



## Importing Data in the Commerce Manager

The Commerce Manager comes with a tool called an “Import Manager” for use in importing data. This guide will explain the features and capabilities of the import manager. Through the use of this very effective tool you can import entire catalogs of data in a quick and easy-to-manage manner. I have broken up the guide into two sections: The first includes the importing of data using a .csv file, then using the import manager to generate the mapping file. The second example shows how a mapping file may be structured as well as using existing a .csv and mapping file to import data.

Let’s now navigate to the import manager. Go to **Configuration > Import Manager** and here you will see the following screen:

### Import Manager

Use import manager to import data from other systems

[Home](#) > [Configuration](#) > Import Manager

---

#### Upload Data File:

Overwrite existing file

---

#### Edit Mapping File:

Select mapping file for editing rules.

---

#### Create mapping files and import data:

	Description	Action
<b>Mapping data</b>	Create Mapping file from data file	
<b>Importing data</b>	Importing data using mapping file and data file	

**The Import Manager**  
Figure 1

In the above picture you will notice an area to upload your data file. If you have a mapping file associated with your data file you can upload it here as well. We will now begin to step through this tutorial of how to import products into the eCF, and by doing such you will know how to import other elements such as customers, skus, etc. since they follow the same logic.

## Example 1 – Importing a single product and generating a mapping file

In this example we will do the simplest of operations – we will import a single product. This is actually all you need to know how to do to be effective with the import manager since importing multiple products is just a repetitive operation of what we will do here. We will create a .csv file loaded with a single product, and we will import that product into the e-Commerce Manager.

Here is the file we have created using Microsoft excel:

	2	3	4	5	6	7	8	9
1	product name	description	price	image	thumbnail			
2	Shirt	Plaid Shirt	9.99	../../../../documents and settings/tim/desktop/photos/pshirt.jpg	../../../../documents and settings/tim/desktop/photos/pshirt.jpg			
3								
4								

**.csv data file**  
**Figure 2**

The top line contains our column headings: product name, description, price, image url, thumbnail url. We can now use this file to upload our product:

The screenshot shows the 'Import Manager' interface. At the top, it says 'Use import manager to import data from other systems'. Below that is a breadcrumb trail: 'Home > Configuration > Import Manager'. The main section is titled 'Upload Data File:'. It has a checkbox for 'Overwrite existing file' which is checked. Below that is a text input field containing the file path 'ments and Settings\tim\My Documents\ATest.csv' and a 'Browse...' button. A red circle highlights the file name 'ATest.csv'. Below the input field is an 'Upload Data File' button. The next section is 'Edit Mapping File:', which says 'Select mapping file for editing rules.' and has a dropdown menu and a 'Load Mapping file' button. The final section is 'Create mapping files and import data:', which contains a table with two rows:

	Description	Action
Mapping data	Create Mapping file from data file	
Importing data	Importing data using mapping file and data file	

**Uploading the .csv file**  
**Figure 3**

In this case we will not be using a mapping file. After clicking the Upload Data File we are brought to the following screen:

## Import Manager

Use import manager to import data from other systems

[Home](#) > [Configuration](#) > Import Manager

**Create Mapping File:**

Step 1. Select data file and intended import type.

**Import Types**  
 Products

Data Files (*.csv)	Delimiter	Text Qualifier	Encoding
ATest.csv	,	(none)	Default

Step 2. Generate fields and attributes from import types to map to columns in data file.  
 (1 - required field for Insert, 2 - required field for Update)

Fields and Attributes	Column headers in the data file
Language <sup>1</sup>	<Not Set>
Meta Class <sup>1</sup>	<Not Set>
Code <sup>1,2</sup>	English
	Russian
Name <sup>1</sup>	
Category (Id)	
Product Template (Name/Id)	
Visible	
Page Title	
Meta Description	
Meta Keywords	
Brand	<Not Set>
Primary Image	
Catalog Image	
Description	
Details	

Step 3. Generate mapping file.

Enter to mapping file name to generate  
.xml

**Mapping the fields**  
**Figure 4**

Since we are not using mapping files, this screen made available to map all our fields from our import file to all the fields in the .ecf. Each field with a 1 is a required field and each field with a 2 is required for update.

Some fields may not have associated fields in the data file. One example of this, in our case, is the Language field. However, the drop down will give you a list of available languages to choose from. For the meta-class field you can choose from the available product templates.

From the picture above you can see that we our import is of type Products”, our data file is listed, and we are using comma delimited format with no text qualifiers. You can choose from available product meta-classes to map to. We want our plaid shirt to use the Everything\_Apparel meta-class:

Fields and Attributes	Column headers in the data file
Language <sup>1</sup>	<Not Set>
Meta Class <sup>1</sup>	<Not Set>
Code <sup>1,2</sup>	<Not Set>
Name <sup>1</sup>	Everything_Software Everything_Electronics Everything_Apparel
Category (Id)	Everything_Apparel
Product Template (Name/Id)	
Visible	
Page Title	
Meta Description	
Meta Keywords	
Brand	<Not Set>
Primary Image	
Catalog Image	
Description	
Details	

Assigning the meta-class  
Figure 5

We can continue to map each field as we need, matching up each field to the fields in our .csv file:

Fields and Attributes	Column headers in the data file
Language <sup>1</sup>	English
Meta Class <sup>1</sup>	Everything_Apparel
Code <sup>1,2</sup>	
Name <sup>1</sup>	<Custom value>
Category (Id)	Column 1 - product number
Product Template (Name/Id)	Column 2 - product name Column 3 - description Column 4 - price Column 5 - image Column 6 - thumbnail
Visible	
Page Title	
Meta Description	
Meta Keywords	
Brand	<Not Set>
Primary Image	
Catalog Image	
Description	
Details	

Assigning (Mapping) fields  
Figure 6

With all our fields mapped, now we are ready to generate our mapping file:

Step 1. Select data file and intended import type.

**Import Types**  
Products

**Data Files (\*.csv)** | **Delimiter** | **Text Qualifier** | **Encoding**  
ATest.csv | , | (none) | Default

Step 2. Generate fields and attributes from import types to map to columns in data file.  
(1 - required field for Insert, 2 - required field for Update)

Fields and Attributes	Column headers in the data file	
Language <sup>1</sup>	English	
Meta Class <sup>1</sup>	Everything_Apparel	
Code <sup>1,2</sup>	Column 1 - product number	product number
Name <sup>1</sup>	Column 2 - product name	product name
Category (Id)		
Product Template (Name/Id)		
Visible		
Page Title		
Meta Description	Column 3 - description	description
Meta Keywords		
Brand	<Not Set>	
Primary Image	Column 5 - image	image
Catalog Image	Column 6 - thumbnail	thumbnail
Description	Column 3 - description	description
Details		

Step 3. Generate mapping file.

Enter to mapping file name to generate  
ourTest.xml

Generate Mapping File

**Generate the mapping file**  
**Figure 7**

After clicking generate mapping file we are ready to import:

### Import Manager

Use import manager to import data from other systems

[Home](#) > [Configuration](#) > Import Manager

**Import data file to database:**

Choose the data file and mapping file to use:

<b>Data Files</b>	<b>Mapping Files</b>
ATest.csv	ourTest.xml

Import Data

**Import the data**  
**Figure 8**

Success!

**Import Manager**  
Use import manager to import data from other systems

[Home](#) > [Configuration](#) > Import Manager

**Import data file to database:**

Choose the data file and mapping file to use:

Data Files	Mapping Files
ATest.csv	ourTest.xml

Import was done. It has been successfully processed 1 rows from 1.

Type	Line	Message
------	------	---------

Page Size: 10

**Completed import**  
**Figure 9**

And here is our added product:

**Products**  
List of all products in the system

[Home](#) > [Catalog](#) > Lists > All Products

Search:

<input type="checkbox"/>	Name	Template
<input type="checkbox"/>	256MB Memory Card	Electronics Template
<input type="checkbox"/>	HDMI Audio Video Cables	Electronics Template
<input type="checkbox"/>	Home Theater System	Electronics Template
<input type="checkbox"/>	Camera Tripod	Electronics Template
<input type="checkbox"/>	Cell Phone Cover	Electronics Template
<input type="checkbox"/>	Hands-free Set	Electronics Template
<input type="checkbox"/>	Carrying Clip	Electronics Template
<input type="checkbox"/>	Shirt	Apparel Template

Page Size: 10

**The newly added product**  
**Figure 10**

## Example 2 – Creating and Using Mapping Files

Mapping Files are .XML files which are used to map source columns in a data file to destination columns in a database. We will use the following example to aid you in your creation of your own mapping file:

Given the above example we will accomplish the same thing, except instead of generating a mapping file, we will create one and use it to map the appropriate fields. Here is the mapping file we have constructed:

```
mymappingfile.xml
<?xml version="1.0" encoding="utf-8"?>
<Rule>
  <Id>-1</Id>
  <InnerClassName>TestProducts</InnerClassName>
  <TypeName auto="yes">Products</TypeName>
  <Encoding auto="yes">Default</Encoding>
  <MetaClassId auto="yes">12</MetaClassId>
  <DataFile auto="yes">C:\Documents and Settings\tim\My Documents\ATest2.csv</DataFile>
  <LanguageId auto="yes">1</LanguageId>
  <Delimiter auto="yes" />
  <TextQualifier auto="yes" />
  <RuleItem>
    <SourceColumnName>product name</SourceColumnName>
    <SourceColumnType>System.String</SourceColumnType>
    <DestinationColumnName>Name</DestinationColumnName>
    <DestinationColumnType>ShortString</DestinationColumnType>
    <FillType>CopyValue</FillType>
    <CustomValue />
    <DestinationColumnSystem>False</DestinationColumnSystem>
  </RuleItem>
  <RuleItem>
    <SourceColumnName>product number</SourceColumnName>
    <SourceColumnType>System.String</SourceColumnType>
    <DestinationColumnName>Code</DestinationColumnName>
    <DestinationColumnType>NVarChar</DestinationColumnType>
    <FillType>CopyValue</FillType>
    <CustomValue />
    <DestinationColumnSystem>False</DestinationColumnSystem>
  </RuleItem>
  <RuleItem>
    <SourceColumnName>description</SourceColumnName>
    <SourceColumnType>System.String</SourceColumnType>
    <DestinationColumnName>Everything_Description</DestinationColumnName>
    <DestinationColumnType>LongHtmlString</DestinationColumnType>
    <FillType>CopyValue</FillType>
    <CustomValue />
    <DestinationColumnSystem>False</DestinationColumnSystem>
  </RuleItem>
  <RuleItem>
    <SourceColumnName>image</SourceColumnName>
    <SourceColumnType>System.String</SourceColumnType>
    <DestinationColumnName>Everything_ProductPrimaryImage</DestinationColumnName>
    <DestinationColumnType>ImageFile</DestinationColumnType>
    <FillType>CopyValue</FillType>
    <CustomValue />
    <DestinationColumnSystem>False</DestinationColumnSystem>
  </RuleItem>
</Rule>
```

```

<RuleItem>
  <SourceColumnName>thumbnail</SourceColumnName>
  <SourceColumnType>System.String</SourceColumnType>
  <DestinationColumnName>Everything_CatalogImage</DestinationColumnName>
  <DestinationColumnType>ImageFile</DestinationColumnType>
  <FillType>CopyValue</FillType>
  <CustomValue />
  <DestinationColumnSystem>False</DestinationColumnSystem>
</RuleItem>
</Rule>

```

**.xml mapping file**  
**Figure 11**

By simple examination of our mapping file above you can see easily how the data is mapped. The hierarchy of the file is constructed like so:

- We have the standard XML declaration at the top.
- This is followed by tags which define the type of element being mapped (i.e. product)
- the encoding
- The metaclass we are using (in this case we refer to everything\_apparel by its id)
- The path to the data file.
- The language (English referred to by its id)
- The Delimiter tag
- The Text Qualifier tag

Next we have the main body of our data which is arranged by “RuleItems” each rule item contains:

- The source column name which contains the column name of the source column
- The source column type (i.e. string, int, varchar, etc)
- The destination column name
- The destination column type
- The Filltype (how the destination column will be filled, in our case we are copying the source value)

Once your mapping file is complete you are ready to import. To do this, please follow these steps:

1. Go to the import manager and browse for your .csv file. When you have located the file upload it by pressing “Upload Data File”.
2. The next thing you need to do is find out where your .csv file was uploaded, which will be in your temporary files under a path similar to:

C:\Documents and Settings\{yourcomputer}\aspnet\Local Settings\Temp\eCFImport\data\1373056505

When you have located your .csv file you will need to place your .xml mapping file in a similar location along the same path but in a different folder:

C:\Documents and Settings\{yourcomputer}\aspnet\Local Settings\Temp\eCFImport\rule\1373056505

3. The next step is to go back to import manager: **Configuration > Import Manager** and choose the dropdown next to the button “load mapping file”.

**Import Manager**  
Use import manager to import data from other systems

[Home](#) > [Configuration](#) > Import Manager

**Upload Data File:**

Overwrite existing file

**Edit Mapping File:**

Select mapping file for editing rules.

**Create mapping files and import data:**

	Description	Action
<b>Mapping data</b>	Create Mapping file from data file	
<b>Importing data</b>	Importing data using mapping file and data file	

**Loading the mapping file**  
**Figure 12**

If you have put your mapping file in the right location you will see it in the dropdown. After you have selected your file click “load mapping file”. You will be taken to the next familiar page, but this time all your columns are mapped just as stated in your mapping file:

### Import Manager

Use import manager to import data from other systems

[Home](#) > [Configuration](#) > Import Manager

**Create Mapping File:**

Step 1. Select data file and intended import type.

**Import Types**  
Products

Data Files (*.csv)	Delimiter	Text Qualifier	Encoding
ATest2.csv	,	(none)	Default

Step 2. Generate fields and attributes from import types to map to columns in data file.  
(1 - required field for Insert, 2 - required field for Update)

Fields and Attributes	Column headers in the data file	
Language <sup>1</sup>	English	
Meta Class <sup>1</sup>	Everything_Apparel	
Code <sup>1,2</sup>	Column 1 - product number	product number
Name <sup>1</sup>	Column 2 - product name	product name
Category (Id)		
Product Template (Name/Id)		
Visible		
Page Title		
Meta Description		
Meta Keywords		
Brand	<Not Set>	
Primary Image	Column 5 - image	image
Catalog Image	Column 6 - thumbnail	thumbnail
Description	Column 3 - description	description
Details		

Step 3. Generate mapping file.

Enter to mapping file name to generate  
timestmapping.xml

**Fields mapped by the mapping file**  
**Figure 13**

## Conclusion

As you can see you are able to move large quantities of data with little effort using the import manager. You can take the principles learned here and apply them to larger files with more products. If you need to migrate data into the .eCF you need only your datafiles and corresponding mapping files and you will be able to get your store populated in no time!

If you wish to learn about other great features in the Commerce Manager, please check out our Commerce Manager guide at:

[http://www.mediachase.com/documentation/ecfv4doc/ecf4\\_commerce\\_manager.pdf](http://www.mediachase.com/documentation/ecfv4doc/ecf4_commerce_manager.pdf)