Creating Multilingual Data for WebSphere™ Commerce Suite via WebSphere™ Catalog Manager

Zhang Yu
Globalization Certification Laboratory
IBM China Development Lab, IBM China

August 2002

***Abstract**:*

This paper first states that why we need multilingual business and two types of multilingual business, and give a high level view about the capabilities of WebSphere Catalog Manager(WCM) and WebSphere Commerce Suite(WCS) to support catalog data, then explores a typical process of creating and managing catalog data via WCM. But WCM doesn't have the capability to handle most multilingual data at one time. so some actions have to be taken to make up for the drawback. Lastly it will present different options on how to create multilingual data to meet the requirements of different types of Global stores.

Index

1. Introduction

1.1 Intended Audience

The target readers of this article are those with basic knowledge of the WCS and WCM and those who intend to develop multilingual Commerce Suite stores using WCS and WCM.

1.2 Why Do We Need Multilingual Business and Multilingual data

Nowadays online merchants have to offer much greater levels of customer service and support than ever before. When customers visit an e-commerce Web site, they expect, at a minimum, to do shopping in the language and currency of their choice. Commerce Suite stores can support multicultural features, making them more useful in an international environment. Merchants can develop stores with multicultural online product catalogs, allowing customers to view descriptions appropriate to their locale in their preferred language and currency. In other words, if this information has been entered during the store development process, customers can select cultural parameters such as language, tax and shipping rates, date and currency format, and payment methods. Thus supporting multicultural features' stores have been more and more popular.

2. Types of Multilingual Business

Generally there are two types of organizing multilingual data for online stores.

The first type is similar to the store "In Fashion", a demo sample in WebSphere Commerce Suite product package. In this type of business, when a user visits the products data, the products' description will be translated to the user's preferred language. For example, when visiting the site, an American user may hope to see the English descriptions of products, while a Chinese user may prefer the descriptions in Chinese. Of course, besides the data displayed, cultural convention, currency format, and date are also considered.

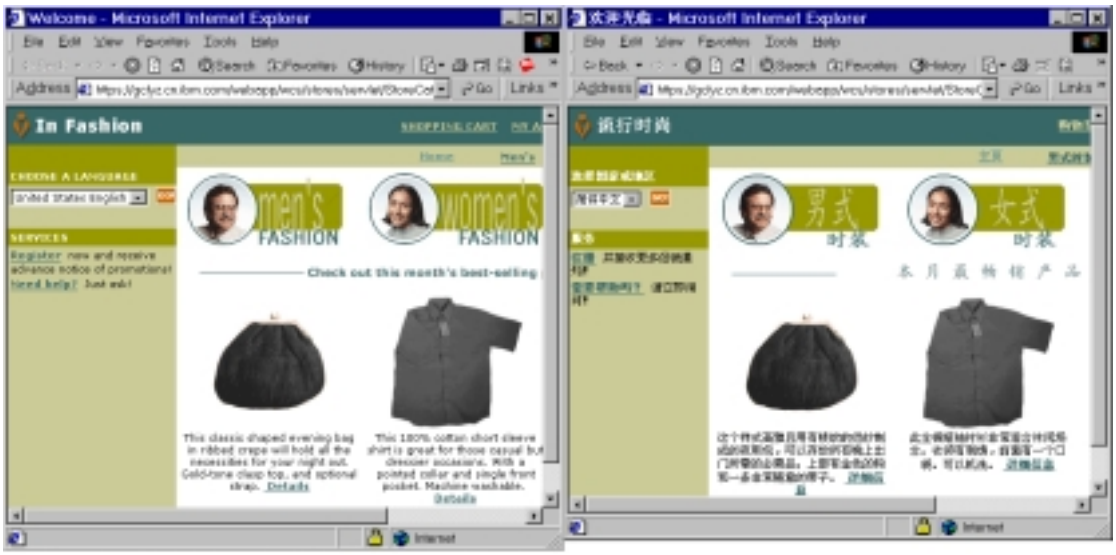

Figure 1 In Store 'In Fashion', different language users will see production's descriptions in different language

In the second type, all users will see multilingual products' descriptions and the Web pages may look the same to all users, except for the culturally sensitive data, such as currency format, date format, and so on. The descriptions of products aren't translated into the user's preferred language, instead, they remain the way as they were initially input.

A multilingual bookstore is typical example of this type. Details may be referred to the Global Bookstore, a demo site produced by Globalization Certification Laboratory of IBM. In this store, users will see that the descriptions of different products/books are in different languages. These multilingual descriptions can imply which language the book is written in. For example, if a Chinese user wants to buy the book *Integrating XML with DB2 XML Extender and DB2 Text Extender*, the store will return him the list of the same book, but in different language versions: Simplified Chinese, English, and French. If the user understands English but not French, he can choose the book written in either English or Simplified Chinese.

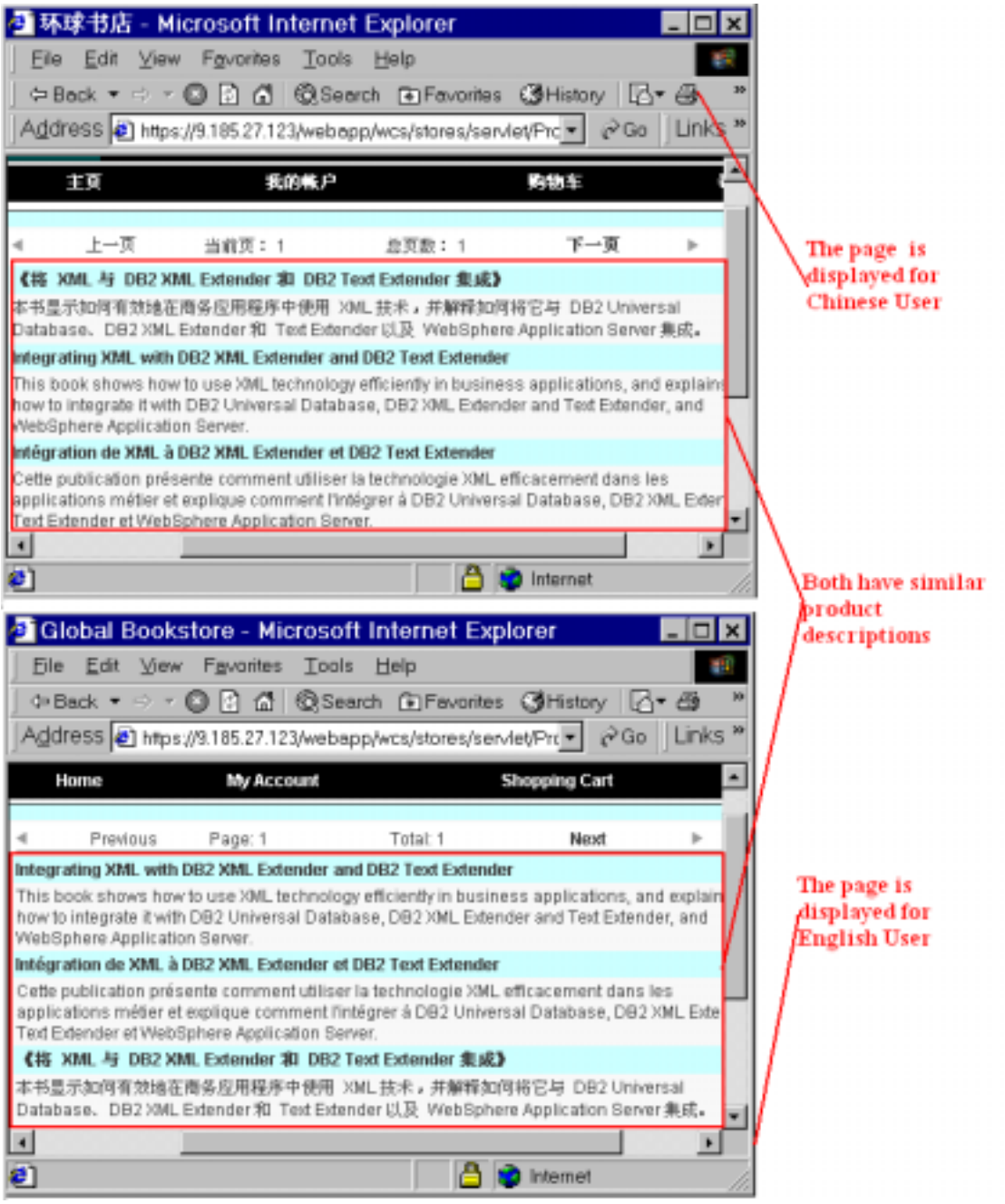

Figure 2 Different language users will see similar product's description.

3. The Capability of WCS to Handle Multilingual Data

However, enabling a commerce site for a global customer base is not simply a matter of translation or formatting. Instead, it is the ability to properly interact with customers and to engage in business activities with people according to their cultural norms. WebSphere Commerce Suite v5.1 recognizes the importance and need for sellers to go global and thus has multicultural enablement functionality built right into the design of the product. The base database architecture and storage model has been fully enabled so that multicultural data can exist in one central repository. An interface

or business objects layer has been constructed to allow sellers full flexibility in terms of the retrieval and formatting of cultural and linguistic data. Rules logic has also been incorporated to allow sellers to model real life business logic into their site which is also accompanied by a fully multi-culturally enabled tool set.

3.1WCS Can Handle Multilingual Data

In order to better understand the capability of WCS to handle multilingual data, the data structure of WCS database will be introduced as below:

An immediate advantage provided by WCS at the least complex stage of multicultural enabling is that data is stored in UNICODE. This allows the database to properly handle data regardless of language.

Another advantage is that linguistically and culture-sensitive data are fully separated from the culture-insensitive data within the database design.

Traditionally, when we need to input product info into a database with multiple descriptions in multiple languages, we may need to input it many times, as we can see in the following illustration:

Figure 3 Three products with different language.

The multi-culture-enabled database design of WCS allows sellers to easily enter and manage cultural data apart from non-cultural data. Multiple data entries concerning the same product can be compacted into one single data entry with multiple category descriptions, as we can see in the following illustration:

One product

	catentry_id	member_id	catenttype_id	partnumber	mfpartnumber	mfna...	markf...	
1	10001	-2000	ProductBean	sku-@product_id_102	sku-@product_id_102	InFashion	0	...

With three descriptions in different language

	catentry_id	name	language_id	shortdescription	longdescription	fullim...	thumb...	
1	10001	cords	-1	Cords	These traditional 5-pocket button-fly cords are light-weight cotton and feature wide-ribbing. Perfect for spring and fall, they are machine washable.	images/mens_pants_cords.gif	images/mens_pants_cords_sm.gif	...
2	10001	灯芯绒裤	-7	灯芯绒裤	这些具有5个口袋的传统灯芯绒裤是用非常轻的棉制成的，棱纹较粗。最适合春季和秋季穿，可以机洗。	images/mens_pants_cords.gif	images/mens_pants_cords_sm.gif	...
3	10001	Cordhosen	-3	Cordhosen	Diese traditionellen 5-Pocket-Cordhosen mit Knopfverschluss sind aus leichter Baumwolle im Breitcorddesign gefertigt. Sie sind perfekt für Frühling und Herbst und können in der Maschine gewaschen werden.	images/mens_pants_cords.gif	images/mens_pants_cords_sm.gif	...

Figure 4 One product with three descriptions in different language.

In this way, the amount of duplicate data stored in the database can be reduced, allowing smoother data management. To an online store, the more languages it supports in product description, the more convenient the structure will bring.

There are many tables in WCS database, but generally these tables can be divided into two types. The first type includes some product entry information, while the other holds culturally-dependent information related to the product entry. For example, the CATENTRY table holds the information related to a Catalog Entry which may include ID,CATENTRY_ID, MEMBER_ID, etc. Examples of Catalog Entries are Products, Items, Packages and Bundles. Contrasted with CATENTRY, the CATENTDESC table holds the language-dependant information to a Catalog Entry, in which one important field is the LANGUAGE_ID. If the value of LANGUAGE_ID is CN, the other fields, such as the description related to language-dependant information will be Chinese characters. In one word, the second type of tables is mainly used for data display and the first type of tables deals with the logic or control to user data. Moreover, in this multilingual data enabled database, if one record describes one product entry in the first type of table, there may be a few records describing the corresponding product entry in the second type of table and the records will have similar description but in different language.

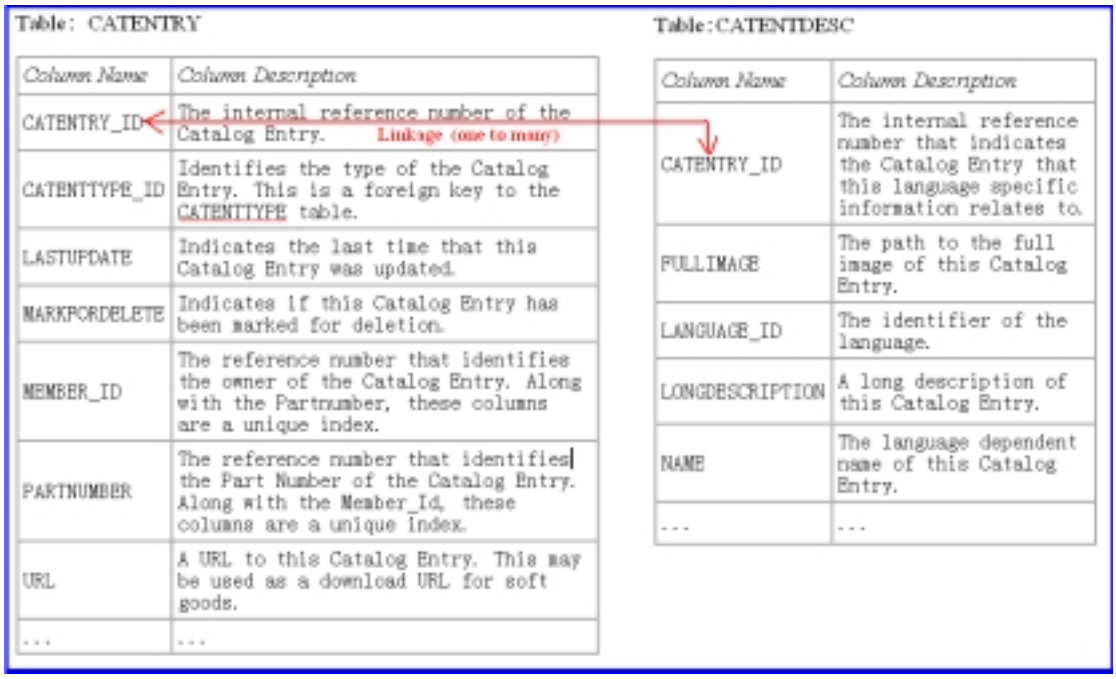

Figure 5 CATENTRY and CATENTDESC table

3.2 The Store Archive(.sar) Structure

In WCS, the fastest and easiest way to create an online store is to use the sample store archive, in-fashion, provided with Commerce Suite. A store archive is a compressed file that contains all the assets necessary (including web and database assets) to create a store. Usually it includes: file assets, store database assets, payment assets and a description. The following is the physical structure of a store archive file when decompressed:

- SAR-INF
  - Contain the sarinfo.xml file, which describe the store archive.
- Data
  - Contain all the store database assets:
    - The following elements which correspond to the first type of tables in WCS database include all store data with non-locale-specific description.
      - Campaign
      - Catalog
      - Currency
      - Offering
      - Shipping
      - Store
      - Tax
      - ……
    - Locale-specific folder(s)
      - If the store supports multilingual description, there will be some

locale-specific folders in the same directory as these products' entry information. One locale-specific folder may include the following XML files which contain these products' entry description in one language. The following elements(XML files) which correspond to the second type of tables in WCS database include language-dependent information.

- Campaign
- Catalog
- Store
- ……

- Property resource bundle
  - Contain the translatable text files for the store
- Webapp
  - Contain all the web assets for the store, including locale-specific files.

In order to save multilingual descriptions, store database assets take the form of well-formed, XML files valid for handling data. For example, the 'store' XML files are located in root directory and locale-specific directory. In the root directory the 'store' XML file includes the information which represents a store entry, such as the owner of the store entity, the store's type, and store's ID, without any description in locale-specific language. However in the locale-specific directory the 'store' XML file contains language-dependent information about a store entry, such as the description of StoreEntity.

In the past, globalizing a Web commerce site would be a difficult and lengthy task, but now, with WebSphere Commerce Suite, the required effort has been reduced to the point where it is almost transparent.

4. The Capability of WCM to Handle Catalog Data

Next we have to face how to create and manage these multilingual store data with WebSphere Commerce Suite(WCS). To our luck, IBM WebSphere Catalog Manager(WCM) provides a broad set of generic tools for WCS catalog management. It is designed to understand the inherent relationships between different catalog elements such as products, categories, SKUs or items, and catalog relationships. The tools enable users to create, manage, import, and edit large amounts of catalog data. Content contributors from within the same enterprise or across enterprises can readily create and input data into an aggregated catalog.

WCM server can accept data in the form of XML data files, which has fast become a standard in the marketplace. Since data comes in many different formats, WCM provides means to accommodate multiple types of data input and to transform them into XML files that map to the needed schemas of the specific target system.

WCM server transform tools enable users to convert data from spreadsheet ASCII files into a generic XML format and back again as needed, and enable data mapping in XML files from one schema to another. This ability provides great flexibility in accommodating different target systems (including WebSphere Commerce Suite, Pro Edition, Start Edition, or MarketPlace Edition) that use very different database schemas. It provides support for customized versions of the out-of-the-box database schemas provided by a variety of WebSphere Commerce Suite editions. Next the typical process to create catalog data using WCM will be introduced.

5. The Typical Process of WCM to Create Catalog Data

In the first phase of implementation, the basic store Web page, logic and control will be created using WCS Store Developer, and the catalog data will be created by WCM. The tools provided by WCM are enhanced to make management and creation of catalogs significantly easier and more efficient for users than the simple mass import utility.

The following is the typical process to publish catalog data to a WCS database.

Step 1. Creating XML File Containing Catalog Data

Normally there are three methods to create XML files containing catalog data:

1. Create XML files from spreadsheets in CSV format. At present WCM can accept almost all spreadsheets, including Lotus 123 and Microsoft Excel, which can all be saved as CSV file. We can first manually input catalog data into spreadsheet or get the spreadsheet from other applications, and then use a WCM tool named Text Transformation to process the transformation of data from the spreadsheet into generic XML format.

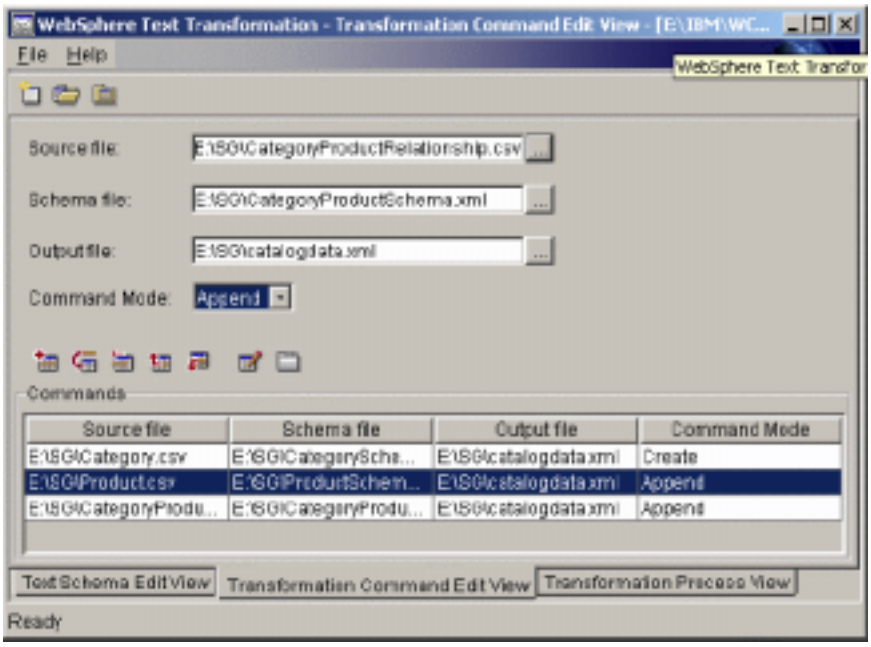

Figure 6 Transformation Command Edit View

2. Create XML file with WCM Client. We use Windows-based WCM client as an advanced spreadsheet-like tool to create and manage catalog data quickly and easily. Then he WCM Client can export an XML file including these catalog data.

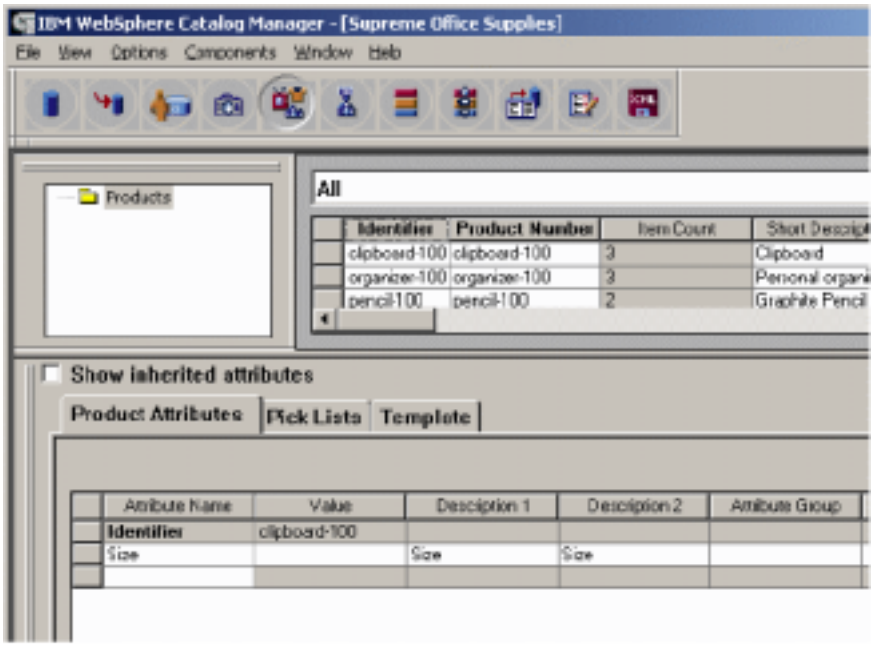

Figure 7 WebSphere Catalog Manager Client

3. Directly create XML file. We also get these catalog data in generic XML format which may be exported from other application or directly from input.

Step 2. Converting XML File

Normally, XML data is not loaded into an actual production WCS database right away. Since the XML file generated from the previous steps with different methods is in generic format and the target database need the specific XML file. So in this step, the XSL Editor tool will be used to convert data into an XML file to be formatted for the target database schema. The main step is to set the XSL mapping rules to do the transformation.

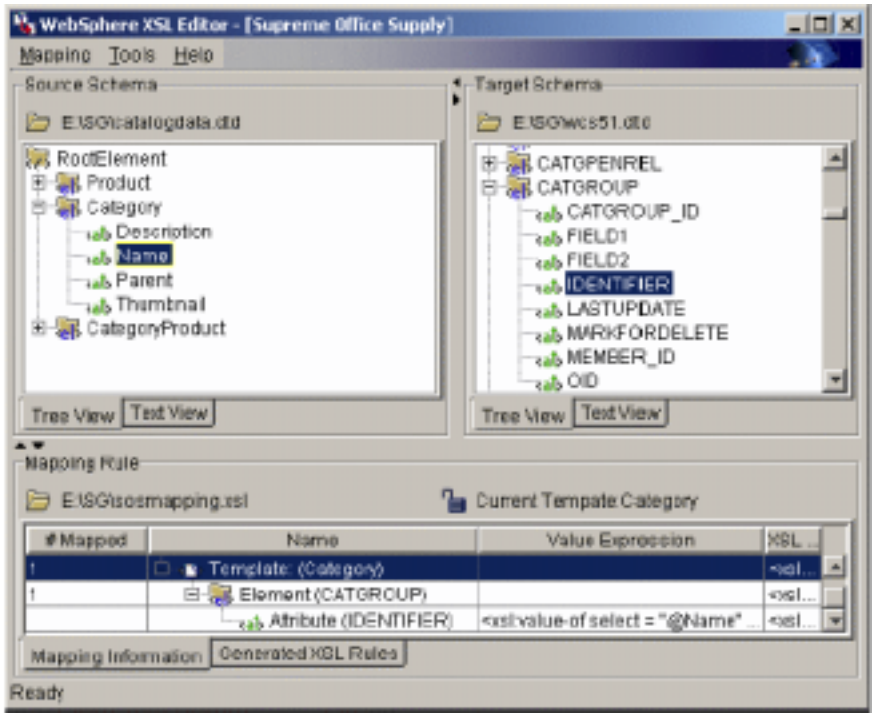

Figure 8 XSL Editor Main Window

Step 3. ID Resolving

After the format of the XML is consistent with WCS database, some additional steps have to be run before the data can be imported into the database. The main step is resolving the ID. Because the XML files map directly to the target database schema, they must include the identifiers required. Normally many tables include the primary or unique key, so duplicated records cannot be imported into the database. In WCS database these key fields don't include any detailed catalog entry information, and only indicate the unique record better for data management. So before the catalog data can be imported into the database, we need to give reference numbers to some of the elements in the XML files. These identifiers link data in the XML file with the database. The ID Resolver component of the WCM updates XML elements with their

associated identifiers to ensure that every element has a unique reference number if needed. The ID Resolver can generate identifiers for new data to be loaded into WCS database. This component can also resolve identifiers for existing data before the Loader is invoked.

To determine an identifier, the ID Resolver uses either of the two methods: unique index resolution or internal alias resolution. Unique index resolution uses any specified unique index on a table as a means of determining an identifier. Internal alias resolution places an alias in the primary key attribute (identifier). This alias can be used throughout the XML file to refer to that element.

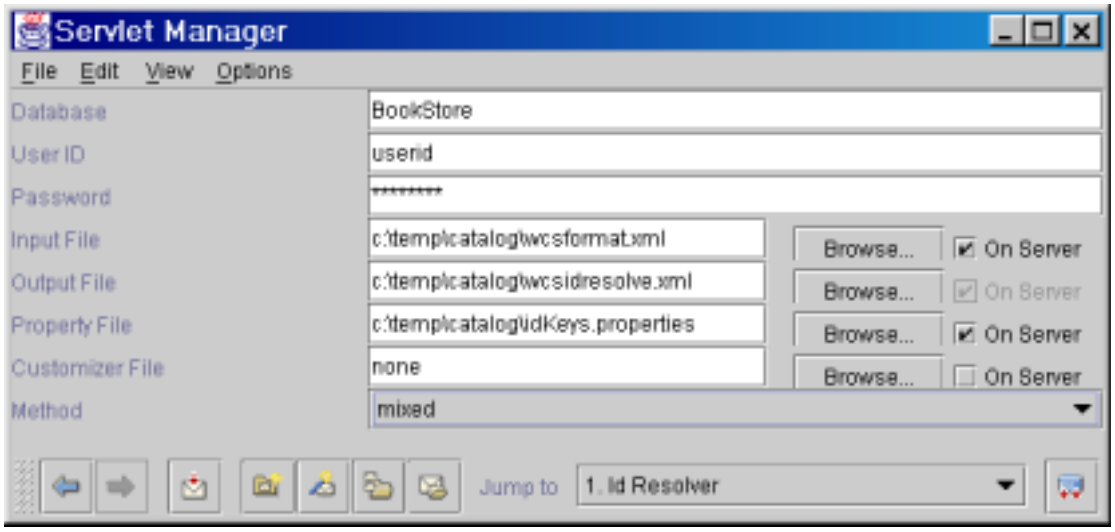

Figure 9 ID Resolver Main Window

Step 4. Mass Loading

After the structure of these data in the latest XML file have been consistent with that of WCS, the WCM Mass Loader tool will be used to import these data into the WCS database. The tool will load large amounts of data and update data in users' Commerce Suite database. The Loader utility uses valid and well-formed XML as input to load data into the database. Elements of the XML document will map to table names in the database, and element attributes will map to columns of tables.

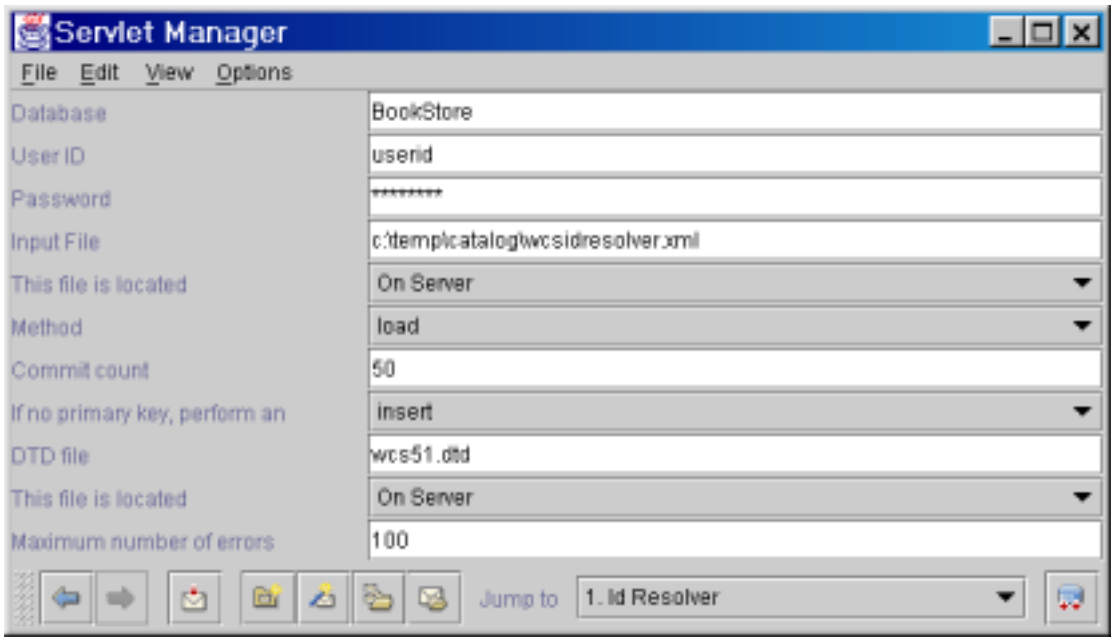

Figure 10 Mass Loader Main Window

Step 5. Web-browser-based Catalog Data Management

After the catalog data have been deployed into the database, users can manage data through Web browser interface. The WCM Web editor enables users to create, delete, or make fast and simple changes to their catalog data through a Web browser. The WCM Web editor is a Web-browser-based application, making it an ideal tool for updating small quantities of data, and for managing catalog information by distributed end users with Web access.

6. The Limitation of WCM on Importing Multilingual Data

WebSphere Catalog Manager has some limitations on creating multilingual data. In general, WCM emphasizes how to create, manager, import, and edit large amounts of catalog data. It doesn't care what data it will deal with. The national language version of WCM only supports its own language data besides English data, and doesn't take multilingual data into account. So if we want to create multilingual data for WCS via WCM, we have to adjust it to meet our requirement.

In Step 1 of the typical process mentioned above, the first method states that these catalog data is saved in spreadsheet, but when they are exported into CSV file, each data entry and its description are in one line. The reason is that this data structure of the CSV file cannot allow a data entry to have more than one description, and otherwise it will become two data entries. So it's impossible to prepare multilingual

data descriptions for one data entry in this step. Moreover even if multilingual data are included in the only one description for each data entry and the file is saved in UTF-8, after the CSV file is converted to an XML file using WCM Text Transformation, due to the limitation of the tool, some garbage characters will be found in the XML file and multilingual characters cannot be correctly shown at the same time. Therefore the XML file cannot include all multilingual catalog data.

The second method in Step 1 states that we can use WCM Client to manage catalog data. Since the WCM workstation client enables users to create, import and edit product data, establish catalog structure, and product definitions, create product relationships and merchandising associations, and validate and correct that information through the WCM workstation client's GUI tools, this GUI provides users with a familiar spreadsheet interface and a graphical means to enter and to manage product information. However, one data entry can have only one description. As a result, a product cannot have multiple descriptions in multiple languages. We cannot directly import the XML generated by WCM into multilingual store since it doesn't include multilingual data descriptions.

In Step 1, only the last method that we create an XML file manually to include catalog entry data can support multilingual characters, since the XML file can correctly handle Unicode characters. However, these data entries are saved in one XML file and there is no tree structure like an Store Archive(.sar) file to store multiple descriptions in different files. Therefore in this file each data entry only has one description and multiple descriptions cannot be stored in this XML file.

So after Step 1 is completed, we cannot get an XML file to meet the first type of business requirement, such as the store "In Fashion". Only the XML file created by the third method of the Step 1 have the possibility to meet the second type of business requirement, such as Global Bookstore, since the XML file includes such data entries with each one having one multilingual description.

However, though this XML file can include multilingual characters in its description, when the following steps are used, the XML file in generic format will be converted to a specific XML file that the target database needs through the XSL Editor. The WCM tool will need a user to provide a language ID for the data entry to identify the language of the description, but we can only provide one language ID which corresponds the LANGUAGE_ID field of the records included these data descriptions stored in WCS database. Though these data descriptions are same in all languages except for the LANGUAGE_ID, we still have no chance to copy the descriptions in one language into the data entry for other languages.

Because of the above reasons, we know we cannot simply follow the typical process to create multilingual data stored for WCS, we have to make some adjustments to make the store data meet the globalization requirement.

7. Our Opinion about How to Create Multilingual Data for WCS

Considering the limitation of WCM on supporting multilingual catalog data, we will enrich the typical process to meet the globalization requirement. As mentioned above we cannot use the tool of WCM to load the XML directly to database, because the single XML file cannot include multiple descriptions for each data entry in different language. Our solution is that the WCS load XML files compressed in a Store Archive(.sar).

The Store Archive(.sar) includes not only multilingual catalog data, but multiple descriptions for each entry data, because the Store Archive(.sar) has a tree structure and places catalog data and its descriptions at different levels, and different lingual descriptions in different directories. Therefore the XML files in a Store Archive(.sar) including multilingual catalog data can be imported to WCS database at one time through loader tools of WCS.

As a result, we will utilize some of the main components of WCM (such as the XSL Editor, ID Resolver, etc.) and other necessary methods to create multilingual data in a Store Archive(.sar) which also includes other data resources such as Webapp, Property resource bundle, etc. And then we deploy it to the WCS database through the WCS tool. After these multilingual data have been stored in WCS, we may manage them through the Web Editor tool of WCM.

As mentioned above, there are two types of business that need multilingual catalog data, so different solutions will be introduced based on these two types as below:

7.1 The Solution to the First Type of Business

In the first type of business, every user browses his/her language-specific category data. We will first follow some steps of the typical process to create catalog entry data. After the XML file has been converted to WCS format and has completed the ID Resolving, we divide the single XML into several small XML files which will be one part of a Store Archive(.sar) according to its structure. Next these data's descriptions will be translated into different languages and saved in some new XML files, and then we place these files in proper position following the Store Archive(.sar) structure and add these XML files and other resources to a Store Archive(.sar) file which can be deployed to WCS correctly. The following are the detailed steps:

Step 1. We will use each one of the three methods of Step 1 of the typical process to

create single language category data and its description as below:

<1>Spreadsheet
WCM can accept many types of spreadsheets because almost all spreadsheets support the extraction of data as CSV file and WCM can accept the import of this format file. This will bring us much convenience, especially on utilizing some spreadsheet format data which is exported from other application.

Internally, WCM manipulates and manages data in XML file format. So when data is contributed in CSV format, the first activity to be performed within WCM is to convert it into XML file. The text transformation tool will be used to map the data that is specified by field position in a spreadsheet or CSV file into a self-describing XML form. The tool that performs this transformation uses a set of configuration files that specify the tags or XML elements to be specified for each field position in the file.
Each configuration file which we will create before the transformation is a specific type of XML file. For more detail refer to WCM Solution Guide.

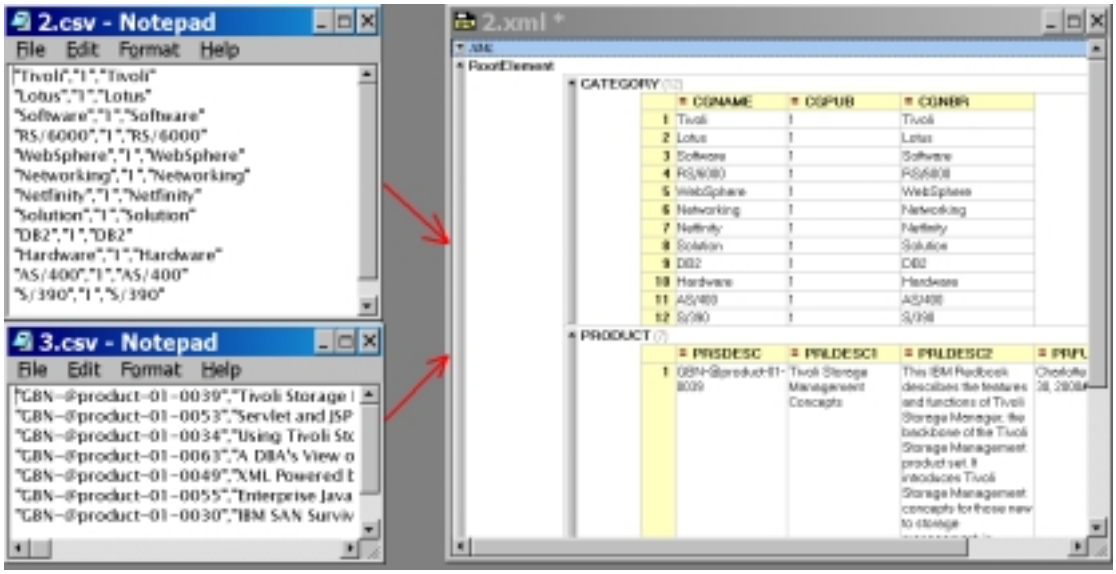

Figure 11 txt files transform to XML file

<2>WCM Client
WCM Client provides a friendly UI to help user create catalog data, which uses inheritance to keep data unique. Inheritance ensures consistency, correctness and completeness of catalog data and also expedites data entry and maintenance because data can be entered once for many objects using the product component. After finishing these data input, these data can be exported into an XML file which will be used in the next step.

<3>Generic XML
If users create catalog data and save them in a generic XML file, The XML file is also easy to be converted into WCM-specific XML file through the WCM XSL Editor. So

any format of XML file can be accepted.

Step2. The output of the ASCII-XML transformation is a very simple XML document form, in which explicit XML tags that describe the field or data within that row replace the field positions of the spreadsheet.

Generally, the simple output format from the spreadsheet will not map directly to the input format of the database tables within WCS. Normally the way business users will view the content and the way it will be mapped in the actual schema can be very different. WCM provides a rich XML-XML transformation and mapping engine named XSL editor, that we will use to transform the format of the input XML to output XML expected by the tool. Before the transformation, we also need to prepare a mapping rule which is an XSL file for the transformation. Different format of XML file need different mapping rule. The following is one example of mapping:

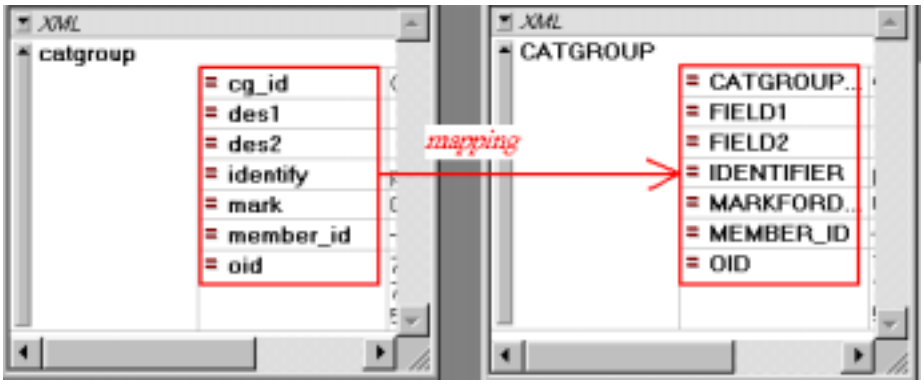

Figure 12 From one generic XML format to the specific XML format

Step3. WCS uses internal reference numbers to uniquely identify much of the data stored within its internal database. These reference numbers represent keys within the database that relate information across tables. However, it is required that these reference numbers be created and maintained by all data that is inserted into WCS. Because these reference numbers are implementation details of WCS, they will normally not be information that business users contributing data to WCS will need to be aware of. Therefore, WCM does not want to impose reference numbers in its content contribution processes that the business user needs to be aware of these. This is the purpose of the ID Resolution component within WCM. This component takes the XML data that is going to be submitted to the loader and creates or resolves the reference numbers needed to maintain the referential integrity of the WCS database.

Once the spreadsheet data has been converted into the proper XML-formatted data, both the ID Resolver will process it so that all of its reference numbers can be loaded into the WCS system. All of these steps can be invoked from a simple batch file, thus the end user is unaware of the details once an administrator has created the appropriate templates and contribution flows and tested them against the WCM and

WCS system.

Figure 13 ID Conversion

Step4. Before the XML file is split into many XML files, some necessary steps will be executed as below:

<1>The elements and attributes in the output file of mapping rule are all uppercase, however, WCS only receives the XML files whose elements and attributes are all lowercase. Thus we have to manually create some tools to make these elements and attributes lowercase.

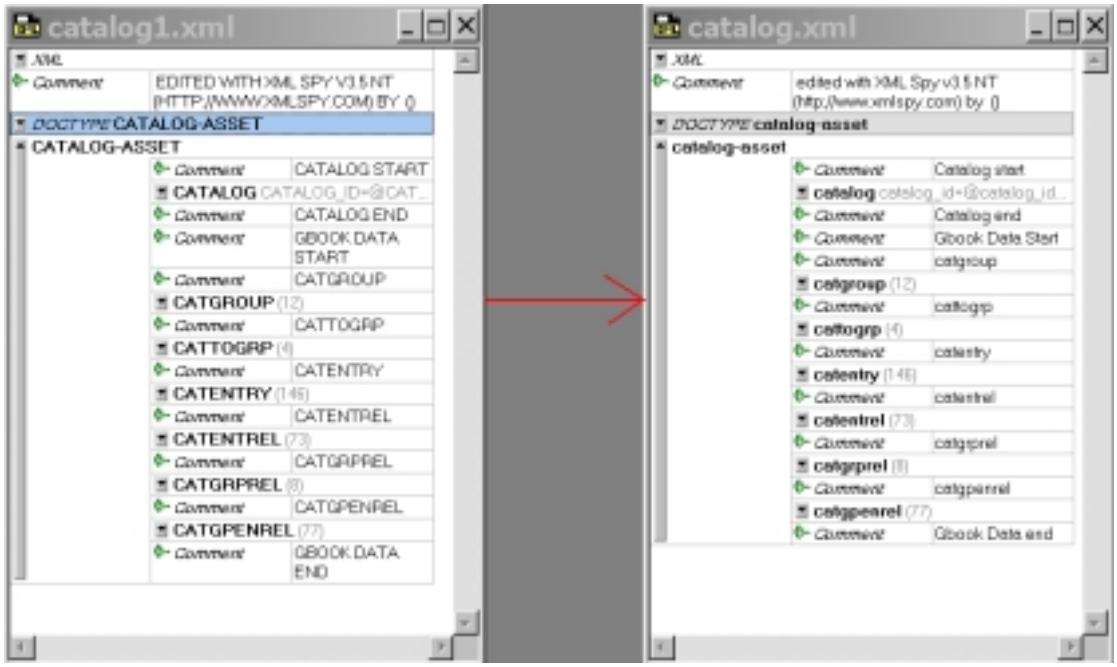

Figure 14 From lowercase to uppercase

<2>There are no listprice elements in the XML file. We have to create new listprice element based on the offer and offerlist element existed in the XML file. The new element, listprice, will include three attributes: the catentry_id from the offer element, the currency from the offerprice, and the listprice attribute from the price attribute of the offerprice element. When the value of the offer_id attribute in the offer element equals that in the offerprice element, we will form a new listprice element with attributes form in both offer and offerprice elements.

For example in offer elements there is
<offer catentry_id="@catentry_id_229" offer_id="@offer_id_201" />
in offerprice element there is
<offerprice currency="USD" offer_id="@offer_id_201" price="1.1"/>
So we could form a listprice element
<listprice catentry_id="@catentry_id_229" currency="USD" listprice="1.1"/>

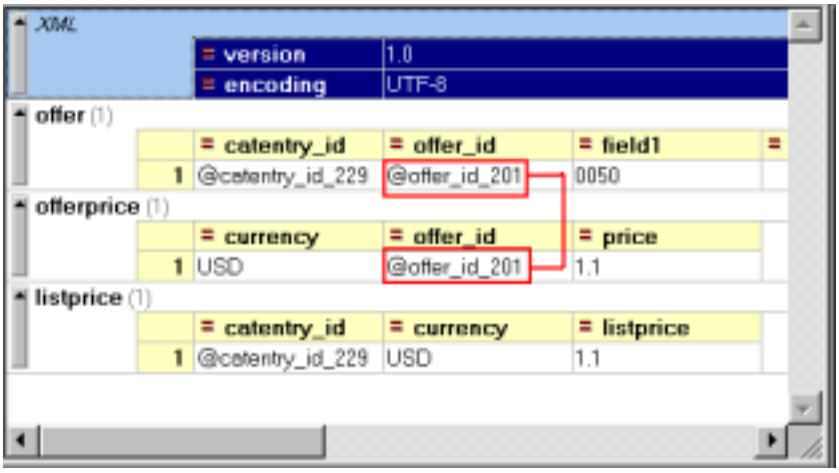

Figure 15 The method of creating a new element based on original elements

<3>The split rules of splitting the elements into these XML files are:

- split these catgroup,cattogrp,catentry,catentrel,catgrprel,catgpenrel elements into the catalog.xml file.
- split these storecent,storecgrp,dispcgprel elements into the store-catalog.xml file.
- split the inventory element into the storefulfill.xml file.
- split these offer,offerprice,listprice elements into the offering.xml
- split these catgrpdesc,catentdesc,attribute,attrvalue elements into the local catalog.xml file.

Note: the only difference of these locale catalog.xml files is the language_id number.

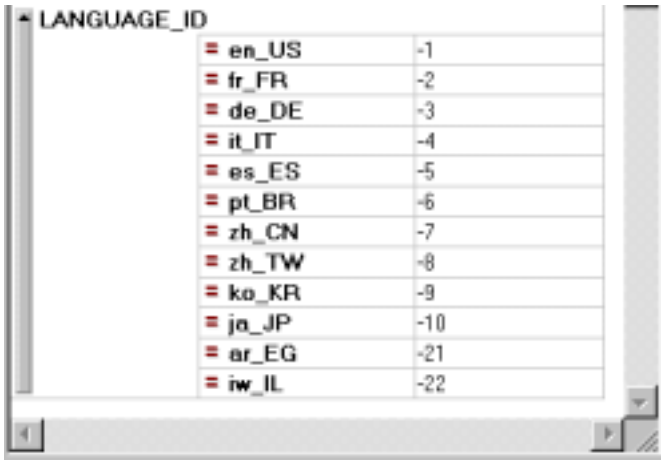

Figure 16 Language_id number

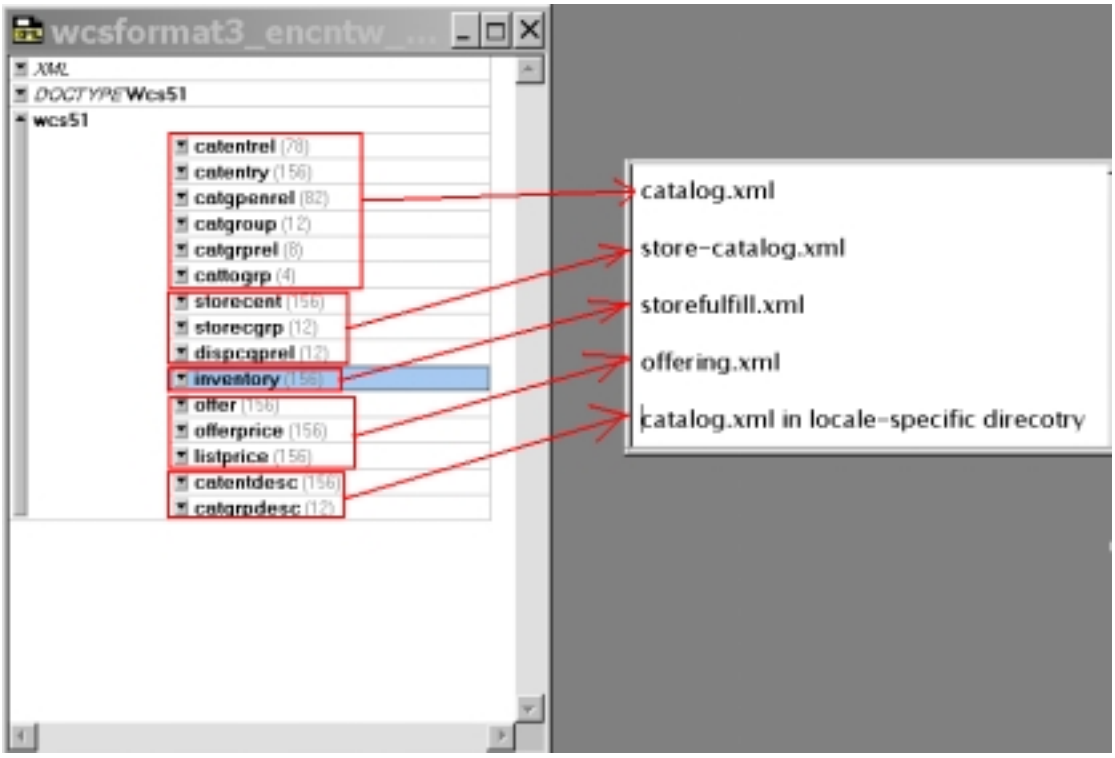

Figure 17 Split the XML file into a few files

Step5. Translate the XML files including store data's description into other language, and replace the language_id with the new language id.

Step6. Compress these files into one ZIP file and deploy it to WCS through WCS loader tool.

7.2The Solution to the Second Type of Business

To the second type of business, the multilingual products' description and the Web pages may be very similar to all users. Of course the culture conventions, including currency format, date etc, are also culturally sensitive. The description of catalog entry data is the same to every language and includes multilingual data in every language. There are only two differences compared with the methods of meeting the first type of business as below:

1. If we save these catalog entry data in an XML file, since the XML file can contain multilingual data, we may directly input or import multilingual data. However, the output file through the spreadsheet and WCM Client cannot contain multilingual data. So if we use the spreadsheet or WCM Client to create catalog entry data, we have to first input single language catalog entry description, get a XML file as the output after some transformation, and then translate some description into other languages or replace some with multi-languagual data

2. After we split the XML file into many XML files, we don't need translate the description of the store data into another language. Instead, we only replace the original language_id with another language_id which should be consistent with corresponding directory name.

Conclusion:

After reading this article, now users may have a better understanding on how to create multilingual data for WebSphere Commerce Suite via WebSphere Catalog Manager. Starting from here, you may begin to consider and enrich the solution to better meet your globalized requirement.

8. Reference

1. WebSphere Commerce Suite Website, http://www.ibm.com/software/webservers/commerce/servers/
2. WebSphere Catalog Manager Website, http://www.ibm.com/software/webservers/commerce/catalogmanager/
3. WebSphere Commerce Suite V5.1 Handbook, http://www.redbook.ibm.com/
4. WebSphere Catalog Manager Workbook and Solution Guide
5. Global Bookstore demo, http://9.185.27.123/
6. Globalization Architecture Imperative, https://d25dbr03.mkm.can.ibm.com/g_dir/gattdocr.nsf