|  |
| --- |
|  |
| Batch Validator |
|  |
| Implementation Documentation |

**Sky business technology**

|  |  |  |
| --- | --- | --- |
| REVISION DATE | AUTHOR | COMMENT |
| February 6, 2013 | Dodds, Stuart | Initial Document Creation |

Batch Validator

Implementation Documentation

# Introduction

The batch validator was created in order to help automate the process of receiving files from the business.

Currently the process for running a batch file consisted of the following steps:

* Business Create a file
* Business Email us the file
* File was given to the team that dealt with that batch job
* Someone in that team would manually check the file to ensure it contained the correct data
* This team member would FTP the file into the correct location
* Batch Ops would be emailed asking for the job to be scheduled.

This process took a lot of time, especially the manual checking of each file; most of the failures would come from human errors within the file.

This system aims to limit the number of human errors by rejecting the file before it is ran and forcing the business to fix the errors before allowing these batch files to run.

# New Process

The batch validator is a PHP web based application which the business will access using the corporate intranet. This is accessible in the following location: [TODO].

The process for the business is as follows:

1. The business will create their batch file, go to the above address and login using their windows username and password. (Providing they have been given access to the application)
2. Once they have logged in they will be shown a page that will allow them to select a batch job.
3. When they click continue they are presented with a form that will allow them to upload the batch file that they have created.
4. The batch validator will then process the fill line by line (using PHPEXCEL) and run validation on each field that is flagged as a specific type. For example: telephone numbers, required fields etc…
5. As this validation is running if the file contains any address information this is collected and stored in an array.
6. Once the validation has been completed the array of addresses is split into groups of 100 and submitted to QAS for verification.
7. The batch validator will then display the results of this file:
   1. If the file has been validated and no errors are found then they may continue
   2. If the file has failed validation then they are unable to proceed and will be expected to fix the displayed errors before attempting another upload.
8. Once validated they are then allowed to submit the file to be ran. Currently this process involves a form being filled in containing their Approving Managers name, what time they would like it to be run and any comments they desire, they then submit the file.

This means the file has now been verified and uploaded onto our webserver; this file should now be in the following directory: $BATCHVALLOCATION/uploaded/validated/$JOBNAME/$FILENAME

Support team will now login to the batch validator, as a support team member you will have admin access which will allow you to approve uploaded files as well as do a number of tasks that the business do not have access too.

Changing the status will email the user who uploaded this file with updates to show how the job is progressing.  
(This will change in the future when it is integrated with SPARK which will then provide the approving of files)

Now that you have changed the status (effectively updating the business of the progress for you) the file must be moved from the above location to the location that is expected by control-m.

The next step is to email batch operations and ask them to run the batch job.

After each step you should update the status of the file in the admin panel to keep the business in the loop.

## Business Process



## Support Process



# Code

In this section I will outline the file hierarchy, how the code is implemented and what each of the complicated functions do.

## File Hierarchy

The project is split into several different files:

**index.php** - Is the general display page that contains all the html header information such as the style import, JavaScript/jQuery and includes the lib.php page. It sets up the page by calling the various different methods that then create blocks of the page for example: menubar() will create a menu bar.

**lib.php** – Contains a library of all the functions that will be used throughout the application, mostly this contains display functions that will create particular pages or functions that will be used across pages.

**proc.php** - Contains all the functions for processing the validation script and any ajax calls that are required.

**db.inc.php** – Contains the database connection information, including the username and password for the file validator database. This also contains functions that when passed a SQL statement will return the results.

**example.php** – This file will display a batch job template when given the batch job name example: SBBBLK053, this is done via a $\_GET[‘batchname’] request in the url. For Example: *example.php?batchname=SBBBLK053* would display the template for this job.

**download.php** – Allows the user to download the csv template shown on the example.php page, this is triggered by using the “Download Example” button show on that page.

**email-template.php** – In order to display graphical emails this page was created to hold the templates for each type of email. (In order to create these graphical emails, everything is displayed in tables – this made the files massive and this is why they are in their own file), just call the functions in this file to generate the email content.

**PHP-Excel.php** – This is a library that has been included from the PHP-Excel libs, this allows the application to use the library to read csv/excel/xml files.

**qas-batch.php** – A PHP class file that will send addresses to QAS and will return the responses in an object.

**windowsauth.php** – This file contains 1 function which expects 2 inputs, a username and password. This will then send these to the Corporate Windows LDAP, if these match it will return true, if they fail it will return false. *NB: If this is wrong five times you will be locked out of your windows account.*

The rest of the structure matches essentially what it does, for example: styles contains style sheets, js contains javascript (including jquery libs), scripts have more js scripts (this could probably be moved).

### Upload Directory



The upload directory will resemble something like the above. Every time a user uploads a file this file needs to be on the host somewhere before the application can use PHPExcel to process it and then validate it. Therefore the upload folder has been split into 2 different directories.

The tmpUpload folder will contain any files that have been uploaded to be processed.

If and only if this tmp file is validated will it enter into the validated directory, It will then be copied into the subdirectory that matches the batchname of this job.