Scandiweb

Case study

Dogstrust ERP integration

Task description

www.dogstrustproducts.com is a company that produces cosmetics for dogs. They have a lot of not very expensive products that they sell a lot (they have sold about 15000 items in the past 8 months). Delivery of all orders is being handled by a fulfill-ment agency (Prism) with their own ERP system.

Our task was to export orders to the ERP system.

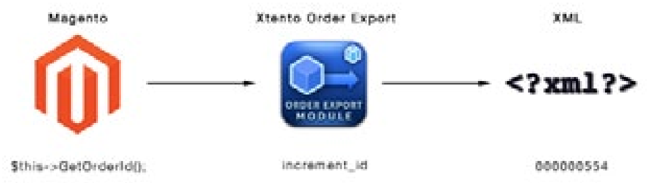
Information provided at the very beginning of integration process

1. ERP integrators contacts
2. Order export sample XML schema
3. Xtento order export Magento extension license
4. Folder structure on our server - where we should put xml with orders and where we will receive updates
5. Scope of work:
   5.1. Scandiweb: Install the order export extension
   5.2. Scandiweb: Create xsl template (configure the extension)
   5.3. Prism: set up xml file download from project server
   5.4. Prism+Scandiweb: test the integration

Step 1 : Order export

There are at least two ways to implement order export:

1. Custom code order export. This will take approx. 3-4 days of development + it will be difficult to amend XML schema in future.
2. Use Xtento order export extension - it costs EUR 149 and will require approx. 3 hours for extension installation and XSL template development. Basically what it does is mapping between Magento internal variables (like SKU, product names, order numbers, etc) to its own variables that then can be inserted in XSL template and output to XML



1.1 Extension installation

To export orders from Magento we have used Xtento Order export extension: http://www.xtento.com/magento-extensions/magento-order-exportmodule.html.

Extension installation itself takes several minutes, just like any other correct extension - just upload files to Magento via FTP and log out / log in to Magento backend.

The main requirement from the ERP side was to export orders at the moment they were placed, so that each XML file contains one order. The extension does not allow it by default, but luckily Xtento staff (Sebastian from Germany) is very helpful and responds really quickly, so we had no problem with modifying the extension to fit our needs.

Extension configuration is done by XSL template - file that describes how the output XML will look like, how xml file will be named and where it will be stored in filesystem.

Customer has provided us an XML schema and file name structure - this is all that is required to create the template.

1.2 XML Schema Analysis

Before creating the XSL template it was essential to understand xml structure and what values should go where. Just like PHP coding - before writing actual code you have to understand the logic, algorithm that you will implement.

One of the huge XML benefits comparing to other data transfer methods is that XML structure is human-readable - you do not need any special software to understand in general what xml document is about.

However, after looking through XML we saw several fields with unclear meaning, for example :

```
<MediaCode>W111</MediaCode>
<RentOptOut>NO</RentOptOut>
<EmailOptOut>NO</EmailOptOut>
<MailOptOut>NO</MailOptOut>
<CCTransactionCode></CCTransactionCode>
```

So we emailed Prism support and asked what should we put in these fields. Turned out that these are some generic field that must be present in the XML file - this was their ERP system limitation. However, when processing actual orders from Magento these fields will not be taken into account.

It was decided that we can just hard-code static values to the template so that these will be the same for all orders.

1.2 XSL template development

XSL template describes how the output XML will look like. It is somehow similar to Magento template files - you can put XML tags and describe what value must be placed inside tags.

Assume you want to produce the following xml (000000554 is real order number):

```
<OrderNumber>000000554</OrderNumber>
```

Lets think about xml as a text document. Starting tag <OrderNumber> is just a piece of static text - it is string that is always present, it is always exactly <OrderNumber> and it is always in the same position in the document.

In PHP you would write this as <? php echo '<OrderNumber>'; ?> - no variables, just line of text.

In XSL the same line will look as follows:

```
<xsl:text disable-output-escaping="yes">&lt;OrderNumber&gt;</xsl:text>
```

The next part of the line is order number - "000000554". This is variable, because for each order it will be unique - so we can not just hard-code one number in the template.

In PHP you would write something similar to <?php echo $order_id; ?>, where $order_id is variable that stores order number value.

In Xtento order number is contained inside increment_id variable. XSL code to output this variable value:

```
<xsl:value-of select="increment_id"/>
```

The last thing is to add closing tag - </OrderNumber>. This is again just a static text:

```
<xsl:text disable-output-escaping="yes">&lt;/OrderNumber&gt;</xsl:text>
```

XSL templates support foreach loops - it is very useful when you need to export dynamic content like items (order can contain just one item, two, three...).

In php you would write foreach loop as <? foreach($items as $item){ ... }; ?> where $items is array of data and $item is one element of that array.

In XSL for Xtento module it will look like this:

```
<xsl:for-each select="items/item">
...
</xsl:for-each>
```

Items/item is reference to single item - single product - in array of products within one order.

For example, to export all product SKUs from one order, you can write the following code:

```
<xsl:for-each select=”items/item”>
    <xsl:text disable-output-escaping=”yes”>&lt;OrderLine&gt;</xsl:text>
        <xsl:text disable-output-escaping=”yes”>&lt;ProductCode&gt;</xsl:text>
             <xsl:value-of select=”sku”/>
             <xsl:text disable-output-escaping=”yes”>&lt;/ProductCode&gt;</xsl:text>
    <xsl:text disable-output-escaping=”yes”>&lt;/OrderLine&gt;</xsl:text>
</xsl:for-each>
```

If order contains two products - SKU-1 and SKU2-2 - then produced xml will look like this:

```
<OrderLine>
     <ProductCode>SKU-1</ProductCode>
</OrderLine>
<OrderLine>
     <ProductCode>SKU-2</ProductCode>
</OrderLine>
```

All order values are well-structured - this is described in the Xtento extension documentation - they provide XSL template with all possible fields. For example to get shipping address city, you should refer to **shipping/city** value. For billing address the same value will be **billing/city**.

Step 3 : File transfer and processing

This step was out of our scope of work - it was completely set up by Prism.

To import/export files they are using SFTP protocol that is easy to use.

For order export every minute Linux crontab job checks order export folder and if there are new files - it transfers them via SFTP to the ERP system. After the file has been sent it is moved to processed orders folder, so that order export contains only NEW orders.

Faced problems

1. Slow ERP users response time. When we were starting the project we did not estimate the fact that ERP integrators on the customer side will respond on our email in several days instead of several hours. Because there were not much work for our developers, it resulted in long communication and overtimes, because by the time ERP guys answer emails our developers were already working on another projects.
2. Unclear XML structure. Some fields like CCTransactionCode are not present in Magento and it was necessary to clarify whether it is OK to leave these fields blank, put there some static value or just remove from XML schema.

3. Communication through third-party people. For some reason ERP integrators (although we were introduced in the very beginning of the project) never contacted us directly. They were sending emails to our client, to the end-client but not to us directly. This added some additional time lag in the communication, as well as it led to some miscommunication in the middle of the project, because a couple of our emails were just lost.
4. Due to poor communication we have missed out is that stock lists (product SKUs) are never synchronized - this might lead into situation when shop is selling products that does not actually exist in the warehouse and vice versa.
5. Originally task scope was also to import stock updates from ERP to Magento, but due to poor communication this was never implemented. We have sent questions on this topic to ERP guys but never heard back from them.

Conclusion

Although this type of integration is probably one of the easiest, there are some underwater rocks:

1. Communication. Integration heavily depends on the communication speed, because when you are connecting two systems you can not do it just one way - there is some work to be done on both sides. Really due to slow communication speed in this project we have lost approx. 2 weeks of time and after that had to work on Saturday/Sunday full-time to meet the deadline.
2. Communicate directly. If you are writing to your client emails that begin with “Please forward this to your integration specialists”, you are going to have a bad time because of time lag and miscommunication.
3. XML data. It is essential to understand what data will go in which order export XML field - without this understanding you will not be able to create correct XSL template.

Author	Scandiweb.com (info@scandiweb.com)
Category	Training materials
Title	Case study: Dogtrust ERP integration
Copyright	(c) 2013 Scandiweb.com, Wonderland Media LTD (http://www.scandiweb.com)
License	License: http://opensource.org/licenses/afl-3.0.php Academic Free License (AFL 3.0)