server location here : default=localhost (127.0.0.1)
MagentoServer=127.0.0.1
# MagentoUser - put the magento user here
MagentoUser=CatherineL
# MagentoServer - put the magento psw here (needs to be hashed)
MagentoPsw=August16

```

Here enter the magento server url along with the Magento user and password details. Be careful not to include any spaces at the end of the values that you enter in this file. The adapter code strings the values in this file together, and so any spaces will invalidate the string.

Once these 2 files have been edited, you are ready to run the Magento adapter. Do this by running the start.cmd file from a command window.

You must have syndication server running to pick up changes made to the Agility structure. The Magento adapter will pick up messages from syndication and update, create, move, link, unlink and delete options, products and categories within Magento as appropriate.

Link and Unlink

When linking a product to another category, the product will be added as a product in the Magento category it has been linked to in Agility. There will be only one physical instance of the product in Magento. When unlinking a product, it will be removed from the appropriate category in Magento.

When linking an Agility variant of a product to another product, the equivalent Magento option will be visible on both products. When the variant is unlinked, in Magento the option will be removed from the product it has been unlinked from in Agility. The Magento options will be added and removed from the Magento category as appropriate. There will be only one physical instance of the Magento simple (option) product.

Note that it is not possible to copy a category to another point in the Magento structure, and so linking a category in Agility will result in the following message appearing in the system.log:

'AgilityImport:link it is not possible to link a category in the Magento structure'.

So any categories that are linked in Agility will not have that structure reflected in Magento.

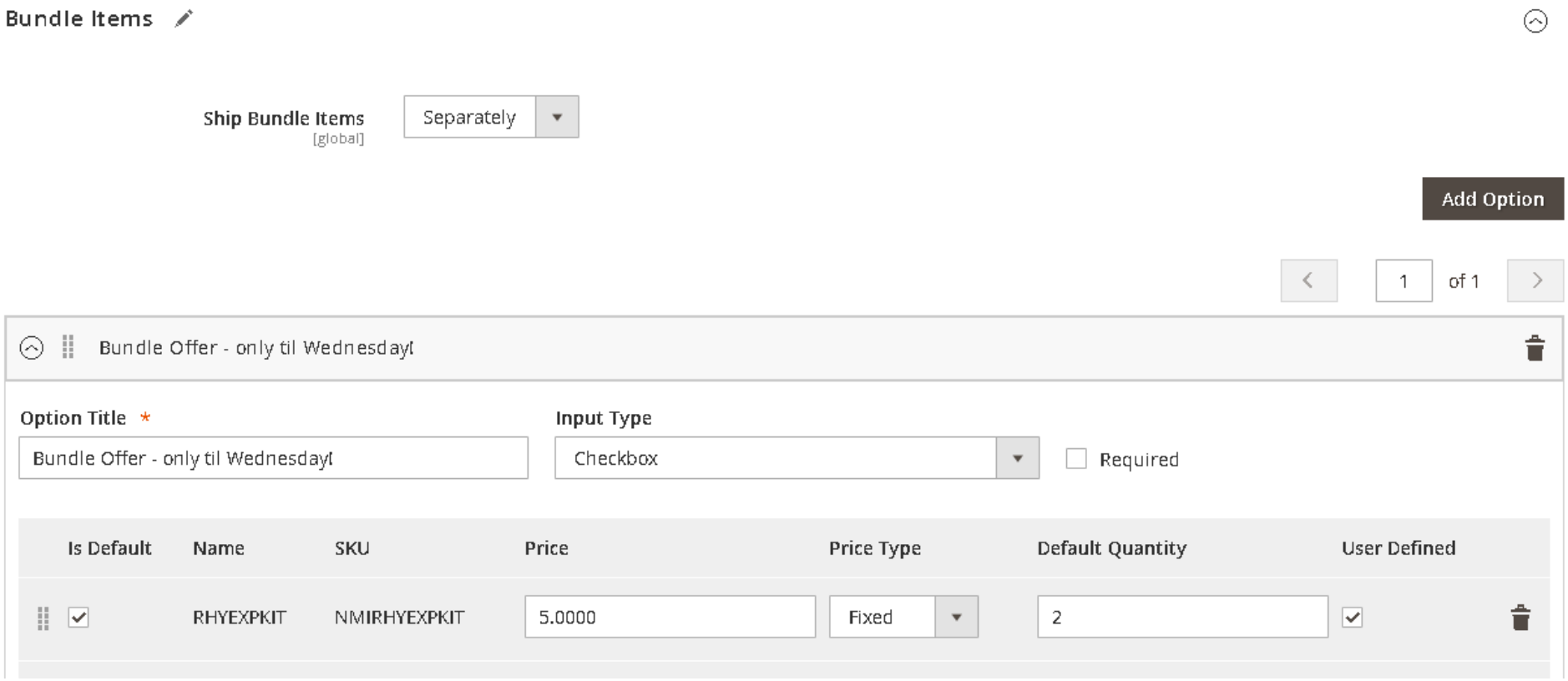
Move

When moving a variant in Agility from one product to another, this change will be reflected in the corresponding Magento option. The option will be removed from the old parent, and added to the new parent. Both variant and product moves will show in their new category structure if that has changed.

It is possible to move a category in Magento, and so a category that has been moved in Agility will be reflected in Magento.

Bundle Product Setup

A Magento bundle product is a 'build your own' set of products. Each item in a bundle can be either a simple or a virtual product. An example bundle in Magento looks as follows:



You can set up a bundle product in Agility so that the connector can create a bundle product in Magento automatically, by following the instructions below.

Create a product in Agility that will be the 'main' product and should contain the bundle option value attributes. Then relate the bundle products to this main product using a product relation. Each of the bundle products should contain the bundle link value attributes.

The main bundle product should have the following attributes

Price Type	Value is either fixed or dynamic. A fixed price will display alongside the linked sku in the bundle links. Set up a choice list with these 2 values and an attribute that will use them. Add the attribute to the main bundle product.
Shipment Type	One of separately or together. Set up a choice list with these 2 values and an attribute that will use them. Add the attribute to the main bundle product. This sets the 'Ship Bundle Items' field.

Title	This is the title of the bundle, rather than the title of the product and sets the Option title field.
Type	One of select, radio, checkbox, multi. Set up a choice list with these values and an attribute that will use them. Add the attribute to the main bundle product. This sets the Input type box on the bundle option.
is Required	This is a boolean yes/no value. Set up an attribute that uses a choice list with a yes/no value and add the attribute to the bundle product. This checks the 'required' box on the bundle option if set to yes.

Once you have set up the attributes as shown in the table, and added them to the bundle product, you can now link the products that comprise the bundle to the bundle product.

Each of the linked products should have the following attributes set:

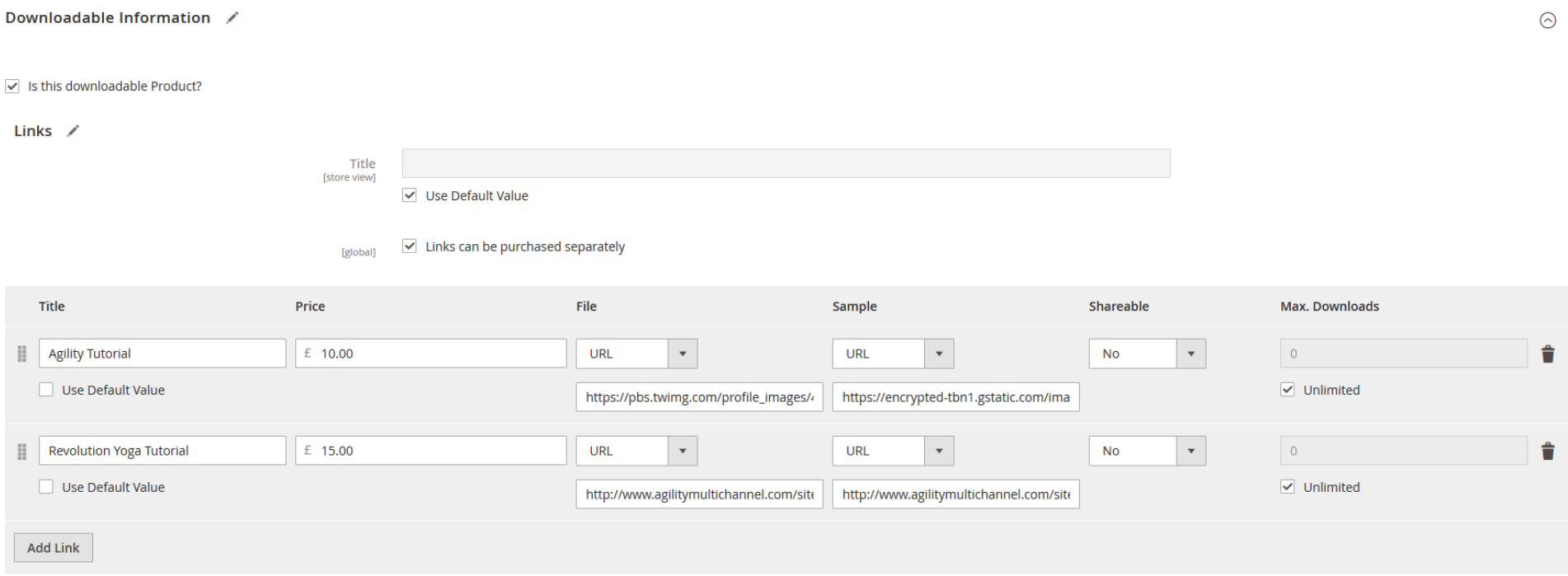
Can Change Quantity	This is a boolean yes/no attribute. Set up an attribute that uses a choice list with a yes/no value and add the attribute to the linked bundle product. When set to yes this allows the quantity field in the option link to be editable or not.
is Default	This is a boolean yes/no attribute. Set up an attribute that uses a choice list with a yes/no value and add the attribute to the linked bundle product. When set to yes, the 'default' checkbox is ticked on the product list option.
Price	This is the price of the linked product. Add the price attribute to the linked product in Agility.
Quantity	This is the quantity of the linked product. Add the quantity attribute to the linked product in Agility.
Sku	This is the sku value of the linked product and should be set on the linked product. Note that the SKU should reference a sku in Magento that already exists, or that will be added as part of a full export.

So now you have the setup in Agility for a bundle product, you will need to configure syndication so that it will extract the bundle product information.

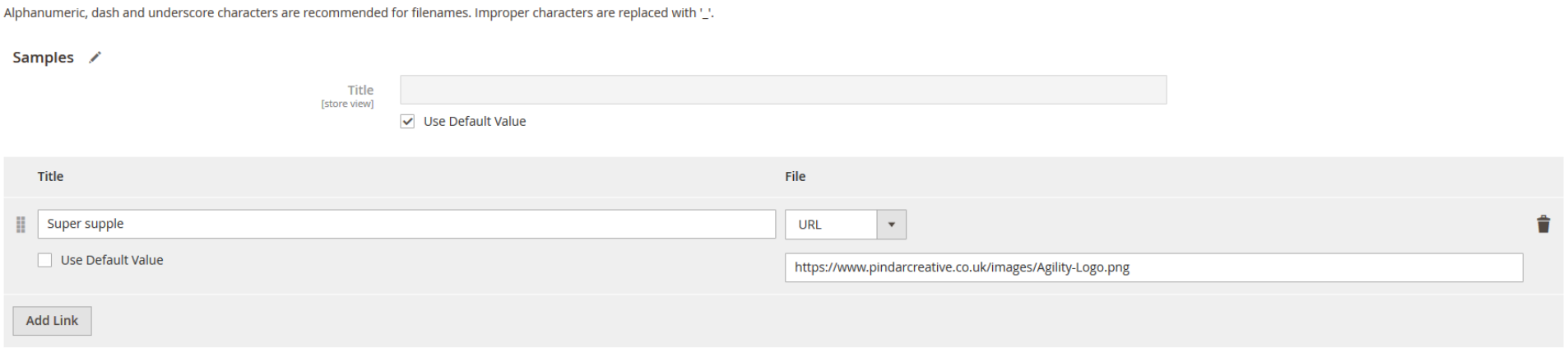
Downloadable Product Setup

If you would like the Magento plugin to create downloadable products in Magento from Agility data, you'd need to set up suitable data in Agility for this.

A Magento downloadable product can contain one or more download links as per the screenshots:

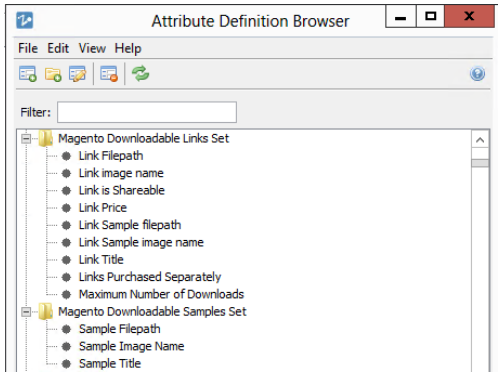


and one or more download samples:

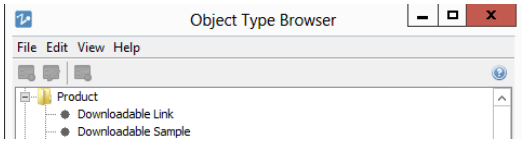


We need to set up equivalent data in Agility to be able to automatically create Magento downloadable products from Agility data using the plugin.

First create 2 new attribute sets, one for download links and the other for download samples. The attribute sets should look as follows:



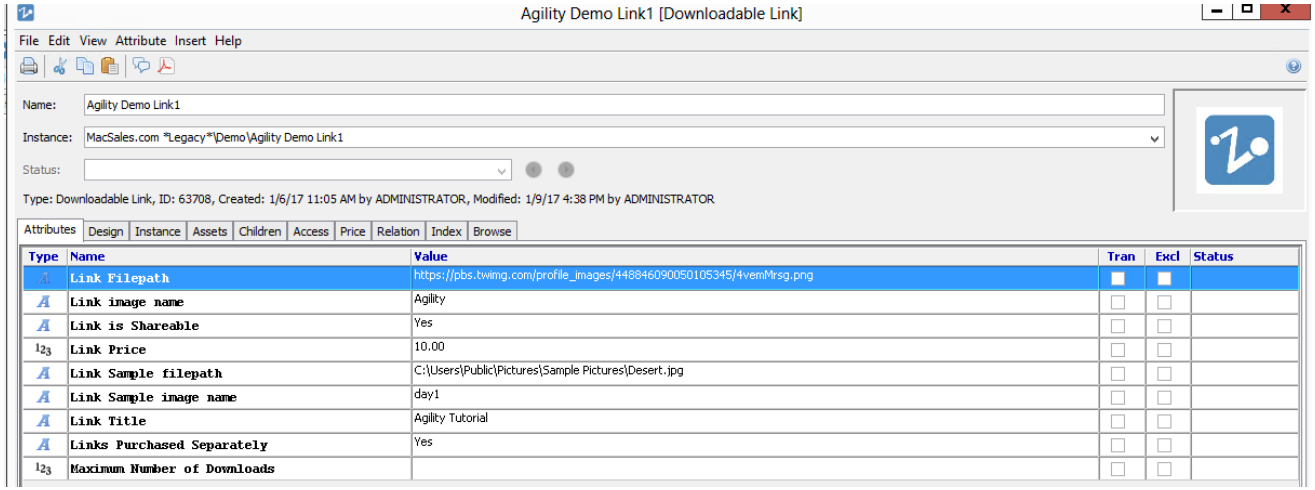
Then create 2 new product object types, one for download links and the other for download samples, each using their respective attribute set.



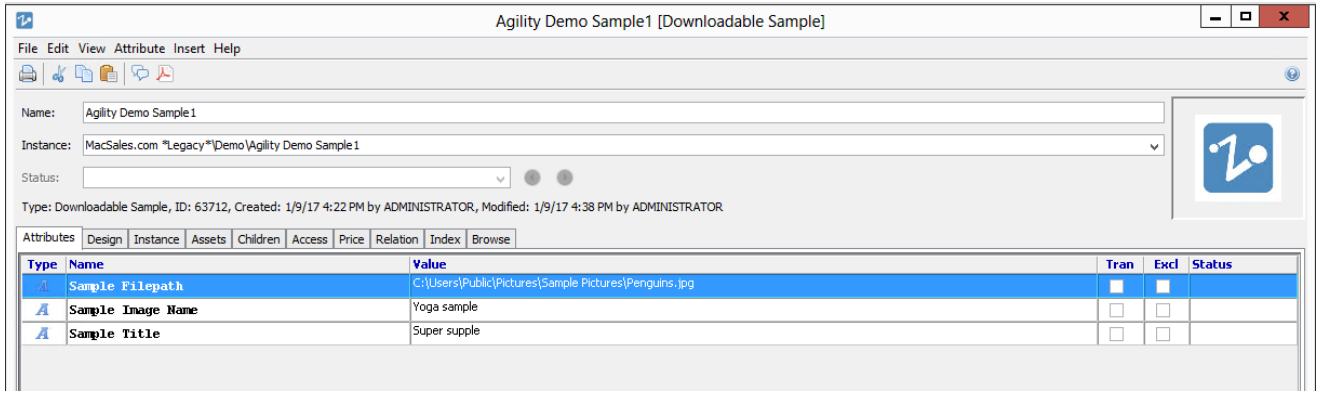
Then create the download link and download sample objects and populate with data.

Note: Set up the download links and samples products in a different structure that won't be syndicated.

A download link product object example would look like:

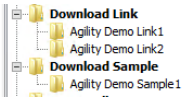


A download sample product object example would look like:



Once they are created, you can relate the objects to a product that you want to be downloadable. Use new product relation attributes, one for download link and the other for download sample. Ensure that the 'Allow Multi instance' check box is ticked on each of the new product relation attributes, as a product can have more than one download link and sample.

A product with related downloadable links and samples would have them showing on the relation tab as:



Once this is set up, you are ready to use syndication to extract the downloadable products for input to Magento.

Agility attributes mapping to Magento attributes

This article describes how attributes from Agility are added into the Magento eav attribute table.

The Agility Magento connector code reads through all of the attributes on the xml file from syndication and adds them to the eav attribute table if they do not already exist.

Syndication for attributes should be set up to have a 'type' value as this is used to assess the type of attribute being passed and how it should be set up in Magento.

Note: Magento will not create an attribute that has a name longer than 30 characters. So the attribute name should be less than 30 characters.

If you want to create a new attribute in Magento, camel case the attribute label on the syndication file. For example, an attribute with a name of 'newAttributeName' would be created in magento with an attribute code of 'newattributename' and a front end label of 'New Attribute Name'.

Text attributes

Text attributes from Agility (type 2) will be written to Magento with a backend type of 'varchar'. If the length of the attribute is greater than 50, the frontend input type will be a textarea, otherwise it will be text. (Textarea is a text box, text is a single line box). If the text is a selection of choice list values, this will be processed as a multi select choice list, with a frontend input type of multiselect. All option values within the choice list are created as new option values in Magento if the option does not already exist.

Multichoice options

If an attribute contains a choice list value, the choice list option values are loaded into Magento if they do not already exist. Choice list options have both a value and an attribute id within Magento, and are loaded on to the eav_attribute_option and eav_attribute_option value tables.

Date attributes

Date attributes from Agility (type 3) are written to Magento with a backend type of datetime and the front end input is date.

Numeric attributes

Numeric attributes from Agility (type 1) are written to Magento with a front end input of text, and a back end type of decimal if the value contains a decimal point, otherwise it is int (integer).

Path attributes

Agility path attributes (type 14) are written to Magento with a front end input of text and a back end type of varchar.

eav_attribute table

Attributes are added to the eav attribute table with the following values set:

```
  $this->attributeObj->setData(
 [
    'attribute_code' => name of the Agility attribute,
    'entity_type_id' => $installer->getEntityTypeId('catalog_product'),
    'is_global' => (\Magento\Eav\Model\Entity\Attribute\ScopedAttributeInterface::SCOPE_STORE),
    'is_user_defined' => 1,
    'frontend_input' => front end input type as described above,
    'is_unique' => 0,
    'is_required' => 0,
```

'is_searchable' => 1,

'is_visible_in_advanced_search' => 0,

'is_comparable' => 0,

'is_filterable' => 1,

'is_filterable_in_search' => 0,

'is_used_for_promo_rules' => 0,

'is_html_allowed_on_front' => 1,

'is_visible_on_front' => 1,

'used_in_product_listing' => 1,

'used_for_sort_by' => 0,

'frontend_label' => *front end label using a modified version of the attribute name*

'backend_type' => *back end type as described above*

]

The attribute scope is set to store level. The newly created attribute is added to the set 'catalog_product', 'Default', 'General'. When a product is created, it uses the ‘default’ product attribute set.

Agility ID attribute

All attributes are added to the eav attribute table on the fly as part of the Agility connector processing, with the exception of the agility id attribute, which is added as part of the project install. This is because the agilityid attribute is a known constant which must exist whereas all other attributes are variable.

The agilityid attribute is added for products and categories as part of the Agility Import install to Magento process.

All products, variants and categories from Agility are tagged with the agility object id as an attribute when written to Magento.