**Introduction**

The script split.js is a NodeJS based javascript that performs roll splitting or similar functionality similar to bulb move operation.

**Split**

Currently a complete list of UID for tags are saved in a single file in comma separated value format (csv), the task of the split script is to automate the task in splitting blocks of tags from the original csv file into one or more new csv files.

1. Single block split

product.csv

starttag

B70430742E67732F4F43773530665CE1

…

…

…

…

end tag

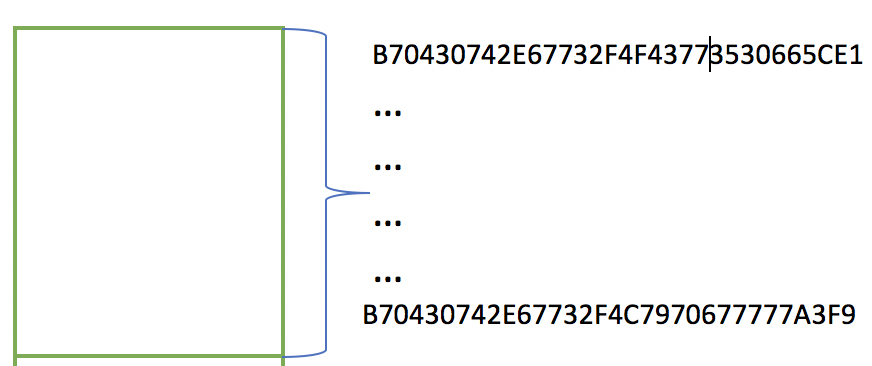
B70430742E67732F4C7970677777A3F9

The start tag and end tag define a single block of UID, and subsequently multiple pairs of start tag and end tag can be used to define multiple blocks inside a roll.csv file. In the above diagram, start tag is B70430742E67732F4F43773530665CE1, end tag is B70430742E67732F4C7970677777A3F9.

The split script will look at the input csv file *product.csv*, based on the specified *start tag* and *end ta*g, generate one output csv file with all the tags inside the block including the start tag and end tag.

The content in the new file will look like the following diagram, the output file will have a name similar to the original file name but appended with a \_split suffix, in this case *product\_split.csv*.

product\_split.csv



1. Multiple Blocks split, and merge into a single block

product.csv product\_split.csv

Block A

Block A

Block C

Block B

Block C

Block B

1. Multiple Block Split for multiple new products

Because of the scripting logic, we can easily change the merging behavior into saving separate blocks into separate files to achieve similar functionality of the bulb-move operation by providing a different set of parameter to the split script, see the usage section below.

product.csv product\_split\_a.csv

Block A

Block A

Block C

Block B

product\_split\_b.csv

Block B

product\_split\_c.csv

Block C

**Split Usage**

1. Installation of the split script from split.zip

This split script is written using Javascript and node.js, admin of the script should have familiarity in setting node.js, and be able to run simple procedures to setup the libraries that are used by the script. The steps needed are

Uncompress the split.zip package in the working directory and run the following CLI commands

$ cd ./split

$ npm install

Use editor to update two environment variables pointing to the backend database if validation against database is required.

$ vim .env

DATABASE\_URI='mongodb://localhost:27017/thinfilm\_dev'

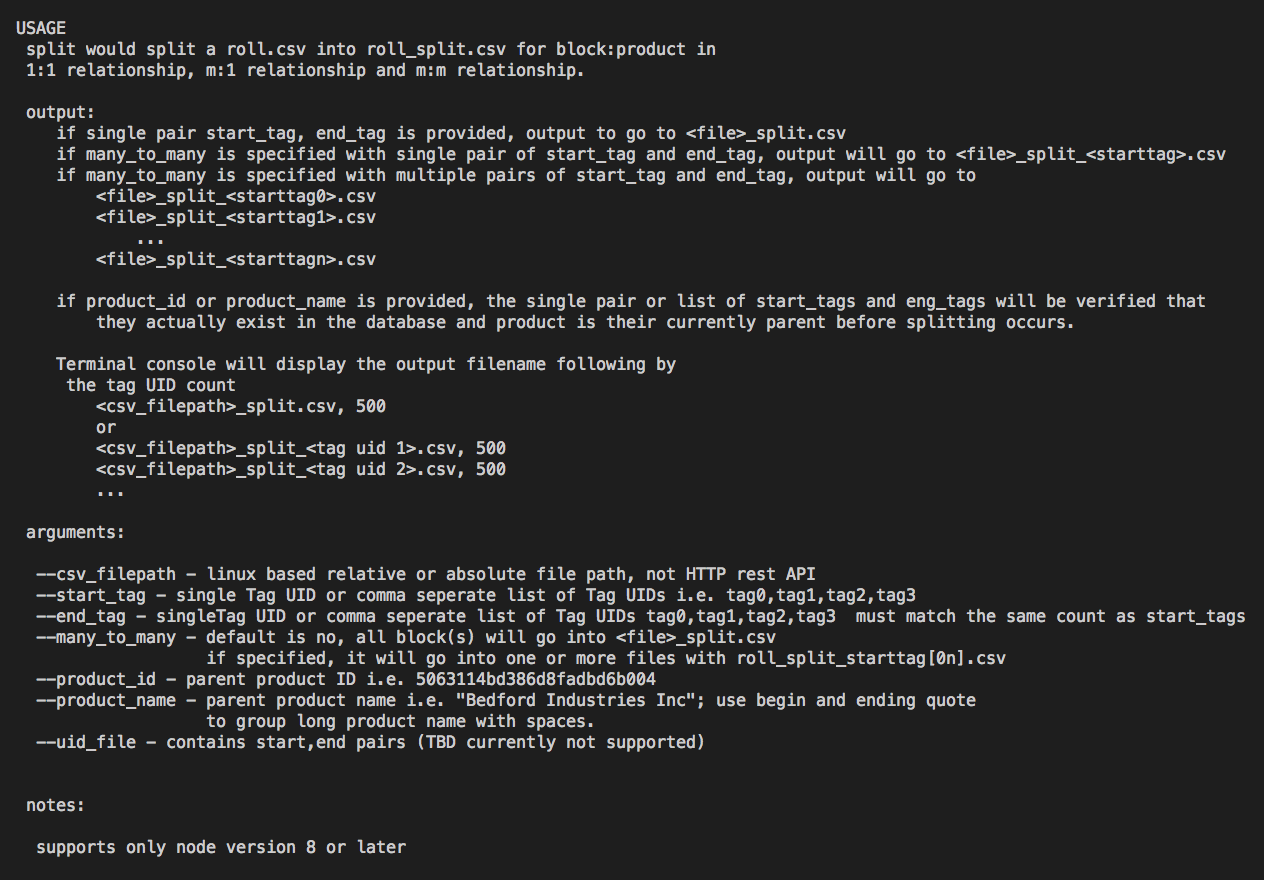
SOURCE\_DB='thinfilm\_dev'

1. Displaying the split script usage screen on a console terminal

Run the CLI command

$ node src/split.js

You should be able to see all the parameters needed for properly running the split operations as illustrated in the below diagram:

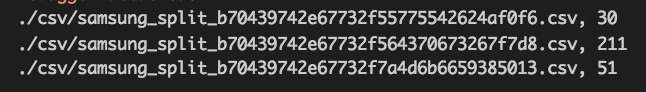


A success split operation will also the corresponding output of list of files and the corresponding tag count inside the file.

The following output shows 29 tag IDs are extracted from P011707033.csv and saved in P011707033\_split.csv



The following output shows 292 tag IDs are extracted from P011707033.csv and saved in separate P011707033\_split\_.csv files ready for further operations.



1. Arguments for the split script

In general, arguments are composed of key value pairs linked by an “=” equal sign, for example

–start\_tag=B70430742E67732F4F43773530665CE1

is referring to the start\_tag key with the UID value of B70430742E67732F4F43773530665CE1.

There is exception too where an argument can be provided with the corresponding value.

* --csv\_filepath

This argument points to the file path of the csv file where tag ID blocks will be extracted, the csv file is required to be in a clean state i.e. coming out of a SAP/SY/Conversion folder. The script supports relative and absolute path location. It currently does not support user directory ~ expansion.

* --start\_tag

This argument points to the list of the start\_tag of all the blocks to be extracted from the input csv file. If a list of start tags is provided, it is expected that ids in the list be separated by comma. At least one tag is needed. The tag ID’s line number in the input csv file is looked up by the script.

* --end\_tag

This argument points to the list of the end\_tag for all the blocks to be extracted from the input csv file. If a list of end tags is provided, it is expected that each ids in the list be correctly aligned with its’ corresponding start tag in the start\_tag list to avoid range overlaps.

* --many\_to\_many

This argument tells the script to instead of merging all the blocks or a single block into a single file, save them into separate output csv files. Each output file will be uniquely identified with its’ start tag suffix delimited by an underscore character.

* --product\_id

This argument tells the script to carry out database validation by looking up every individual tag id in the ‘*to be extracted’* block to be a real entry in the database for the specified product. The id is the numeric portion of the Products document in the database. For example, the Samsung conversion may have a product id of ‘Products$349cdb9434726c34671e’, the script expects only the numeric portion of 349cdb9434726c34671e to be provided.

* --product\_name

If access to the database is not available, user can specify the name of the product, the script will perform one extra step to lookup the product id prior to the validation process. If spaces is used in the name of the product, a leading and ending double quote is needed to capture the only name to be passed correctly to the script.

1. Errors currently being checked by the script

* Script detects the one of more of the tag block from the csv file cannot be validated against the database

`input split block from "${r.product\_name}" can not be validated against the database record, ${r.splitError}!`

* Script cannot read the csv file from the file path argument

`error: reading file ${JSON.stringify(error, null, 2)}`

* Script cannot write extracted tags into the output file.

`failed to write buffer in ${output\_file\_name} due to ${JSON.stringify(error, null, 2)}`

* Script detects that the product name may have variant in the database

`error: more than 1 product is found with the same name`

* Script detects that the product does not exist in the database

`error: product name ${product\_name} not exists in database`

* Script detects that the start end tags block have overlapping each others.

`start tag ${stag.uid} at line (${stag.offset}) is found overlapping with others`

* Script detects that the start tags does not exist in the input csv file

`start tag ${stag} not found in csv file`

* Script detects that the end tags does not exist in the input csv file

`end tag ${etag} not found in csv file`

* Script detects that the start end tag list does not match as pair

`error : cannot found complete start end tags as pair(s)`

* Script detects that the tag codes from the database has integrity issues and may not be reliable.

`error found linking error in joining tag and tagCodes documents`

* Script detects that it is running in an unreliable environment.

` split must be running on node version 8 and later`

*Notes\* there may be more system level errors that can trigger abnormal script abort which are not documented here, for example, the database is temporary out of service etc.*

1. Example on split script usage

3 blocks from Parent Product

$node /src/split.js

--product\_name="Bedford Industries Inc”

--csv\_filepath=./csv/P011707033.csv

--start\_tag=B70430742E67732F4F43773530665CE1,B70430742E67732F6C554F39345711C7,B70430742E67732F62744B775368C618

--end\_tag=B70430742E67732F4846567632323B8F,B70430742E67732F4B327239654527D8,B70430742E67732F326B52737869D774

$node /src/split.js

--many\_to\_many

--product\_name="2016-04 Engineering and marketing samples split from Altria sub-batch 0 (Samsung SEA)"

--csv\_filepath=./csv/samsung.csv

--start\_tag=b70439742e67732f55775542624af0f6,b70439742e67732f564370673267f7d8,b70439742e67732f7a4d6b6659385013

--end\_tag=b70439742e67732f77335836714f1190,b70439742e67732f4a42615a7547037f,b70439742e67732f3368436b7a7606a1