**Instructions For CED Label Print**

Table of Contents

[What is the CED Label Print tool for? 2](#_Toc7015805)

[How Does it Work? 2](#_Toc7015806)

[Can the Output Include Pictures of the Products? 3](#_Toc7015807)

[Two Very Common Uses 4](#_Toc7015808)

[Using CED Label Print For VMI Barcode Labels 4](#_Toc7015809)

[The first step in using this tool is to go to http://vmi.myced.com and use the website to download the products. 4](#_Toc7015810)

[The second step is running the CED Label printer tool. 5](#_Toc7015811)

[Using CED Label Printer To Product Labels from a Price List 6](#_Toc7015812)

[Barcode Labels by Avery 7](#_Toc7015813)

[Configuration Settings 8](#_Toc7015814)

[Each Configuration in Detail 9](#_Toc7015815)

[Part 2 – Advanced description on the content of the “Input Type” and “Output Type” templates 11](#_Toc7015816)

[Node 11](#_Toc7015817)

[Attribute 11](#_Toc7015818)

[Value 11](#_Toc7015819)

[Input Type 12](#_Toc7015820)

[Reference Example VMI 12](#_Toc7015821)

[ColumnMap Node 12](#_Toc7015822)

[headerRow 12](#_Toc7015823)

[validateHeader 13](#_Toc7015824)

[fileType 13](#_Toc7015825)

[fixedWidth 13](#_Toc7015826)

[ColumnMapItem Node 13](#_Toc7015827)

[columnNumber 13](#_Toc7015828)

[headerValue 13](#_Toc7015829)

[overrideMode 13](#_Toc7015830)

[Value 14](#_Toc7015831)

[Data Entities List 14](#_Toc7015832)

[Text Example VMI Input 15](#_Toc7015833)

[Text Example Generic CEDNet Price List 15](#_Toc7015834)

[Output Type 16](#_Toc7015835)

[Reference Example VMI Output 17](#_Toc7015836)

[Paper 18](#_Toc7015837)

[LabelTemplate 18](#_Toc7015838)

[Text Item 18](#_Toc7015839)

[Image Item 21](#_Toc7015840)

[QR Item 22](#_Toc7015841)

[Barcode 128 Item and Barcode 3 of 9 Item 23](#_Toc7015842)

[Sheet Template 24](#_Toc7015843)

[Text Example Avery 5162 Output (for copying and editing) 24](#_Toc7015844)

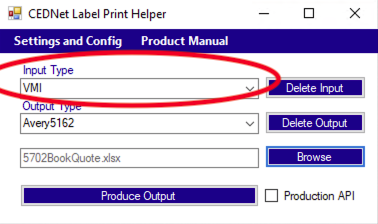
# What is the CED Label Print tool for?

CED Label Print is a tool for creating barcode labels, creating product-ordering booklets, and also works for a variety of other printing needs. What makes this tool unique is that it can take product data in a variety of common formats and layouts, and is not restricted. With this data, it can then output to a standard laser printer, a custom label printer, label sheets, and even pdf files.

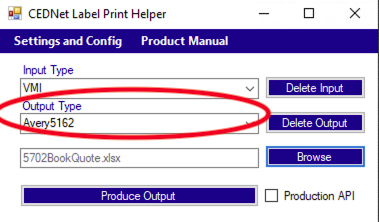
# How Does it Work?

CED Label Print works by selecting the input type from the dropdown, selecting the output type from the other dropdown, browsing to your data file and clicking the “Produce Output” button.

The “Input Type” describes what kind of input you’re using. For example, if the products are in an excel file where the second column is the manufacturer code, the third column is the catalog number, and the fourth column is the description, this “Input Type” selection tells the CED Label Print Tool where to find that information. This way, if you get a different layout where the catalog number is in a different column, you can still use that input file.



The “Output Type” is similar because it tells the CED Label Print Tool what format the resulting output should be in. For example if you want your output to print to a laser printer with the description of the product in large letters across the top, the catalog number right underneath it in bold font, and a barcode of the product underneath with a picture on the right, all that information is included with the output type.



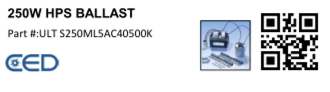
It is possible to load custom input types, but some of the standard ones should be preloaded for you. It is also possible to load custom output types. Some of the most common output types such as Avery 5162 sheets with QR code should be preloaded also. In time, you may become a whiz at making custom output types on your own, but it is recommended to request assistance from the helpdesk, and someone will get in contact with you and send you an output load file that meets your specifications.

The input types and output types are created in a format called xml. Please see the “Example Files” section of this document if you are interested in learning more about creating your own formats.

# Can the Output Include Pictures of the Products?

Yes. One important thing to note is that the CED Label Print Tool actually can get more data than you include in the input file. This works because the tool actually goes out to the Internet, and finds additional data on the product. With this in mind, it is highly recommended to connect to the Internet while running the tool. This way, it can retrieve pictures, customer part numbers, customer prices, and a variety of other information and add that to the information included in your data file.

Example Output:



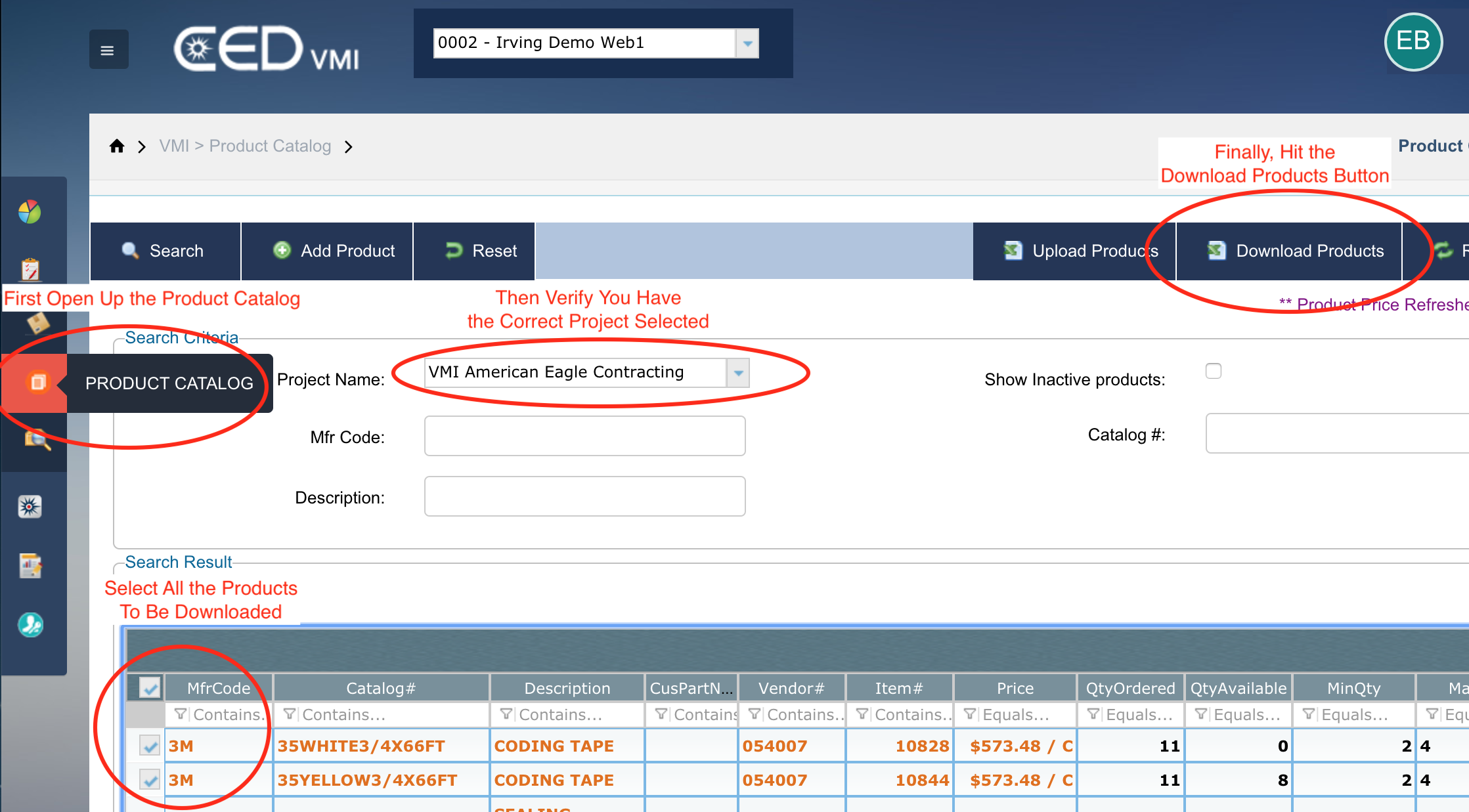
# Two Very Common Uses

## Using CED Label Print For VMI Barcode Labels

When using the tool for VMI, the idea is that you should extract the parts from <https://vmi.myced.com>

### The first step in using this tool is to go to http://vmi.myced.com and use the website to download the products.

* Login to the site
* On the left side of the page there will be some icons. Find the one that says “Product Catalog” and click it. You may need to expand the “VMI” Icon to see it.
* Verify that you have the correct account selected next to “Project Name”
* Note that your products will need to be added to the catalog before they can be exported. So if you don’t see any products, that’s probably what’s happening.
* Select the products you will want to export using the checkboxes. The checkbox at the top will select all products for export.
* Press the “Download Products” button. Depending on your browser, the file will either automatically appear in your downloads, or it will give you the option of saving it to your desktop. Wherever it is, just make sure you know how to browse to that file, because, this is your “data source” file for the tool.

****

### The second step is running the CED Label printer tool.

* Simply select “VMI” from the Input Type
* For the Output Type, select the kind of label you’ll be using
* Click Browse, next to the “Choose Data File” textbox. Note that you’ll be using the file that you just downloaded.
* Click, Produce Output

If you haven’t configured the tool yet for information like your PC Number and default account, the tool may force you to fill that information in before it proceeds. This information is used to retrieve additional information about the product. You can use things like customer part number on your labels and even price if you so choose. If you are not interested in using that information, you can put in a Cash or COD account number, and it will work fine.

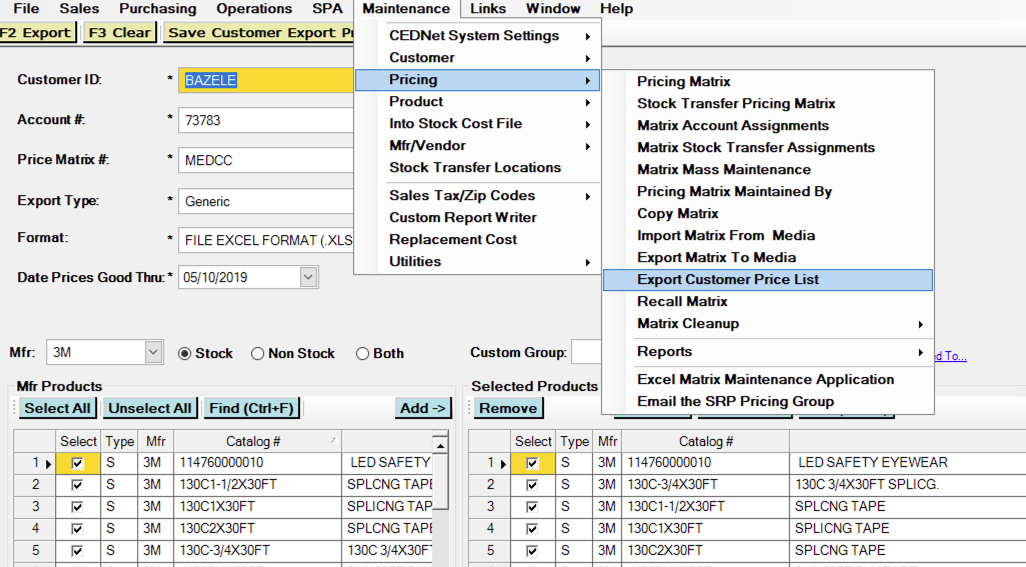
If your favorite label type is not already loaded into the tool, contact the helpdesk, and we can produce a load file and that can be loaded into the tool, and then it will be available anytime.

## Using CED Label Printer To Product Labels from a Price List

In this case you will be using CEDNet to produce the data file. Go to Maintenance🡪Pricing🡪Export Customer Price List

CEDNet will ask you to put in a customer and account number. Keep in mind that just because this tool will produce a data file that has prices included does not mean that this information is going to be included on the final barcode labels. So, it is possible to pick a customer account arbitrarily so that you can add the appropriate products to the list, and disregard the pricing information for the purposes of label creation.

For the export type, there are a few options. You can just pick “Generic”, and the format “FILE EXCEL FORMAT (.XLS)”. That will work. The tool also supports any of the formats, but for each of the formats, you would have to have the appropriate input-mapping file, and we have included the Generic XLS already in the tool. If you have a special need just contact the helpdesk and we can provide you the input-mapping file for a different format.



# Barcode Labels by Avery

Avery labels have product numbers that describe the labels. For example, the 5162 labels have two across, and the 5160 labels have 3 across. Just to be clear, it is worthwhile to have a deeper understanding of the product number.

The first digit “5” specifies that these labels are for laser printers. Avery also makes labels that start in “8” which means inkjet. If you have an inkjet, these template forms will also work, but you may be looking for 8162, or 8160 labels in the store.

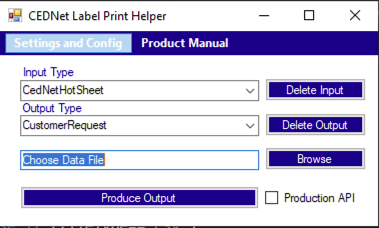
The second digit “1” specifies how many sheets come per package. So, the 5162, and the 5362 are the exact same labels, but they come with a different number of labels per package.

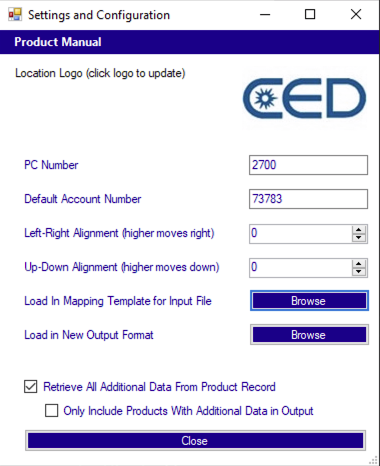
The last 2 numbers are the ones that are important. As mentioned earlier the “62” means they are 2 across, and the “60” means they are 3 across.

Generally, it will always be easier and better quality to use the 2 across, or 5162 labels compared to the 3 across. The 3 across come with more labels per sheet and are cheaper, but the barcodes are smaller and harder to read.

# Configuration Settings

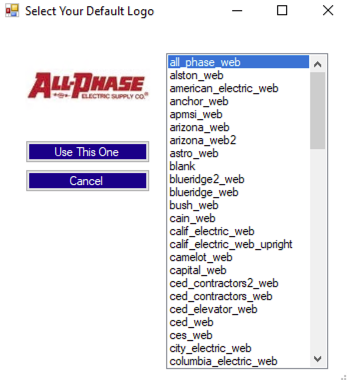
To get to the configuration settings, click the “Settings and Config” menu button, and it will open up the Configuration Settings screen. If you haven’t gone through and configured your settings on the first time you run the tool, this will happen automatically when you try to print.

****

****

## Each Configuration in Detail

* **Logo** – It is a good idea to set the logo to what makes sense for your location. The default is a CED logo. If you would like to change it, click on the logo and a new window will appear.



* **PC Number** – Enter your 4 digit PC number
* **Default Account Number** **-** Enter a 5-digit account number. This can just be a COD account number, but if you are printing a label for a customer and it will include customer part number or even pricing for that customer, this tool supports those features. So updating this value for each customer that you print labels for is a good practice.
* **Left-Right Alignment** – Not all printers are exactly the same. If you print and find out that for your labels, everything is a little too far to the left or right, you can update the general offset. This is particularly useful when the typing is small and the labels are on a sheet. Note that higher number will push the text to the right. Each 100 represents an inch, so an adjustment of a tenth on an inch to the right would mean putting a 10 in this box. 10 are about the maximum need for a shift in practice.
* **Up-down Alignment** – This is the same as Left-Right Alignment, but on the up-down axis. Note that higher numbers shift the content down. Also note that negative numbers are permitted in either alignment if needed.
* **Load In Mapping Template for Input File** **–** This is where you can load in custom file types to describe the input content. Once you click Browse, and find the xml file that was created, and click OK, then it is done. Later in this manual, there will be further instructions about the content of the file. After closing the advanced settings, a new item will be available in the “Input Type” dropdown.
* **Load in New Output Format** –This is the same as the Input file, but it loads in your label template for the output. Again, it is looking for an xml file. Also, the details of the format of this file will be described later.
* **Retrieve All Additional Data From Product Record –** This tool works in 2 steps. First it loads in your data file – most likely an excel sheet with product data. Then, with the manufacturer codes and catalogs it finds in that sheet, it retrieves additional data. This is how the tool can find things like a product image, or long descriptions. This checkbox will determine if it takes that second step or not. If all the data you’re ever going to need is in the uploaded excel data source, then you do not have to have this checked. The default for this is checked so that it will look for additional product data.
* **Only Include Products With Additional Data In Output** –This checkbox will only be available to you if you opted to turn on “Retrieve All Additional Data From Product Record”. If the tool is going out to find more data, then it goes through a merge process. It is possible that, there are products loaded in the excel sheet that could not be found. In that case, those products that will not be found will be missing information like product image. If this box is checked, then it will not include those products in the final output, and only the ones that had additional product data. Also, if this check is enabled, a file will be placed on your desktop called “ExcludedParts.txt” This file will let you know what products were omitted from the output. If this box is unchecked, then all the parts will appear in the output, some of them may be missing data. The default for this is unchecked so that all the entries on the original data sheet will be included even if they are missing additional product data from the additional online data source.

# Part 2 – Advanced description on the content of the “Input Type” and “Output Type” templates

The input and output types are in a format called xml. If you’re not familiar with working with xml in a text editor, this may be a challenge to you. That being said, these are some of the simplest and shortest xml files that you can possibly work with, so it is a reasonable place to start. But, if you aren’t used to seeing xml, it is recommended that you view this tutorial online.

<https://www.w3schools.com/xml/>

<?xml version="1.0" encoding="UTF-8"?>  
<note type=”message”>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>

In xml, there are “nodes”, “attributes” and “values”.

## Node

The above example has a “note” node. You can see it has an opening tag which looks like <note> and a closing tag which looks like </note>. So everything from the opening tag to the closing tag is considered a “node”. In this case, you can see that there is also a “to” node. The “to” node is inside the “note” node. So, a node can contain other smaller nodes.

## Attribute

Looking at the “Note” node, you will see that inside the opening tag, there is another part. The ‘type=”message”’ is what is referred to as an attribute. An attribute gives additional information about a particular node. In this case it specifies that the type of note is a message.

## Value

Looking at the “to” node, you can see that there is not another node inside it, but just the text “Toye”. This can be referred to as the “Inner Text” of the “To” node, but it can also be referred to as the value of the “To” node since there is just one thing there.

Value is also used to describe the relationship between the attribute “type” and the value for that attribute “message”.

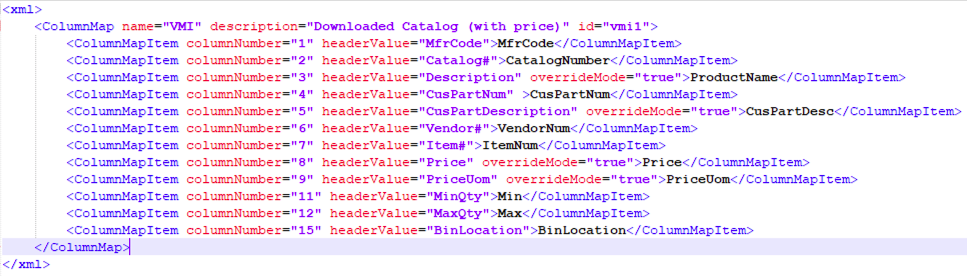
So “value” can be used to describe the “Inner Text” of a node, when there is only one thing there, or it can be used to describe the “value” of a particular attribute.

Here are a couple sentences about the above xml:

* The value of the body node is “Don’t forget me this weekend!
* There are 4 inner nodes included in the note node; They are: to, from, heading, and body
* The value of the type attribute in the note node is message
* The root node of the above xml is note

# Input Type

## Reference Example VMI

****

Looking at what we have above. There is a root node of xml, which contains a ColumnMap node. Inside the ColumnMap node, there are a series of ColumnMapItem nodes.

## ColumnMap Node

The ColumnMap node has 3 required attributes: name description, and id. Those 3 attributes must always be there, but the values can be whatever you like. They are just used internally by the system, and will not appear anywhere on your output.

The ColumnMap node has 4 optional parameters that will be discussed below in greater detail:

headerRow – header row tells the program which row the header is in the data. By default, the header row is 1 meaning that the first row is expected to tell what’s in each column. And then the second row will actually include the product data itself.

validateHeader – the validateHeader attribute, by default is “true”. This means that when the tool is validating your data source with this input file, that the headers that appear in the column appear exactly as stated in the ColumnMapItem nodes. It is highly recommended for this to remain true. The most common reason to set this to false is because no header exists. In that case the headerRow=”0” and validateHeader=”false” This means that the row includes product data

fileType– the fileType attribute is only used when the file extension of the data file is ambiguous. For example, if your data file is .cif, or .001, or .lsq, or even .txt, the system cannot be sure what is the format of the file. The possible options are (and they must be spelled exactly like this: “csv”,”tabDelimited”, “fixedWidth”, “xL”, or “legacyXL”. Although “xL” or “legacyXL” will probably never be used since excel files can typically be determined by their file extension.

fixedWidth **–** The fixedWidth attribute is used only when the fileType is specified as “fixedWidth”. In this case additional information is needed for the software to read the file. A fixed width file means that it is a text file where the first so many characters designate the first column, and then the next number of characters represent the second column. The content of a fixed width, which has, for example, 4 columns might look like this fixedWidth=”5,20,35”. What this means is that the first 5 characters is the first column, the next 20 characters is the second column, the next 35 characters is the third column, and everything from there to the end is the last column. The value of the fixedWidth attribute must always be integers separated by commas, and however many values you specify, there will always be one more column since the width of the last column is always unspecified.

## ColumnMapItem Node

The columnMapItem node will exist for each column that exists in your data file. There are 2 required attributes, for the columnNumber and headerValue. The value is required also. There is also 1 optional attribute

columnNumber– the columnNumber attribute is required, and the value must be an integer. It specifies which column that data will be found in.

headerValue = the headerValue attribute is required. It is used during validation when the data file is being mapped to the input file. Validation is always a good idea because it’s easy to load in the wrong data file. Also, even if the data source does not have headers, just keep this attribute and set the value to “none” as in headerValue=”none”

overrideMode- the overrideMode attribute is an optional attribute. The default value is false. You can see in the above example that it is used for the customer part description column. What this does is resolve a conflict when the input file has the same data as the additional data source from the Internet. The default action is to go with the Internet source when there is a conflict because it’s generally not going to contain errors or typos. However, there are cases when you want the excel file to win in the case of a conflict. For example, what if the customer doesn’t like the description that our system has in their customer part file? You can modify the excel sheet to be whatever they want, and then it will be the chosen customer part number description.

Value **– the Value of the ColumnMapItem is critical.**  How the tool works is that it takes the input and it maps it to a data element using the value. The accepted values are restricted to exactly what is in the list below. These are placeholders in the system, and when the system goes out to the Internet, it will place data for the products in these same data elements. Most importantly, it is these data elements that will be referenced in the output file to specify which data goes where on your label. In other words, it is these data elements that bind the input with the output.

As a side note, the Flex1, Flex2, Flex3, and Flex4 fields are used in case there is additional data in the input source that really doesn’t belong to a specific product’s data, but you might want to use in your labels anyways. What if you wanted to include a category on your labels like “Lighting”, then you can specify that as a column on your input source, and map it to a Flex field, and use it on your labels?

## Data Entities List

Please note to keep these values case sensitive. In other words, type them exactly as you see them!

**FullTechDescripion**

**MfrLongDescription**

**MfrShortDescription**

**ProductName**

**StockCode**

**ImageUrl**

**ThumbnailUrl**

**Unspsc**

**Upc**

**MfrCode**

**CatalogNumber**

**SpecificationSheet**

**Price**

**PriceUom**

**PriceSource**

**CEDProductID\_Key**

**UniqueProductID\_Key**

**Min**

**Max**

**BinLocation**

**CusPartNum**

**CusPartDesc**

**VendorNum**

**ItemNum**

**CartonQuantity**

**SalesMinOrderQuantity**

**SoldInMultiples  
AverageLeadTime**

**QuantityOnHand**

**Flex1**

**Flex2**

**Flex3**

**Flex4**

## Text Example VMI Input

Use this example for copying and pasting out of this instructions document to modify for your own needs. It’s always easier to modify someone else’s work that to build your own from scratch!

The below example is what the mapping file looks for the input of the vmi file

<xml>

<ColumnMap name="VMI" description="Downloaded Catalog (with price)" id="vmi1">

<ColumnMapItem columnNumber="1" headerValue="MfrCode">MfrCode</ColumnMapItem>

<ColumnMapItem columnNumber="2" headerValue="Catalog#">CatalogNumber</ColumnMapItem>

<ColumnMapItem columnNumber="3" headerValue="Description" overrideMode="true">ProductName</ColumnMapItem>

<ColumnMapItem columnNumber="4" headerValue="CusPartNum" >CusPartNum</ColumnMapItem>

<ColumnMapItem columnNumber="5" headerValue="CusPartDescription" overrideMode="true">CusPartDesc</ColumnMapItem>

<ColumnMapItem columnNumber="6" headerValue="Vendor#">VendorNum</ColumnMapItem>

<ColumnMapItem columnNumber="7" headerValue="Item#">ItemNum</ColumnMapItem>

<ColumnMapItem columnNumber="8" headerValue="Price" overrideMode="true">Price</ColumnMapItem>

<ColumnMapItem columnNumber="9" headerValue="PriceUom" overrideMode="true">PriceUom</ColumnMapItem>

<ColumnMapItem columnNumber="11" headerValue="MinQty">Min</ColumnMapItem>

<ColumnMapItem columnNumber="12" headerValue="MaxQty">Max</ColumnMapItem>

<ColumnMapItem columnNumber="15" headerValue="BinLocation">BinLocation</ColumnMapItem>

</ColumnMap>

</xml>

## Text Example Generic CEDNet Price List

The below example is what the mapping file looks like for the input of the Generic Excel price list from CEDNet

<xml>

<ColumnMap name="CEDNet xls field" description="Price List" id="PriceListXls">

<ColumnMapItem columnNumber="1" headerValue="UPC">Upc</ColumnMapItem>

<ColumnMapItem columnNumber="2" headerValue="Mfr">MfrCode</ColumnMapItem>

<ColumnMapItem columnNumber="3" headerValue="Catalog #" >CatalogNumber</ColumnMapItem>

<ColumnMapItem columnNumber="4" headerValue="Description">ProductName</ColumnMapItem>

<ColumnMapItem columnNumber="5" headerValue="Price">Price</ColumnMapItem>

<ColumnMapItem columnNumber="6" headerValue="Per">PriceUom</ColumnMapItem>

</ColumnMap>

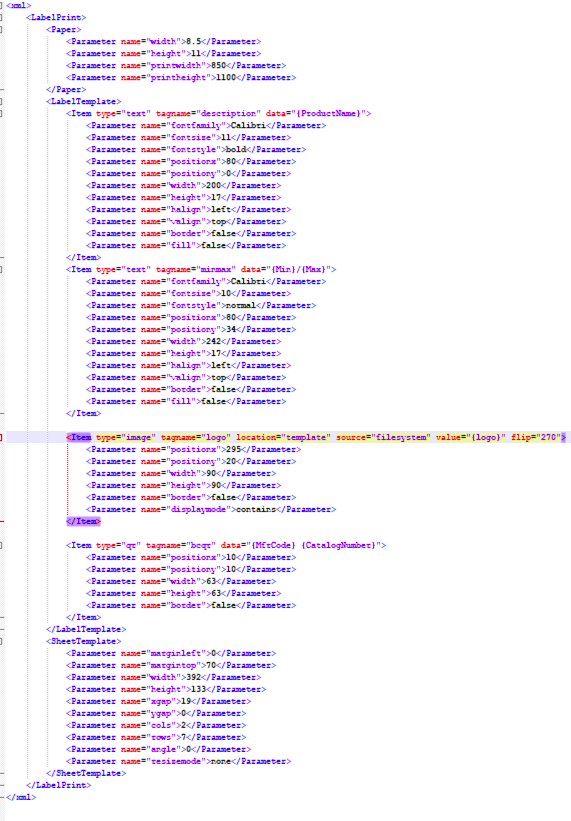
</xml>

# Output Type

The output type file is an xml file like the input type, but it has quite a few more details and it somewhat more complex. But looking at the type of information that it accepts also gives an idea over how much control is at your hands.

Let’s look at an example output file (next page).

## Reference Example VMI Output



In this file, the root node is also <xml>, and the whole thing is wrapped in the <LabelPrint> node also. To break down the contents, I will cover each of the 3 main node segments separately. The 3 nodes are <Paper>, <LabelTemplate>, and <SheetTemplate>.

## Paper

The paper segment is very easy. If you plan on printing on standard 8.5 x 11 paper, this will never change. This feature is there to support spool printers and customer printers that may be of a non-standard size. There are 4 Parameter nodes in the Paper node, and all are required

**Parameter node with attribute name=”width”** – This node describes the width of the paper in inches. 8.5 inches is the standard width.

**Parameter node with attribute name =”height”** – This node describes the height of the paper in inches 11 inches is the standard height.

**Parameter node with attribute name=”printwidth”** – This node describes the width of the label in terms of pixel width. The standard for what we are doing is 100 pixels per inch, so the standard printwidth is 850. However, when dealing with different paper sizes, sometimes the pixel width may be slightly less than the paper. For example, on a spool printer with a width of 2 inches, sometimes there are a few pixels that are designated for margin, and the appropriate printwidth for those is 192. Otherwise, the printer may print a second blank page after every label.

**Parameter node with the attribute name = “printheight”** This node is the vertical equivalent to the print width. The same margin pixel anomalies exist in smaller labels.

## LabelTemplate

This is where all the information about the layout of the label is contained. The <LabelTemplate> node contains any number of <Item> nodes. Each item designates an element that belongs on the label. There are 5 kinds of item nodes, and will cover each independently.

### Text Item

The most commonly used item is a text item. That is to say that in the item, the type attribute is set to text (i.e. type=”text”). For each item type, there are different required parameters. Also for each item type there are 3 required attributes, the type, tagname, and data.

**Type Attribute** – Will always be “text” in a text item

**Tagname Attribute**– it is required, but it can be whatever you think is adequate. However, the tagname should not be the same for more than 1 item. The system uses the tagname as a handle to reference a particular element, so it’s possible that problems could occur if a tagname were reused.

**Data Attribute** – This is what specifies the data content of this particular text. In other words where does the text come from? In the data if you put the words “Hello”, then on every label, the word “Hello” would appear. However, if you use one of the data entities as specified above in the “Value” section of the input mapping, then that is the data that will appear. When referencing a Data Element, always put that element is curly braces.

**Further explanation on the Data Attribute**

There are lots of capabilities here. Look above on the example for the first Item with the tagname “description”. You will notice that the data attribute is set for {ProductName} (i.e.. data=”{ProductName}”. Since the ProductName is specified as a data element (being in curly braces), then the system puts the ProductName value in for thee text.

Where does the ProductName come from? In the input file, we mapped a certain column to the ProductName element, so that is one primary source is the input file. Also, the system went out to the Internet, and tried to find the Product name for that product also.

Look at the next Item where the type is text and the tagname is “minmax”. This one has a data value of data=”{min}/{max}”. That this means is that the output will be whatever the content for the min is will be followed by a forward slash, and then whatever the value for the max is will be displayed after that. This means that it is possible to include more than one data element into the data attribute, and also that it is possible to mix standard text with data elements.

Lastly, there is not an example of this above, but you can also place conditional arguments inside the data attribute. For example, let’s say you wanted to display a price on a label, but not all products are retrieving a price. This could be because not all products are being found on the Internet, and you didn’t include a price in the input file, so essentially the {Price} element was left blank. In this case, you can specify that if no price exists, then display “call for price”. How would you do this? By setting a conditional data argument. It would look like this data=”${Price}/{PriceUom}||call for price” How to read this is that the output will put a dollar sign followed by the price, a forward slash, and then the unit of measure. If either the Price data element is empty or the PriceUom data element is empty, it will go to the value after the double pipes || and display “call for price”.

**Required Parameters Within a Text Item**

**Parameter name = fontfamily** – the font used for the text is the value of the Parameter node

**Parameter name = fontsize** - the size of the font is the value of the Parameter Node

**Parameter name = fontstyle** – the style is the value of the Parameter Node. Accepted values are “regular”, “bold”, “italic”, “underline”, “strikeout”, “bolditalic”, and “boldunderline”

**Parameter name = positionx**  - position from the left edge in 100 pixels per inch

**Parameter name = positiony** – position from the top of the label in 100 pixels per inch. Meaning a value of “25” specifies a quarter of an inch. For reference 20-30 typically constitutes about the height of a line of text.

**Parameter name = width** – to understand this it must be explained that we’re basically defining a box where the text will go. This is important because once the text reaches the end of the box, it will be cut off, or depending on your settings, it may go to the next line (if a multiline parameter is added). This “box” can also be used for positioning text in the center, or for shading the box, or placing a border around it.

**Parameter name = height** – same features as width, but applied for the height. Note that all values are based on 100 pixels per inch.

**Parameter name = halign** – this specifies where the text will appear in the “box” that we have specified for it. Accepted values include “left”, ”right”, and ”center”. Also an integer value can be specified as to a certain number of pixels from the left.

**Parameter name = valign** – this specifies where the text will appear in the “box” that we have specified for it. Accepted values include “top”, ”bottom”, and “center”. Also an integer value can be specified as to a certain number of pixels from the top.

**Parameter name = border** – this specifies if there will be a border drawn around the “box”. This feature can also be used to draw a blank box of a given size if the data element is left blank. Accepted values are “true”, “false”

**Parameter name = fill** - this specifies if there will be a grey fill as a background for the text in the “box”. This feature can be used to draw a gray box of the given size if the data element is left blank. Accepted values are “true”, “false”

**Parameter name = multiline** – (OPTIONAL PARAMETER) this specifies for really long text, if the text is supposed to go to the next line of a print area once it exceeds the width of the “box”. Typically this is not common since labels are meant to contain succinct information and if the text goes beyond the right edge of the box, it will be cut off. However, if a pamphlet is being made and the long description is being displayed as a multiline textbox at the bottom of the page, this feature makes sense.

### Image Item

This is the second type of Item. It specifies information about an image that will be inserted into the label template. For the Image item, there are 4 required attributes, and a number of optional attributes. The 4 required attributes are type,tagname, location, and source

**Type Attribute** – Will always be “image” in a text item

**Tagname Attribute**– it is required, but it can be whatever you think is adequate. However, the tagname should not be the same for more than 1 item. The system uses the tagname as a handle to reference a particular element, so it’s possible that problems could occur if a tagname were reused.

**Location Attribute** – it is required. It specifies what is the data source for this image. Possible values include “data”, “template”, and “parameter”

**Data** – If the location attribute is set to “data”, then, just like the text Item, a data attribute is required, and it will follow all the same rules. Most commonly in this case if it is a picture of the image, the image data is specified as data=”{ThumbnailUrl}” or data = “{ImageUrl} The thumbnailURL is usually large enough for labels. What this means is that the URL for the image is coming from the data file OR the additional product data.

**Template** – If the location attribute is set to “template” then that means the image URL is specified right here as an additional “value” attribute. This is the best way to go for logos. The reason one would use a location = “template” is because the image will stay the same for each label, and we don’t want the tool to download the image for every single label. That can seriously make things slow to download the same image 1000 times if your printing 1000 labels. In that case, use template, and the image will download once, and the tool will use it 1000 times – much faster! Also if you want the logo to be dynamic, be sure to set the value attribute to value=”{logo}” and the content of the logo will update based on your settings.

**Parameter** – It is possible for the tool to accept a location=”parameter”. However this will not be able to be used in the CED Label Print tool because it requires the software to feed an additional parameter. So, don’t use this one.

**Source Attribute** – it is required. This is used in conjunction with the location attribute. The location attribute only has 2 options “web” and “filesystem”. It tells the system whether the data that was referenced in the location attribute represented a web URL, or a filesystem location like “C:\Users\fLast\My Documents\Images\file.jpg”. With this in mind, it is way more common to have Web URLS but it is also possible to download all the files to your desktop and reference them as filesystem entries. Keep in mind filesystem entries are not found using additional product data, but must be fed in with you excel sheet data source. This is just because it is a personal path on your personal computer. Another time it is extremely common to use filesystem is when referencing a logo using the dynamic logo option (where you put value=”{logo}” because that actually references a logo that is stored within the program on your filesystem.

**Value Attribute** – (OPTIONAL ATTRIBUTE). This is only used when your location attribute is set to template **then it is required**. Then this is where you specify the value of the file. Remember, if you want a logo that can be configured by the tool, just set value=”{logo}” and the system will replace {logo} with a reference and that goes along with the source=”filesystem”

**Flip Attribute** – (OPTIONAL ATTRIBUTE). This is used to flip the image on its side or upside down. There are only 4 accepted values “0,90,180,270”.

The 6 required Parameters for the Image Item are:

Parameter name = “positionx” – position of the image from the left edge

Parameter name = “positiony” – position of the image from the top

Parameter name = ”width” – width in 100 pixels per inch

Parameter name = ”height” – height in 100 pixels per inch

Parameter name = “border” – “true” or “false” draws a box around image

Parameter name = “displayMode” – “contains”, “stretch”, “zoom”

### QR Item

This is the third type of item. It will encode the data you specify into a QR style barcode, and place it in the label where you specify. It is best practice to pick QR sizes that are exactly square and the width and height are divisible by 21. 63 is the most common width and height as specified in existing templates. This is because with the small amount of content included in our QR codes, the actual barcode is typically 21 boxes across by 21 boxes high. This way the output will be in the highest resolution possible.

The item has 3 required attributes and 5 required parameters. This should be pretty much the same for every label. The 3 required attributes are type, tagname, and data. The 5 required attributes are “positionx”,”positiony”,”width”, “height”, and “border”

**Type Attribute** – Will always be “qr” for a QR code

**Tagname Attribute** – is it required, but it can be anything. Since there’s typically only 1 QR code per label, it is recommended to keep it simple and use “bcqr” or “qrcode”

**Data Attribute** – this is required and follows all the same rules as the data attributes in the image items and text items. In CED, we typically encode, the manufacturer code followed by a space, followed by the catalog number. So how to do that is simply to specify the data attribute as follows: data=”{MfrCode} {CatalogNumber}”. Knowing the flexibility of the tool would give you an understanding of how to encode, for example the UPC into the qr code if you so chose.

Parameter name = “positionx” – position of the image from the left edge

Parameter name = “positiony” – position of the image from the top

Parameter name = ”width” – width in 100 pixels per inch

Parameter name = ”height” – height in 100 pixels per inch

Parameter name = “border” – “true” or “false” draws a box around image (not generally recommended with QR)

### Barcode 128 Item and Barcode 3 of 9 Item

These 2 Items types are almost identical and have very similar attributes and parameters. In fact, both of these Item Types are actually Text type, but they just each have a little special encoding. There are 3 required attributes, type, tagname, and data.

**Type Attribute** – This will be “bc128” or “bc39” respectively

**Tagname Attribute** – This can be anything, just be sure to keep them different from the other tagnames that are in use in your label.

**Data Attribute** – This follows the same rules as the other data attributes mentioned previously. To reiterate, the most common format for the content of our barcodes is the manufacturer code followed by the catalog number as in data=”{MfrCode} {CatalogNumber}”

For the parameters see the text item for additional details. But what you should note is that these types of barcodes leverage a font.

Barcode 128 – Parameter-”fontfamily”=”code128bWin”. If you don’t have that font installed, it won’t work correctly

Barcode 3 of 9 – Parameter- “fontfamily = “Free 3 of 9”. If you don’t have that font installed, it won’t work correctly.

The reason that the type bc128 and bc39 aren’t just text items is that the software has to encode those text items special before applying the font to them, and the tool takes that extra step out for you.

Also note that you don’t want to do anything unusual like apply a bold or italic font. It will probably just cause problems.

## Sheet Template

The Sheet Template Node defines the sheet characteristics for the labels. Some labels fit on a sheet 2 across, and are 7 rows per page. Other labels may be just one per page. Either way, it is important to define the <SheetTemplate> node.

The Sheet Template node has no attributes, and it has 10 required parameters.

**Parameter name=”marginleft”** - sets the left side margin of the label sheet. If you are doing 1 label per sheet, then use 0.

**Parameter name=”margintop”** – sets the top margin of the label sheet. If you are doing 1 label per sheet, then use 0.

**Parameter name=”width”** – sets the width of the individual labels. If you are doing 1 label per sheet, then use the papersize.

**Parameter name=”height”** – sets the height of the individual labels. If you are doing 1 label per sheet, then use the papersize.

**Parameter name=”xgap”** – this specifies the gap between columns of labels on the sheet. If there is only 1 column, or this is the only label on the sheet, then use 0.

**Parameter name=”ygap”** – this specifies the gap between rows of labels on the sheet. If there is only 1 row, or this is the only 1 label on the sheet, then use 0.

**Parameter name=”cols”** – this specifies how many labels there are across on the sheet. If this is the only label on the sheet, then use 1.

**Parameter name=”rows”** – this specifies how may rows of labels there are on the sheet. If this is the only label on the sheet, then use 1.

**Parameter name=”angle”** – this specifies the angle of the whole label. Most the time this will be 0. There are some labels that print horizontally however. The tool does even support labels at any degree.

**Parameter name=”resizeMode”** – the typical value for this will be “none”. This is used in conjunction with the angle parameter. If the whole label is being turned by 90 degrees, there are various options. The most logical option in that case is “contains”. However, the total list of options are “none”,”contains”, and “zoom”.

## Text Example Avery 5162 Output (for copying and editing)

<xml>

<LabelPrint>

<Paper>

<Parameter name="width">8.5</Parameter>

<Parameter name="height">11</Parameter>

<Parameter name="printwidth">850</Parameter>

<Parameter name="printheight">1100</Parameter>

</Paper>

<LabelTemplate>

<Item type="text" tagname="description" data="{ProductName}">

<Parameter name="fontfamily">Calibri</Parameter>

<Parameter name="fontsize">11</Parameter>

<Parameter name="fontstyle">bold</Parameter>

<Parameter name="positionx">80</Parameter>

<Parameter name="positiony">0</Parameter>

<Parameter name="width">200</Parameter>

<Parameter name="height">17</Parameter>

<Parameter name="halign">left</Parameter>

<Parameter name="valign">top</Parameter>

<Parameter name="border">false</Parameter>

<Parameter name="fill">false</Parameter>

</Item>

<Item type="text" tagname="minmax" data="{Min}/{Max}">

<Parameter name="fontfamily">Calibri</Parameter>

<Parameter name="fontsize">10</Parameter>

<Parameter name="fontstyle">normal</Parameter>

<Parameter name="positionx">80</Parameter>

<Parameter name="positiony">34</Parameter>

<Parameter name="width">242</Parameter>

<Parameter name="height">17</Parameter>

<Parameter name="halign">left</Parameter>

<Parameter name="valign">top</Parameter>

<Parameter name="border">false</Parameter>

<Parameter name="fill">false</Parameter>

</Item>

<Item type="image" tagname="logo" location="template" source="filesystem" value="{logo}" flip="270">

<Parameter name="positionx">295</Parameter>

<Parameter name="positiony">20</Parameter>

<Parameter name="width">90</Parameter>

<Parameter name="height">90</Parameter>

<Parameter name="border">false</Parameter>

<Parameter name="displaymode">contains</Parameter>

</Item>

<Item type="qr" tagname="bcqr" data="{MfrCode} {CatalogNumber}">

<Parameter name="positionx">10</Parameter>

<Parameter name="positiony">10</Parameter>

<Parameter name="width">63</Parameter>

<Parameter name="height">63</Parameter>

<Parameter name="border">false</Parameter>

</Item>

</LabelTemplate>

<SheetTemplate>

<Parameter name="marginleft">0</Parameter>

<Parameter name="margintop">70</Parameter>

<Parameter name="width">392</Parameter>

<Parameter name="height">133</Parameter>

<Parameter name="xgap">19</Parameter>

<Parameter name="ygap">0</Parameter>

<Parameter name="cols">2</Parameter>

<Parameter name="rows">7</Parameter>

<Parameter name="angle">0</Parameter>

<Parameter name="resizemode">none</Parameter>

</SheetTemplate>

</LabelPrint>

</xml>