**BMWS Standard Feed Format**

Aug. 30, 2013

Product venders supply product data (Import Data File) to BMWS in non-standardized formats with varying amounts of information. Every detail of each product must be translated into a standard CSV file which BMWS may import. Developers are expected to thoroughly read and understand this document before beginning any work.

**Basic Process**For each product data Import Data File that is provided to BMWS by a vender, a Python script must be written to extract the product data from it. The developer will be provided with a sample export data file, a blank CSV file containing the expected output header and a template Python script file to work in. Upon completion of the Python script the developer will submit via e-mail a sample output as well as the script file for review. If the output is acceptable the task will be considered complete.

**Python, Script and Library Dependencies**The version of the Python interpreter that the Python script will be run with is 2.7.4. The module name will be provided with the assignment e-mail. The Argparse and Logging modules from the standard library are used. The command line arguments are:

|  |  |
| --- | --- |
| **--infile** | *Path to the input CSV file* |
| **--outfile** | *Path the output CSV file should be written to* |
| **--debug-level** | *The level to configure logging to. 'debug', 'info', 'warning', 'error', 'critical'* |

Additional arguments may be added as necessary, but they must have sane default values that allow the script to operate normally. The standard logging library should be used for all output to STDOUT or STDERR. The script file should contain no **print** statements. Non-essential output should be written with **logger.debug(“***text”***)**.

If the script requires additional dependencies, they should be packaged together and included with the script when it is submitted. The dependencies must be documented at as part of the doc string in the head of the Python module.

For information regarding the use of argparse or the logging module please refer to the following documentation:

|  |  |
| --- | --- |
| **logging** | [*http://docs.python.org/2/howto/logging.html*](http://docs.python.org/2/howto/logging.html) |
| **argparse** | [*http://docs.python.org/2/howto/argparse.html*](http://docs.python.org/2/howto/argparse.html) |

**CSV Output Format**All output files must be written as UTF-8 encoded CSV files. Text fields must be wrapped in double quotes. If a text field contains a double quote, it must be escaped with another quote. For example:

"This string has ""doubled quotes"" in it."

Each line must be terminated by a Windows newline character sequence of linefeed, carriage return **(** "\r\n" **)**.

An empty CSV file containing only the expected header will be provided to the developer. Additional headers that should be added but were not included in the empty CSV file must be approved before they are added.

**The 'advertise' Field**Some vender products are provided to BMWS that should not be sold directly to consumers. In this case the special '**advertise**' field is used to denote that. If a product may be sold directly then the feed field value must be **"True"**, otherwise the value must be **"False"**. No other Boolean values will be accepted. This concept will be explained in greater detail in the **Set Links and Collections** section. In some cases it may not be possible to determine the value of this field programmatically. In these cases, the field should be left blank.

**Size Links and Color Links**Products may be grouped together into logical units under different circumstances. When a product is available in multiple sizes or colors then an ID must be generated to identify that unique set of options and applied to each product that makes up part of the group. The two special fields '**sizelink**' and '**colorlink**' are used for that purpose. If the vender's feed does not already provide such an ID then the ID shall be the SKU of one of the products of that set. Generally this would be the first product encountered that belongs to that set. The example below illustrates this concept.

|  |  |  |
| --- | --- | --- |
| **sku** | **color** | **colorlink** |
| abc123 | red | abc123 |
| abc124 | white | abc123 |
| abc125 | blue | abc123 |
| abc126 | green | abc123 |
| abc127 | purple | abc123 |

*\* Note that the first sku encountered is abc123, so it becomes the colorlink ID*

**Collections**A collection consists of products that are designed to go together. Care must be taken to identify items that belong to a collection and properly enter the collection name in the collection column.

**Groups**Groups are products that are sold together as a set. An example would be a Dining Room Set which may include a table and chairs. Occasionally the vender import feed will have a group record with a separate SKU for the group, but not always. If a Group record with its own SKU does not exist, one will need to be created for each Group. The SKU’s of the products that make up the group will need to be written to the “SKU” column(s) of the feed. If there are multiple identical items as sub-SKU’s (example 4 chairs), then list the same SKU 4 times in the sub-SKU column (sub-SKU1, sub-SKU2, sub-SKU3 and sub-SKU4). Sometimes chairs would be sold in boxes of 2 chairs, and the group would include 2 boxes of 2 chairs to total 4 chairs, so the group sub-SKU for the chairs would be entered 2 times. If an item is a group items, the column Group should be marked True, otherwise, it should be False.

Another common occurrence you will find that will require grouping is as follows. A vender offers a Dining Room Table that includes a table base with a glass top. These items will be marked as **"False"** under the '**advertise'** column as we do not want to sell individual parts or components of a piece of furniture, but a group item (record) needs to be created for the “Table” which would include the sub-SKU of the table top and table base.

**Images**

* Main Image shows the product (if the product is a chair, the image would be of a chair)
* Set Image shows the product as part of a set (if the product is a chair, the image would be of a table and Chair Set, there will likely be multiple set images for one product)
* Room Image shows the product in a room (if the product is a chair, the image would be of a room full of furniture, maybe a table and chairs, hutch, buffet cabinet, utility cart, etc….)
* Additional Image will be another image, like a close up of a drawer, may the unit with the doors open, etc…
* Diagram Images – Instructions, Line drawing or other plans or sketches or pdf’s but not photographs.
* Swatch Images – Color samples or Fabric Choices

Images file names must be included in the output file. Every product will have a main image. Some products may have additional images and some images may be used for more than one product in the case that the image is a set image. The main image fields are **'mainimage1', 'setimage1'**, **'additionalimage1',’roomimage1’, ‘diagramimage1’, ‘swatchimage1’**. The developer should add additional image fieldsas required by adding extra enumerated columns such as **'setimage1'**, **'setimage2', 'additionalimage1'** and **'additionalimage2'**. The fields should contain image file names only, not web URLs.

The developer must **not** write code which downloadsimages in the case that an image URL is provided by the vender. You will have a list of image names in “image list.csv”. You will need to determine if an image is a main image, additional image, etc... by examining the image files names and photos. In most cases, you will see words, like “detail” in the file name that will tell you it is an additional image. If the file name is just the SKU, then it is likely a main image. If it has 2 SKU's in the file name, it is likely a set image for the 2 items combined. If the word room or bedroom is in the name, it is likely a room view. You can also look at the import file for picture name hints. For example, if an item is a part of a group, then the group item image would be included as the group image for all items in the set. The room view image should be included in all items that make up sets in that room view, plus groups in that room view.

A python script, imagechecker.py, is provided to the developer that will read the output feed against a file containing a list of image names expected to be found in the feed. imagechecker.py will report any problems that it finds in the output feed. It does not report every possible error or missing image file names, it only reports on the errors it was written to detect. If imagechecker.py does not produce an empty error report from output feed then the job is considered incomplete. Unresolved errors must be reported.

**Maintainability and Style Guidelines**The developer is expected to design and implement the Python module in such a way that it would not be difficult for others to read the code and maintain it as necessary. The code should be properly formatted according to the PEP-8 style guide that may be reviewed at <http://www.python.org/dev/peps/pep-0008/>. Style validation may be performed using PyFlakes, the pep8 command-line program or via the web form at <http://pep8online.com/>. Functions, classes and modules are expected to have doc strings describing the intent of each section of code. In-line comments are not required unless the code is particularly complex. All variable names and comments must be in English. Code that does not validate, or that is not properly documented will be rejected and the work will be considered incomplete.