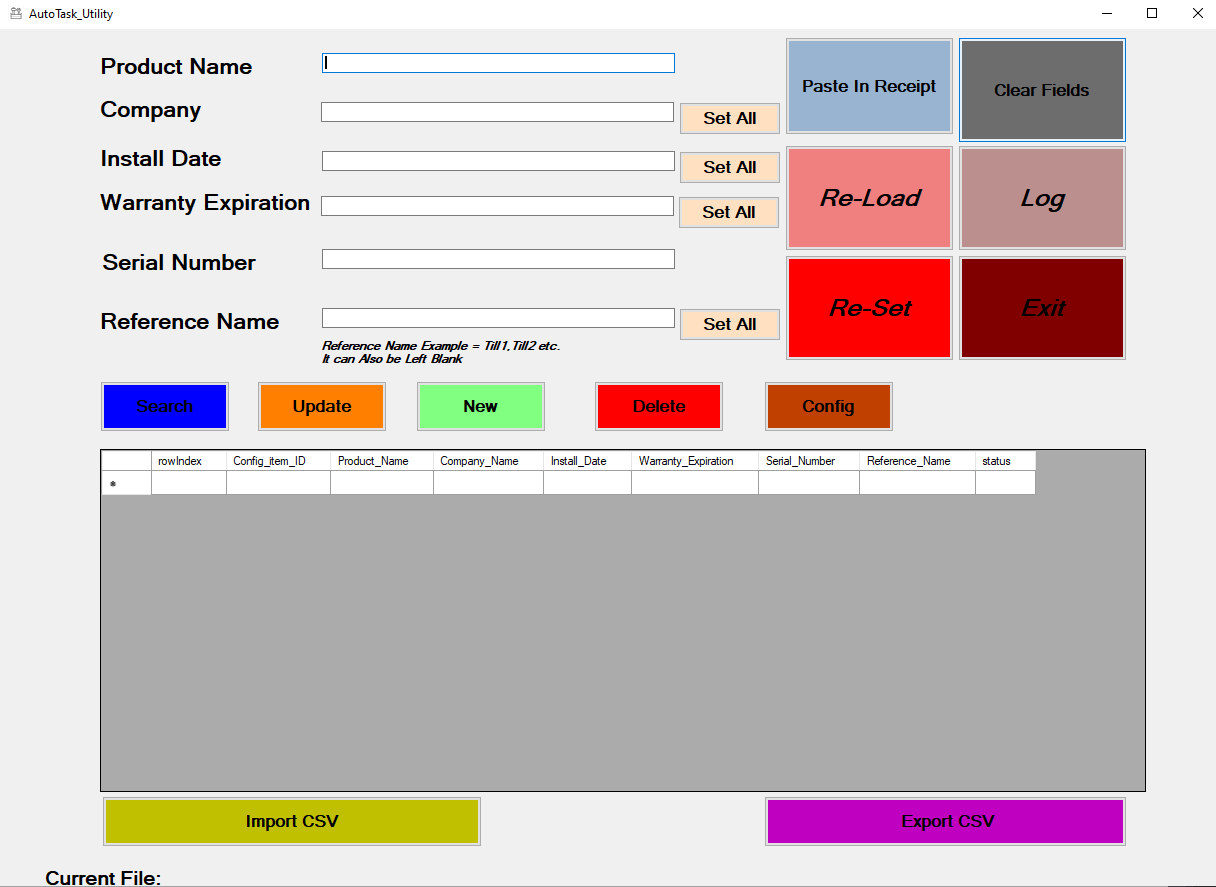
WinRetail/ Auto Task Utility SDS



# Table of Contents

[1. Table of Contents 2](#_Toc76630219)

[2. About this Document 4](#_Toc76630220)

[2.1 Document Versions 5](#_Toc76630221)

[2.2 Acronyms & Abbreviations 5](#_Toc76630222)

[2.3 Notation 5](#_Toc76630223)

[3. System Overview 6](#_Toc76630224)

[3.1 High Level Overview 6](#_Toc76630225)

[3.2 References 7](#_Toc76630238)

[4. Architectural Strategies 8](#_Toc76630239)

[4.1 Overview Diagram of GUI 8](#_Toc76630240)

[4.2 Paste In Receipt 9](#_Toc76630241)

[4.3 Import CSV 12](#_Toc76630242)

[4.4 Update Function 13](#_Toc76630243)

[4.5 Delete Function. 15](#_Toc76630244)

[4.6 New Function 16](#_Toc76630245)

[4.7 Set All Function 17](#_Toc76630246)

[4.7.1 Set all Company. 17](#_Toc76630247)

[4.7.2 Set All Install Date 17](#_Toc76630248)

[4.7.3 Set All Reference Name. 17](#_Toc76630249)

[4.7 Clear fields Function. 17](#_Toc76630250)

[4.9 Reload Function 18](#_Toc76630251)

[4.10 Reset Function. 20](#_Toc76630252)

[4.11 Exit Function 20](#_Toc76630253)

[4.12 Config Function 21](#_Toc76630254)

[4.12.1 Mapping Config Form 21](#_Toc76630255)

[4.12.2 Mapping Config Form Error 22](#_Toc76630256)

[4.12.3 Mapping Config Form Delete Button 24](#_Toc76630257)

[4.12.4 Mapping Config Form New Button 25](#_Toc76630258)

[4.12.5 Mapping Config Form Import Function 26](#_Toc76630259)

[4.12.6 Mapping Config Form Export Function. 27](#_Toc76630260)

[4.12.7 Mapping Config Form OK Function 28](#_Toc76630261)

[4.12.8 Mapping Config Form Exit Function. 30](#_Toc76630262)

[4.13 Search Function 30](#_Toc76630263)

[4.13.1 Search Via Product Name 31](#_Toc76630264)

[4.13.2 Search by Serial Number 32](#_Toc76630265)

[4.13.3 Search By Reference Name 33](#_Toc76630266)

[4.14 Export CSV 34](#_Toc76630267)

[5 Receipt Parsing 36](#_Toc76630268)

[5.1 Example of Receipt 36](#_Toc76630269)

[5.2 Data Extraction from Receipt 37](#_Toc76630270)

[5.2.1 Product Description 37](#_Toc76630271)

[5.2.2 Company Name 37](#_Toc76630272)

[5.2.3 Serial Number 38](#_Toc76630273)

[5.3 Example of CSV Column Headers 38](#_Toc76630274)

[5.3.1 Configuration Item ID [updates only] 39](#_Toc76630275)

[5.3.2 Reference Name 39](#_Toc76630276)

[5.3.3 [required] Install Date 39](#_Toc76630277)

[5.4 Design Patterns in Receipt 39](#_Toc76630278)

[5.4.1 Serial Number Pattern 39](#_Toc76630279)

[5.4.2 @ Symbol Pattern 41](#_Toc76630280)

[5.4.3 Two Line Entry Patterns 42](#_Toc76630281)

[5.4.4 Single Line Entry 42](#_Toc76630282)

[6 Handling Refunds 44](#_Toc76630283)

[6.1 Error In Handling Refund 48](#_Toc76630284)

[7 Handing Voided Items 50](#_Toc76630285)

[8 Exceptions and Error Handling. 52](#_Toc76630286)

[9 Configurations 52](#_Toc76630287)

[10 Other Functionality 53](#_Toc76630288)

[10.1 Log 53](#_Toc76630289)

[10.2 Privacy & Security. 55](#_Toc76630290)

[10.3 Deployment, Support & Maintenance 55](#_Toc76630291)

[10.4 Installation Files 55](#_Toc76630292)

[11 Open Questions & Issues 56](#_Toc76630293)

[12 Contingency and Unknowns 57](#_Toc76630294)

[13 Estimate Development Schedule 58](#_Toc76630295)

[14 Testing 59](#_Toc76630296)

[APPENDIX 1 60](#_Toc76630297)

# 2. About this Document

|  |  |
| --- | --- |
|  |  |
| **Version** | Final |
| **Creation Date** | Monday, April 19, 2021 |
| **Last Modified** | Thursday, July 08, 2021 |
| **Author(s)** | Liam Coleman |
| **Contributor(s)** | James Forde |
| **Reviewer(s)** | James Forde/Micheal Cattigan/ Noel Quinn |
| **Audience** | Service Manager/Group manager |
| **Purpose** | To develop a high-level overview, and what is needed in the proposed integration utility between WINRETAIL and Auto Task. This will include: the development of a simple GUI which has the ability to update, delete et cetera.  To develop a high-level overview, what is needed in the proposed integration utility between WINRETAIL and AutoTask. This will include: the development of a simple GUI which has the ability to update, insert new, delete et cetera.  To include high/low level design details for the development team and test teams so that it can be implemented and tested accordingly.  It is also envisaged that this document will highlight any deployment/support/maintenance issue that might arise. |

## 2.1 Document Versions

Listed below are the document versions.

|  |  |
| --- | --- |
| **Date** | **Description** |
| 19 April 2021 | First Draft |
| 20 May 20, 2021 | Revised |
| 25 May 25, 2021 | Revised |
| 08 July 2021 | Final |
|  |  |
|  |  |
|  |  |

## 2.2 Acronyms & Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Meaning** |
| **DB** | Database |
| **CBE** | Central Business Equipment (Concannon Group) |
| **AT** | AutoTask |
| **EPOS** | Electronic Point Of Sale |
| **GUI** | Graphical User Interface |
| **SDS** | Software Design Document |
| **SRS** | Software Requirements Document |
|  |  |

## 2.3 Notation

**Bold** is used to emphasis important items or possible code Functions.

Code extracts in this font. Formatting may be applied to keywords.

First SDS Review meeting notes appear in this font.

# 3. System Overview

The proposed software utility (which should be referred to as utility from now on in this document) will be a piece of software that will interact between WINRETAIL and AutoTask. The overarching goal of this utility is to:

* read the journals which house receipts.
* take the pertinent information from these receipts.
* then dump this information into a CSV file.
* Which is AT friendly for upload.

The GUI should allow for users to interact with WinRetail in a simple straightforward matter. To logon to the utility CBE’s existing active directory should be used. Users will be able to edit existing entries, delete existing entries and add new entries into the CSV file that will be outputted (NB: Not the WinRetail DB). To keep the user on track, the GUI will have a data grid view which will house a predetermined number of entries that will be in the CSV file (Figure 1 Main GUI).

## 3.1 High Level Overview

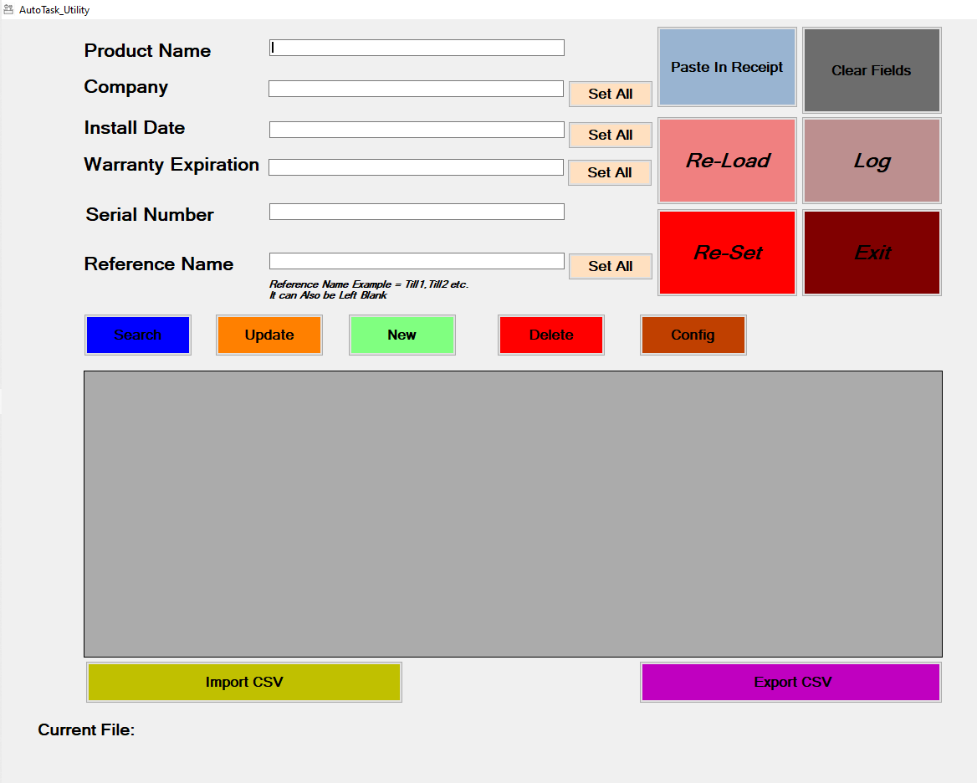


Figure 1 Main GUI

## References

This document and subsequent drafts.

# 4. Architectural Strategies

## 4.1 Overview Diagram of GUI

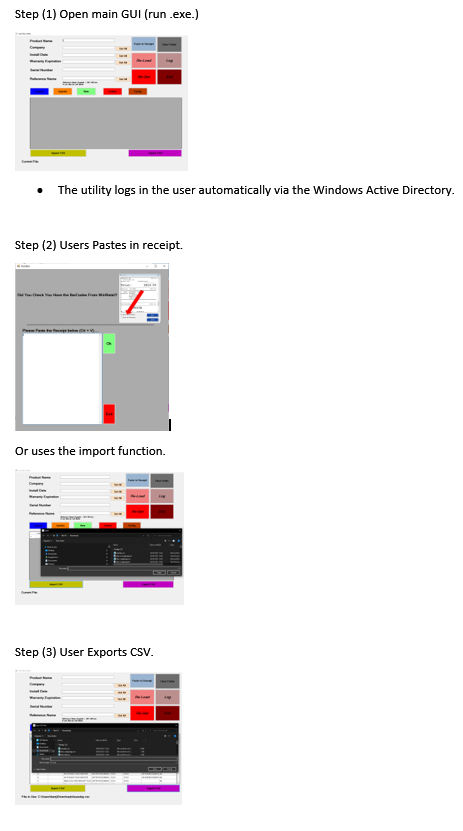
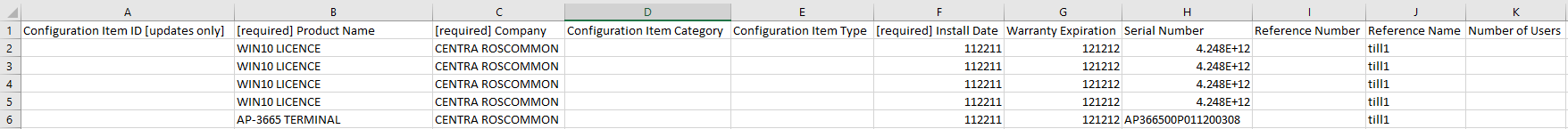


Figure 2 Utility Overview

As per figure 2 above:

* Step 1: the utility GUI opens.
* Step 2: allows the users to create, delete, update and entry et cetera by uploading an existing CSV file, or pasting in a new receipt ad creating a new CSV file.
* Step 3: outputs the new CSV file or the modified CSV file.

Overview of an AutoTask compatible CSV file:



## 4.2 Paste In Receipt

When the user starts the application, the grid will be Blank (Figure 3 Blank Grid).

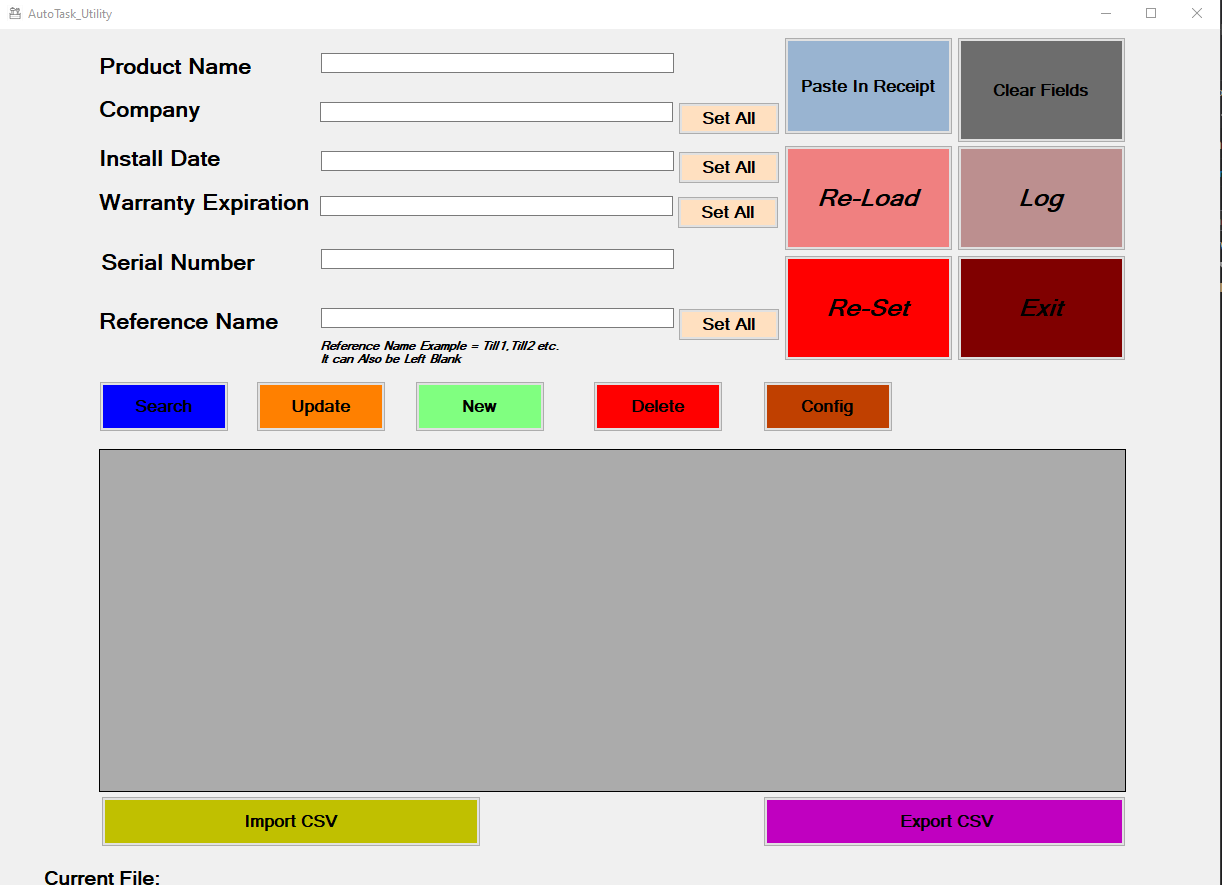


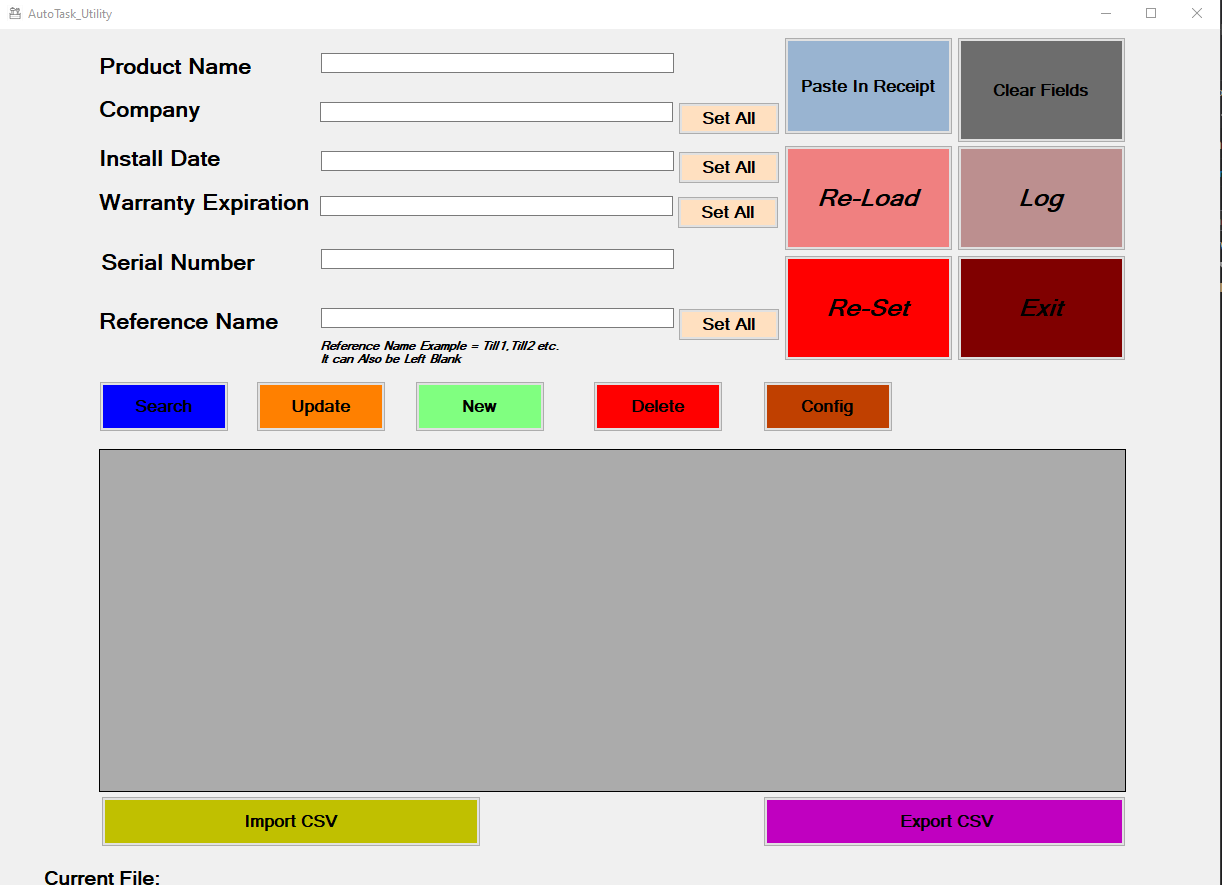
Figure 3 Blank Grid

The user has 2 choices:

1. Paste in a receipt.
2. Import a CSV.

The latter will be described in the next section.

To paste in a receipt the user clicks the Paste in receipt Button.



The PasteBox form will appear (Figure 4 PasteBox Form),



Figure 4 PasteBox Form

A reminder is displayed at the top of the form, this is important, as it reminds the user to check the ‘show with barcodes’ box when extracting the receipt from WinRetail journals (Figure 5 Show with Barcodes Checkbox ). NB: The utility needs these barcodes to work.

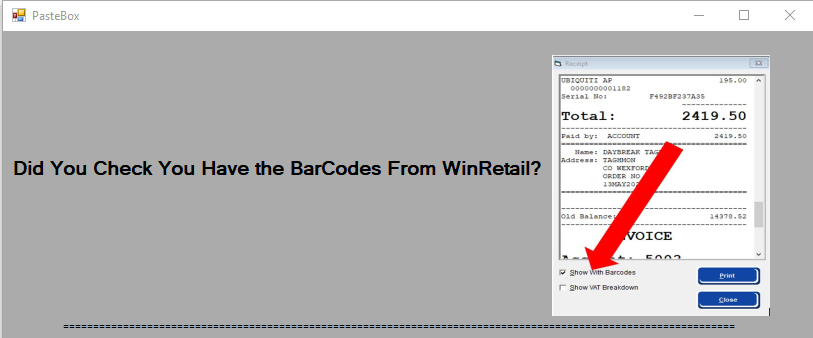
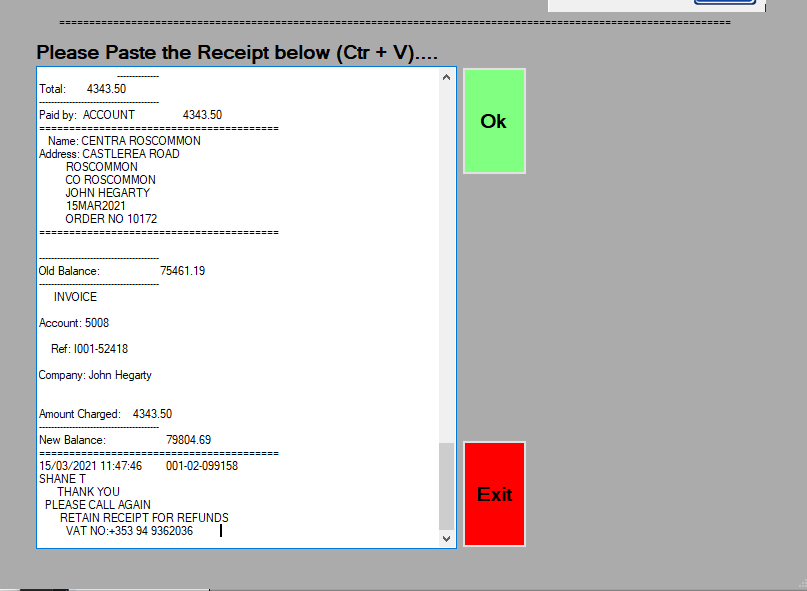


Figure 5 Show with Barcodes Checkbox

Once the user has copied all the info from the receipt in WinRetail, they

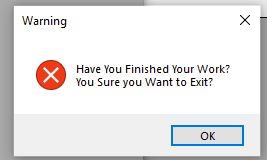
Paste it into the past box by clicking on it and pressing Control +V on the keyboard. Note: More than one receipt can be pasted in.



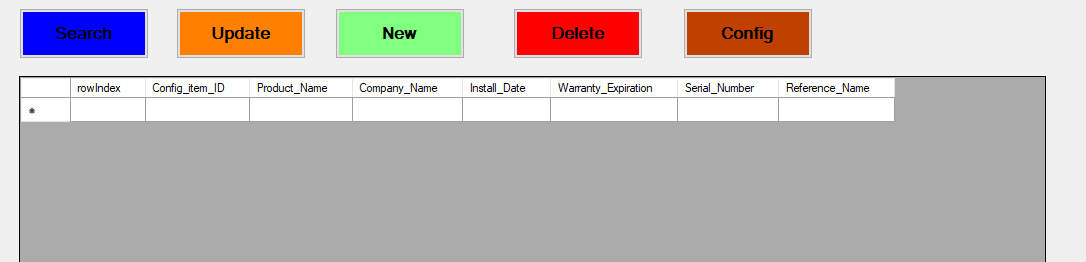
If they have made a mistake and want to start again, they click Exit,



A message box will appear,



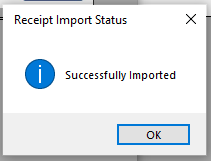
If they click ok it will exit the PasteBox form, return to the main GUI, and the grid will not be populated.



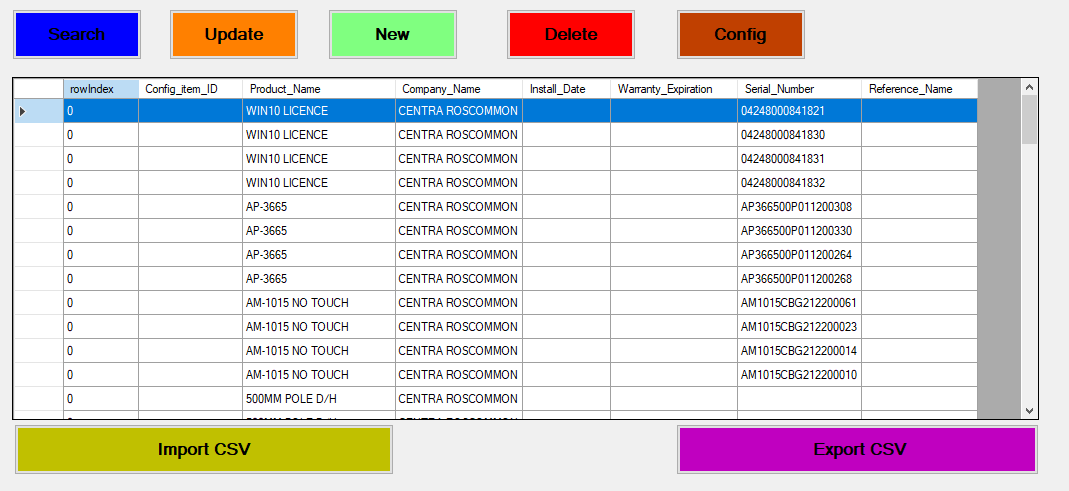
However, if the user populates the PasteBox and clicks ok,



A message will appear,

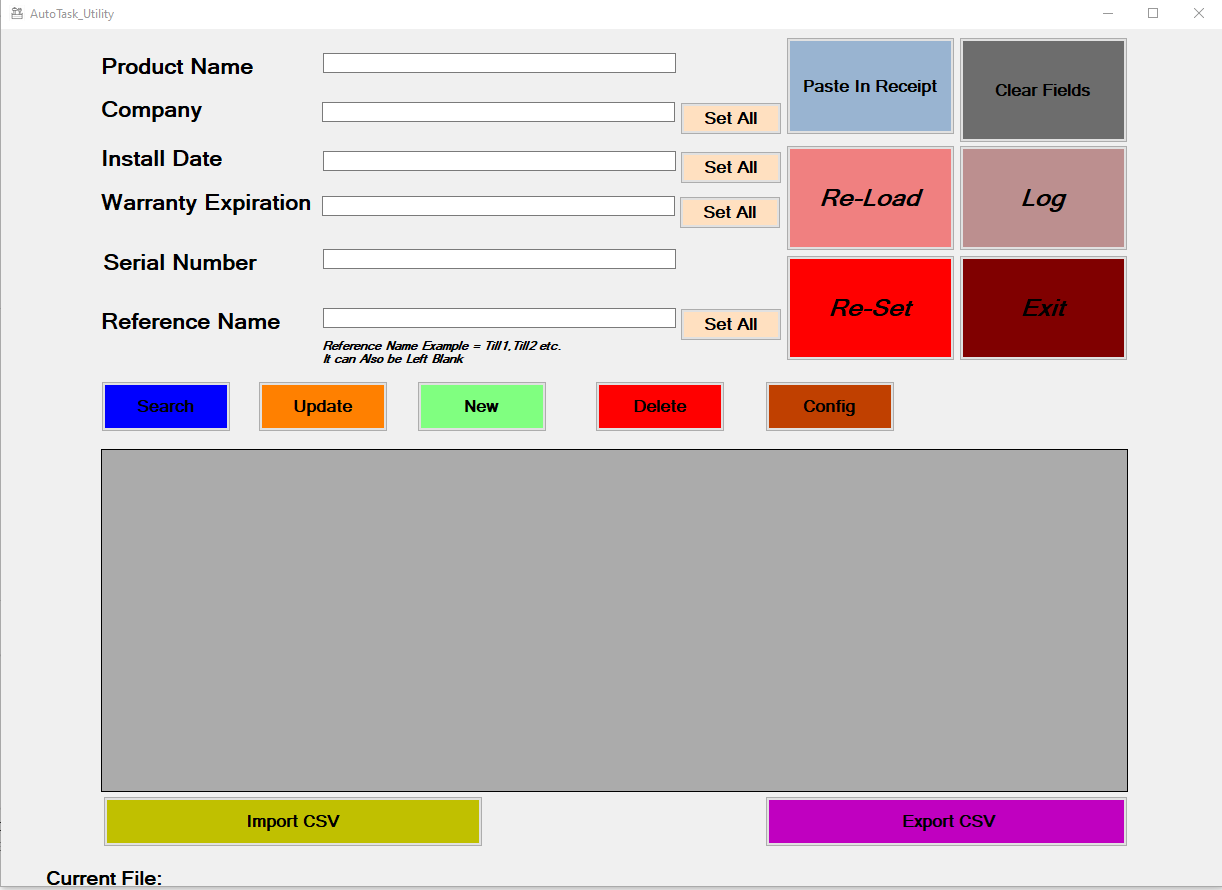


The PasteBox form will close, And the grid will populate in the main GUI.

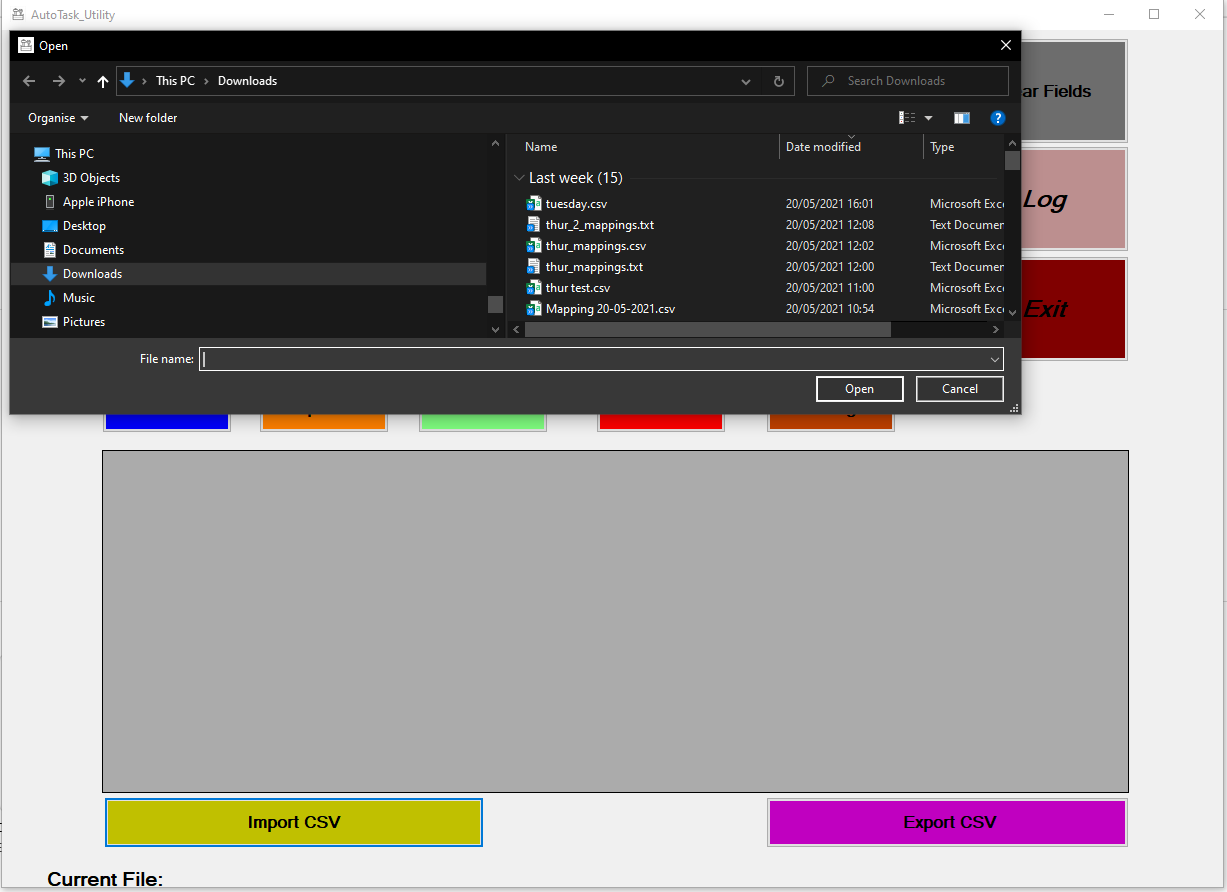


## 4.3 Import CSV

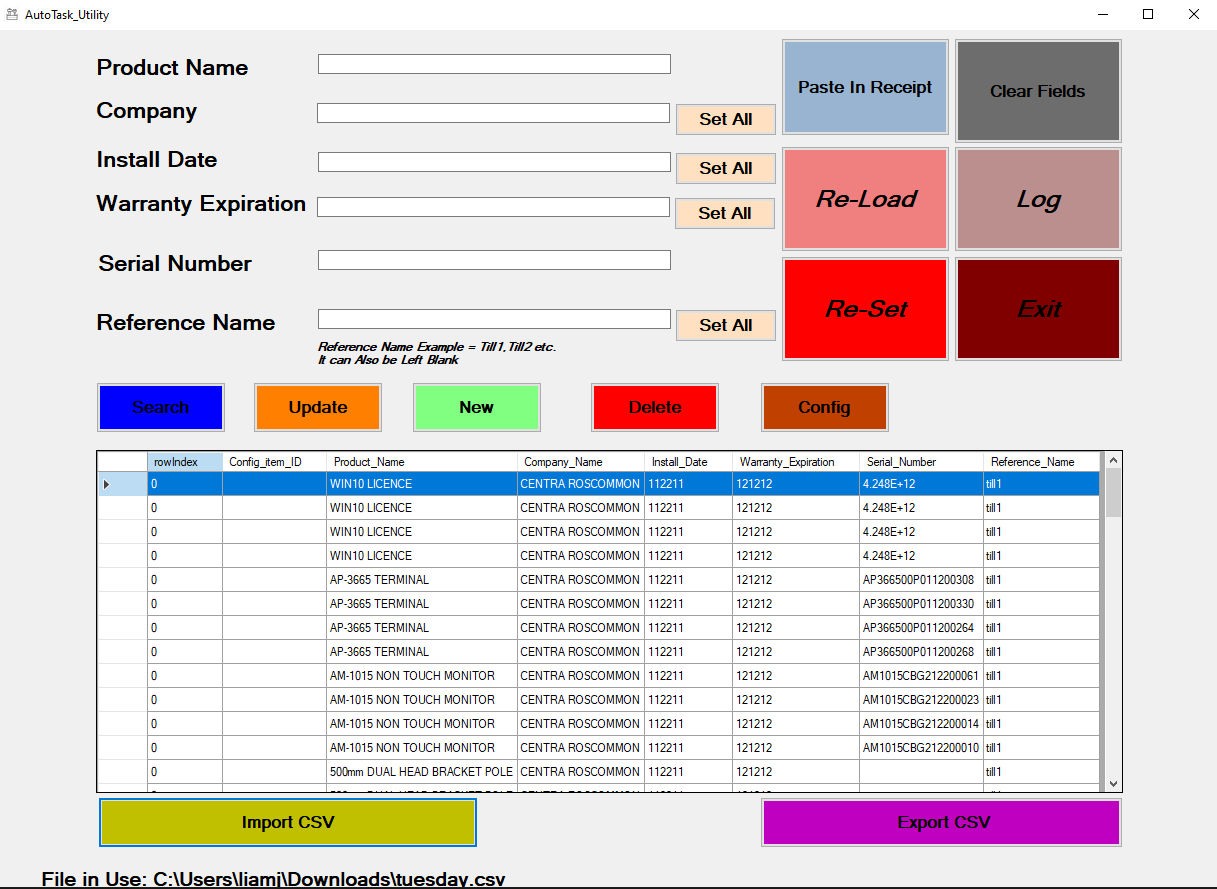
When the utility launches, the grid is empty. Users can import a CSV file using the Import CSV Button to populate the grid.



A file dialog window opens up,



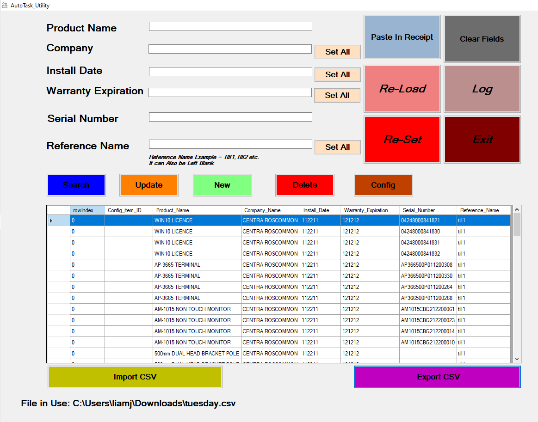
The user enters a name in the File Name textbox and clicks Open, and if the csv file is correctly formatted (see Appendix 1). The grid will populate accordingly in the main GUI, and the file in use label at the bottom of the screen will reflect the users current working file.



the file will populate the grid on the main GUI.

## 4.4 Update Function

The update function allows for the editing of an existing entry.



Grid

Figure 6 Update Button.

To use the update function:

* Users select from the grid (the grid refers to the DataGrid; the rows of entries containing the products) the entry they wish to update.
* Once they find the entry, they click on it.
* After the entry is clicked, it populates the textboxes.
* User modifies the entry and click the update button.
* The entry in the grid reflects the update.

## 4.5 Delete Function.

The delete Button allows users to delete an entry, therefore that entry will be will not be entered into the exported CSV.

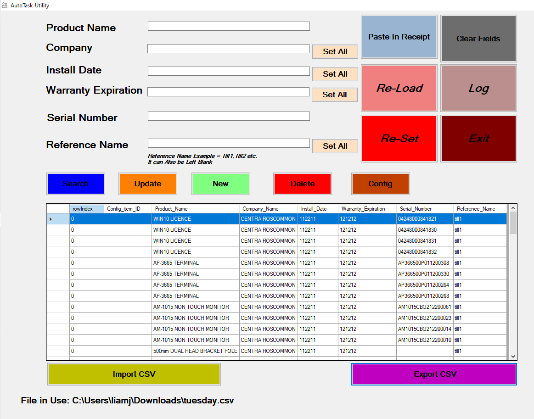


Figure 7 Delete Button.

* Users select the entry they wish to delete from the grid and click on it.
* After the entry is clicked, it populates the textboxes.
* User clicks the delete button.
* User clicks delete and entry is removed from the grid.

## 4.6 New Function

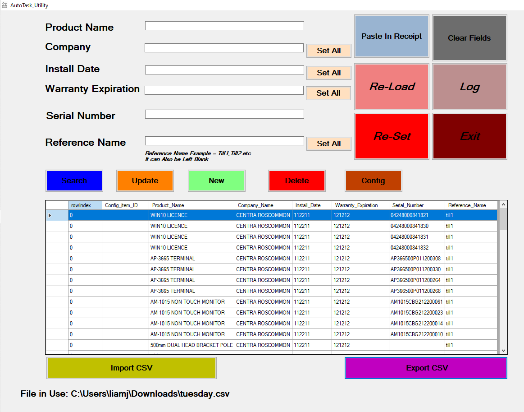


Figure 8 New Button

The new button allows users to create a new product and enter it into the grid. To do this, the user needs to.

* populate the necessary textboxes (i.e., Product Name. Company Etc.).
* click new.
* A new entry will appear in the grid.

## Set All Function

### 4.7.1 Set all Company.

The Set All function allows users to update the Company column on the grid.



Achieved by clecking the Set All button on the form.



### Set All Install Date

Similarly, the Set All Button allows users to update the Install date column in the grid..



This is achieved by clicking the Set All button adjacent to the Install Button textbox.

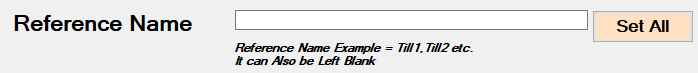


### 4.7.3 Set All Reference Name.

The Set All button allows users to update the reference name column in the grid.



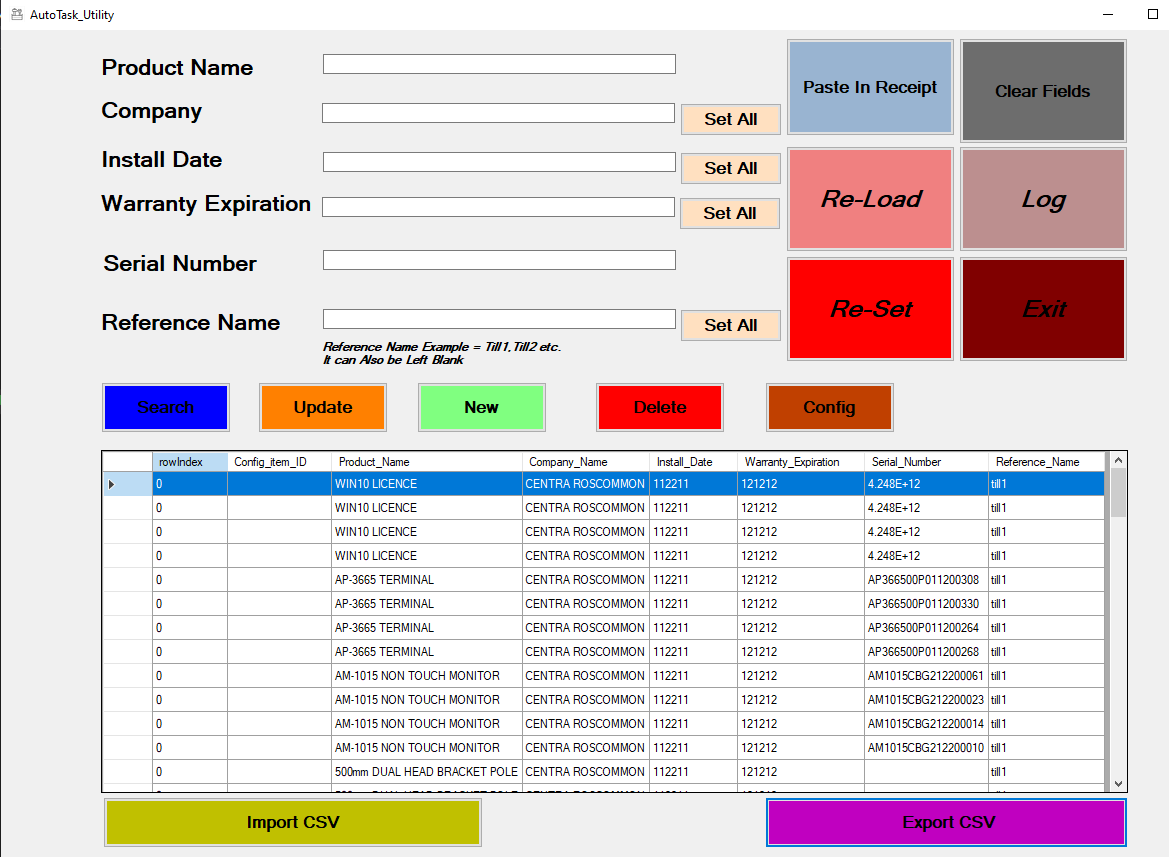
This done by clicking the Set All button next to the Reference Name Textbox.



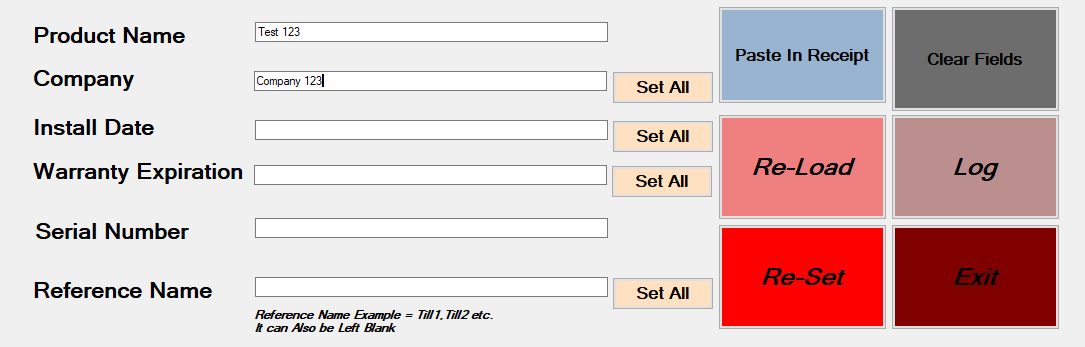
Note the reference name is not required, and its value is entirely down to user discretion.

## Clear fields Function.

The Clear fields button allows users to empty all the values from the textboxes.

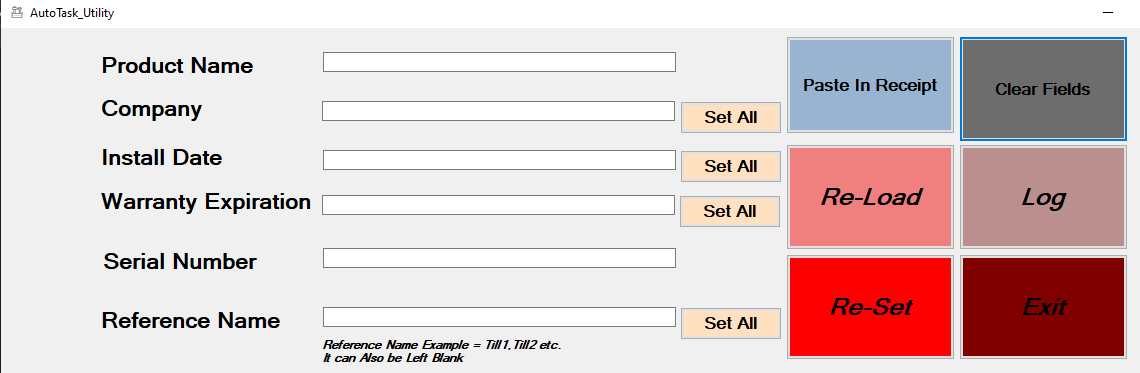


For example, if the user began to populate the text boxes for a new entry,



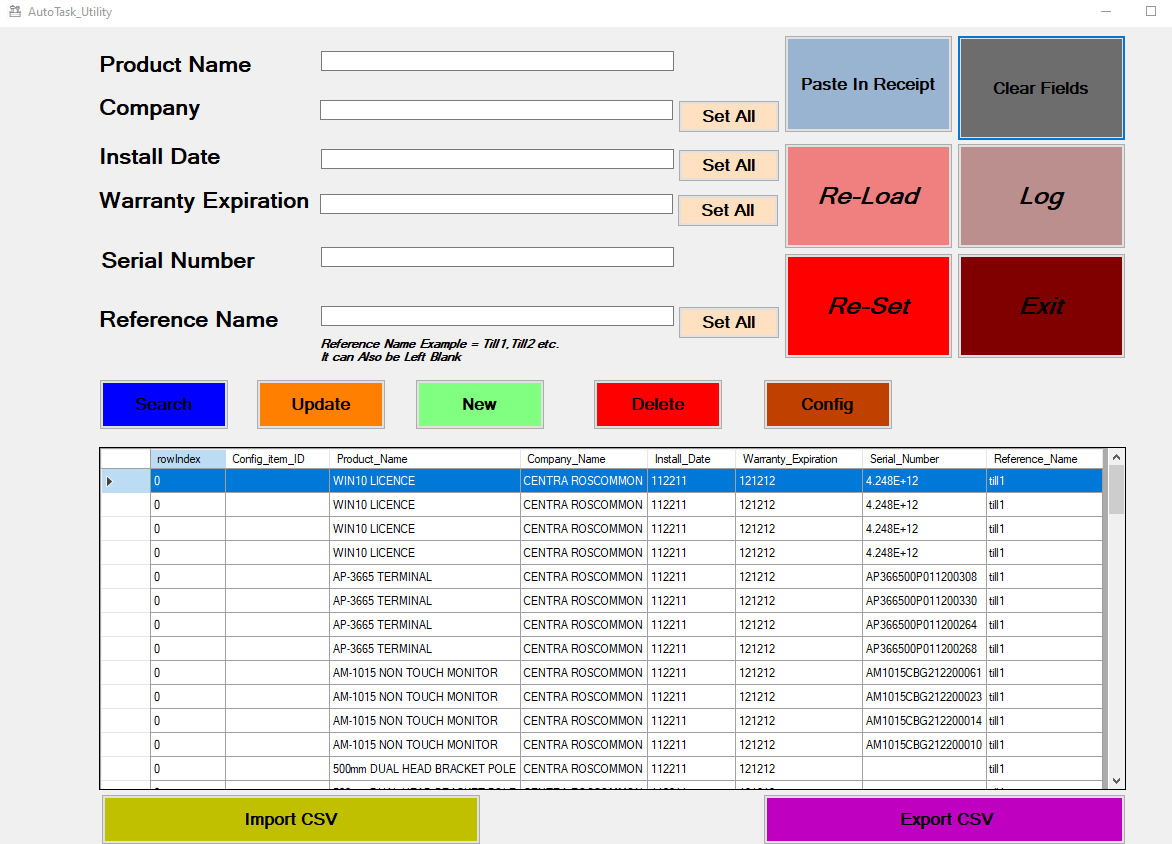
And realised part the way through it was in error and wanted to clear out the entries, they

Would click the Clear Fields button.

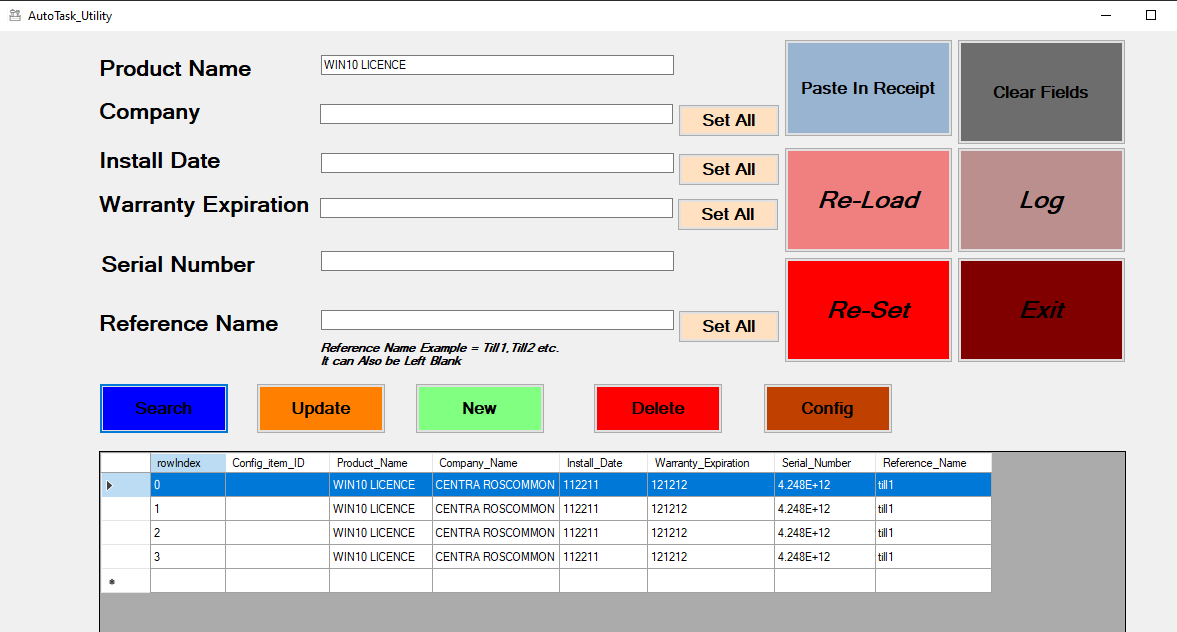


## 4.9 Reload Function

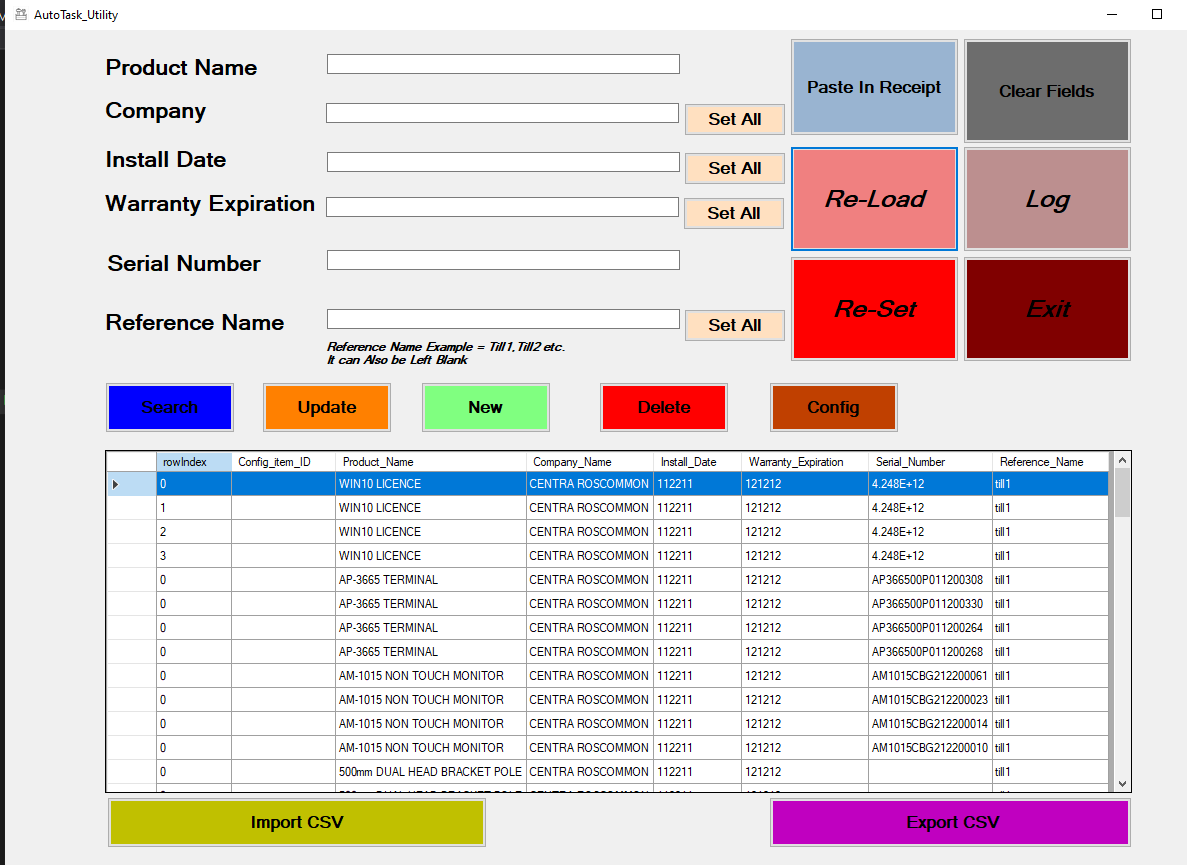
The re-load Button,



Is used when for example a user carries out a search function (See section on Search Function for full details). When the user carries out a search, the grid is populated with items from that search. Here we can see a product Name search and the grid populating accordingly.

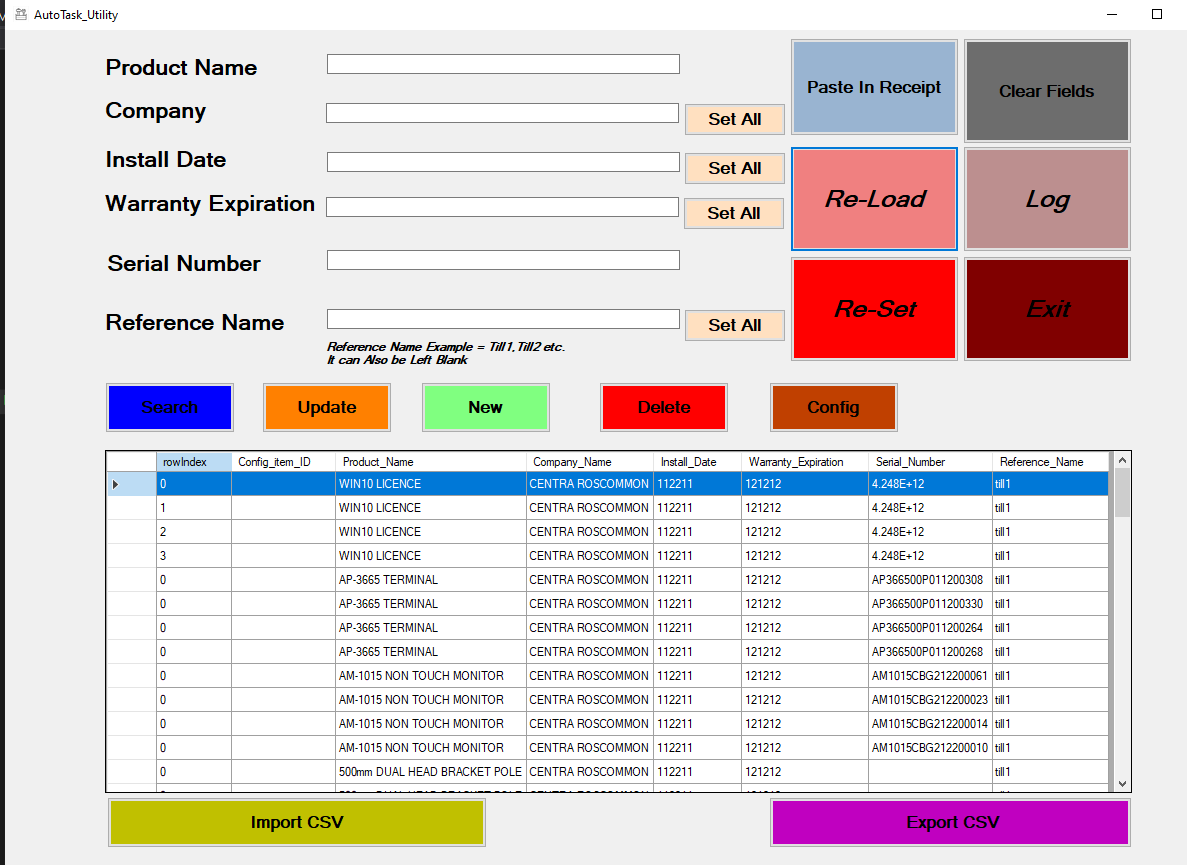


Once the user is done with the search, they would click the Re-Load button, this reloads the ‘full’ version of the csv file back to its unfiltered state.

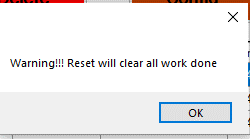


## 4.10 Reset Function.

The Re-Set button,



Allows users to reset the grid (zero entries) and reset the utilities memory. Useful when moving on to a new project/company entry. A warning pops up when this button is clocked,



Clicking okay will result in an empty utility.

## 4.11 Exit Function

The Exit button closes the utility down completely.



## Config Function

The Config button (Figure 9 Config Button), allows users to perform a mapping of the product name from what is stored in the receipt against what AT expects.

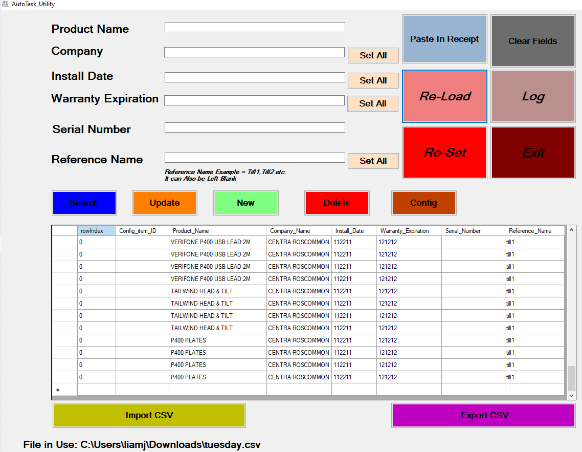


Figure 9 Config Button

### 4.12.1 Mapping Config Form

After the config button is clicked, the user will be presented with the *Mapping Config Form.*

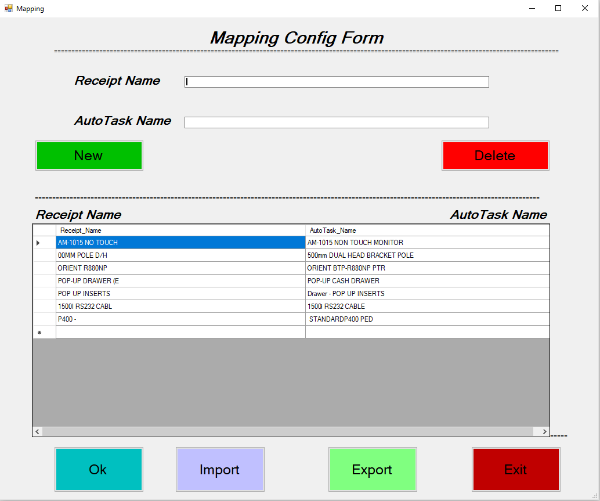


Figure 10 Mapping Config Form

When the utility is converting the receipt into the CSV, it will perform a quick look up of the mappings file (CSV file stored locally), see if the description has a different name in AT, if it does it will replace it with the AT name.

This CSV file will exist to check/save/edit all mappings, as mentioned it will live alongside the utility, and is represented visually on the form in the grid, with the *Receipt* name on the left and the *AT name* on the right (Figure 11 Mapping Grid.). The *Mapping Config Form* is a means of editing this CSV file.

When the utility is converting the receipt into the CSV, it will perform a quick look up of the mappings file, see if the description has a different name in AT, if it does it will replace it with the AT name.

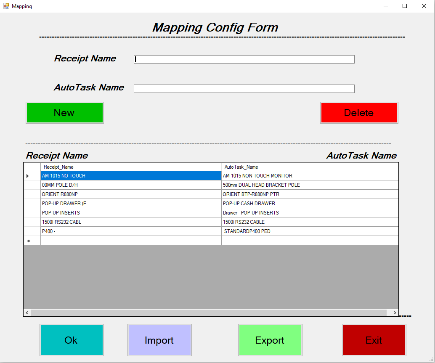


Figure 11 Mapping Grid.

### 4.12.2 Mapping Config Form Error

Note: If there is an error in the mapping CSV file, the most common being a duplicate entry, the utility will show a message box alerting the user. In the example illustrated below we can see that the last 2 entries in the grid are identical.

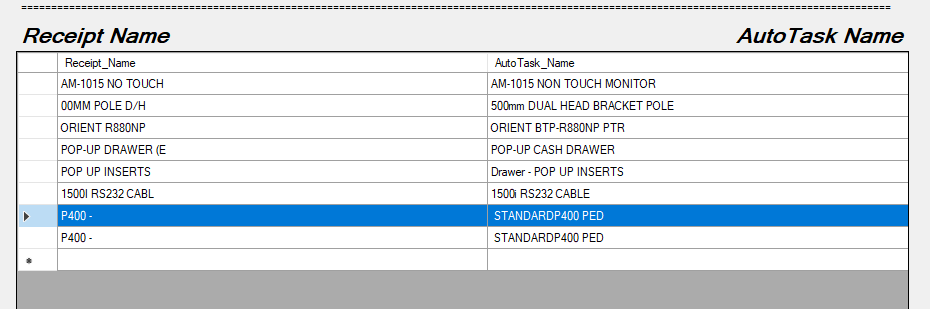
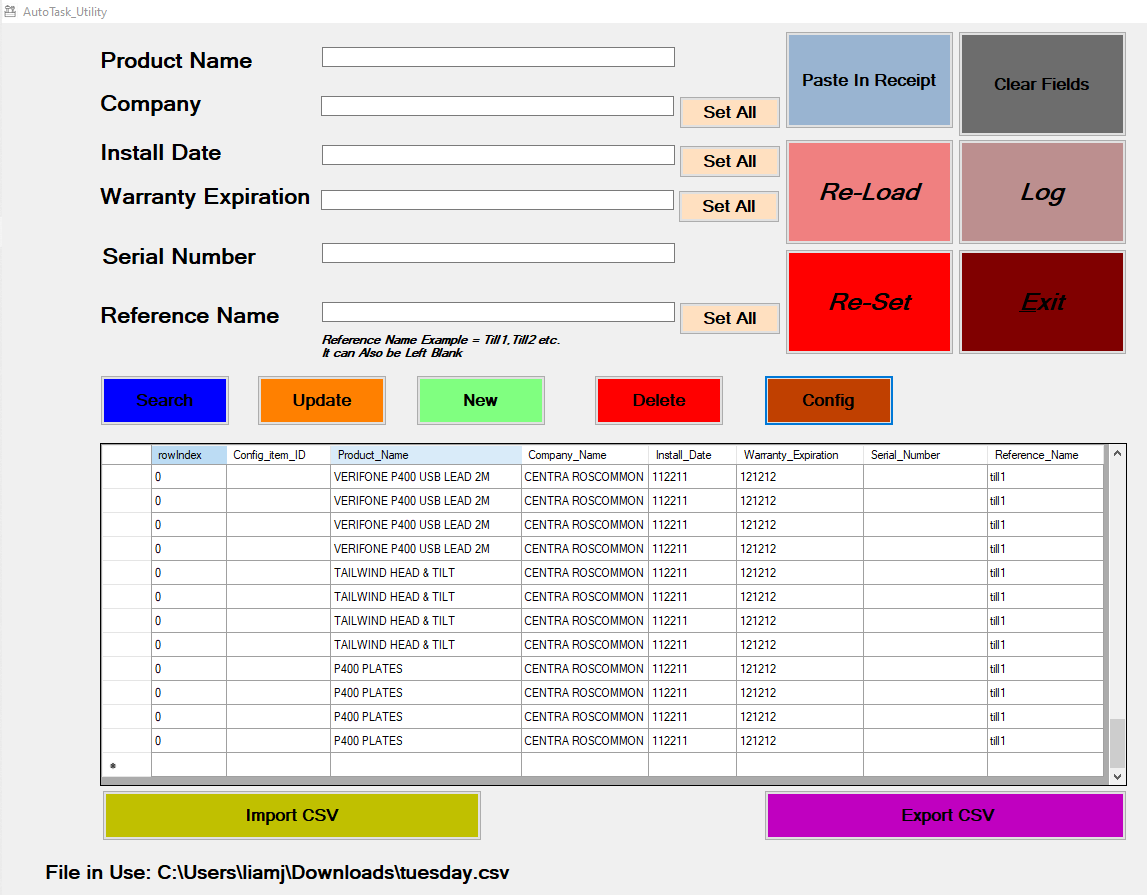
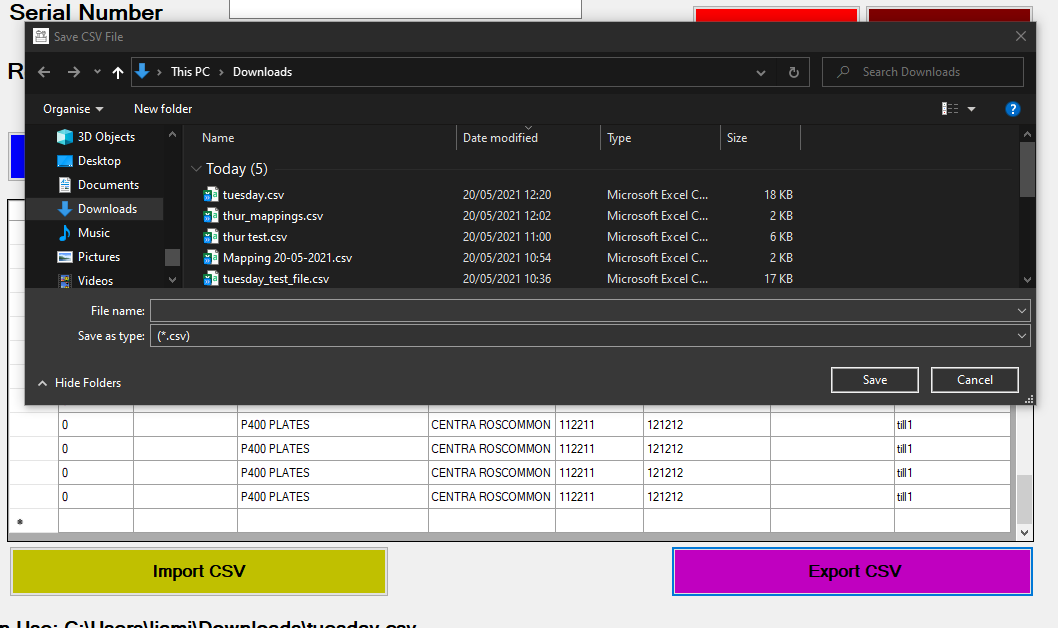


Figure 12 Config Mapping Duplicate Entry.

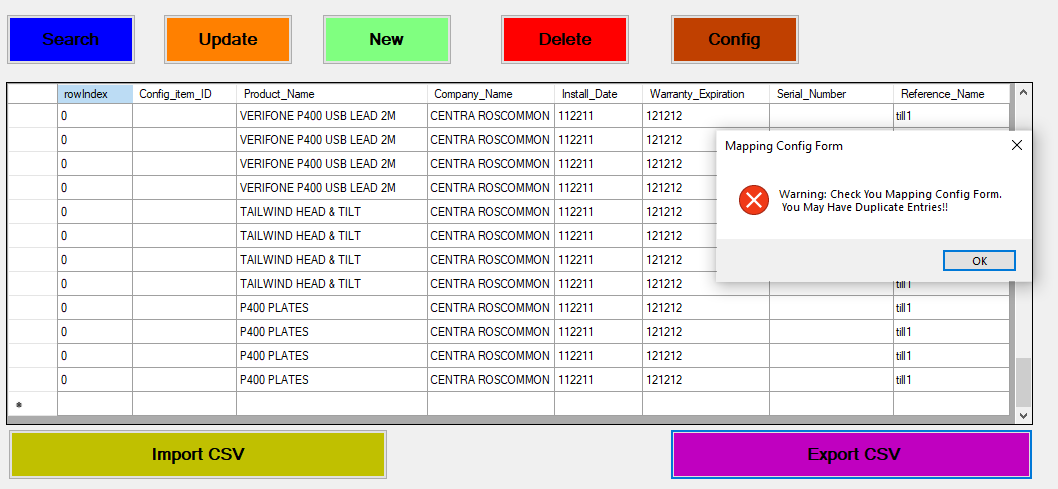
When the user goes to export the CSV in the main GUI by clicking the Export CSV Button,



The file save dialog will appear,



When the user enters a filename and location and clicks save, A warning message will appear. Alert them to check the Mapping Config Form Grid for possible duplicates.

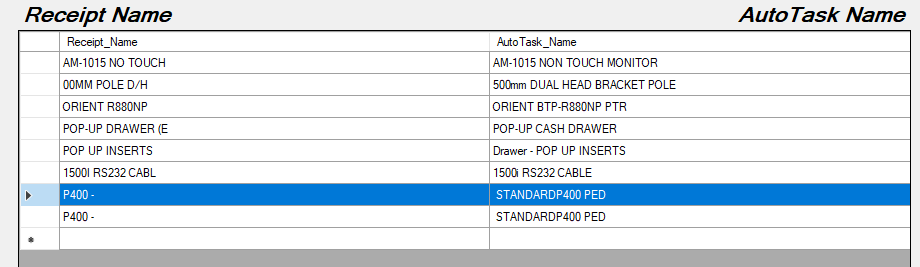


By returning to the Mapping Config Form grid and deleting the duplicate entry, it will fix this error.

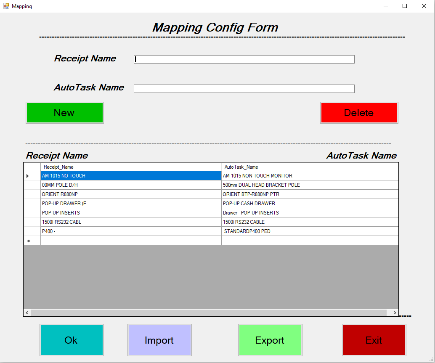
(see the next section for the Delete Button Function).

### 4.12.3 Mapping Config Form Delete Button

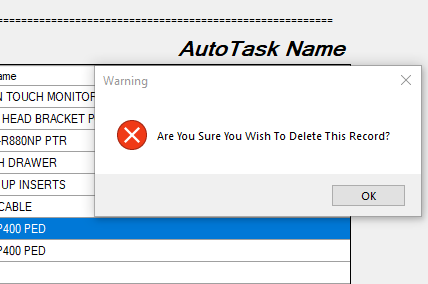
In the previous example we have seen how it may be necessary to delete an entry from the Mapping Config Form grid, due to duplicate entries in this case. To perform the deletion the user would select the entry this wish to delete (the entire row will be highlighted in blue)



The user then clicks the Delete Button,



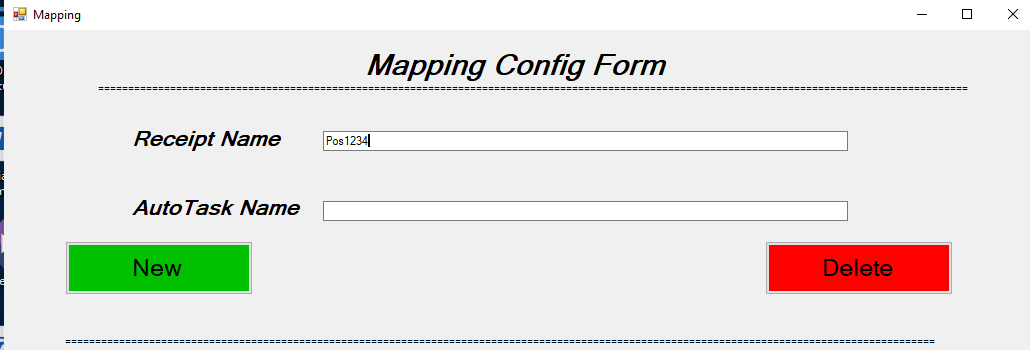
A waring will pop up asking for confirmation of this deletion,



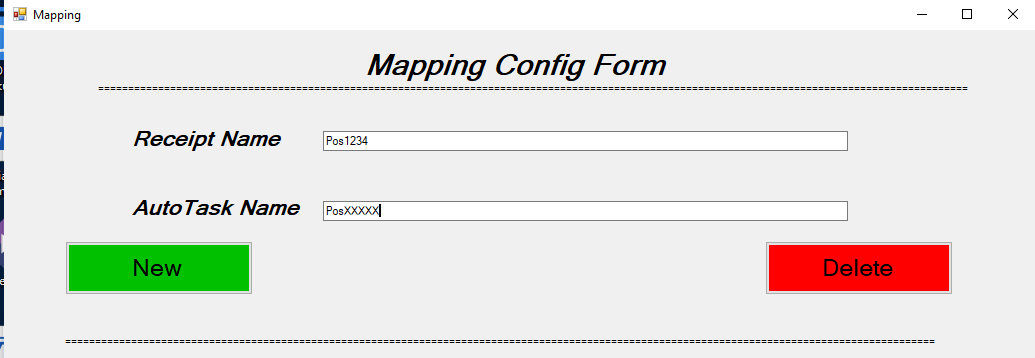
Users clicks ok and the record will be deleted from the grid.

### 4.12.4 Mapping Config Form New Button

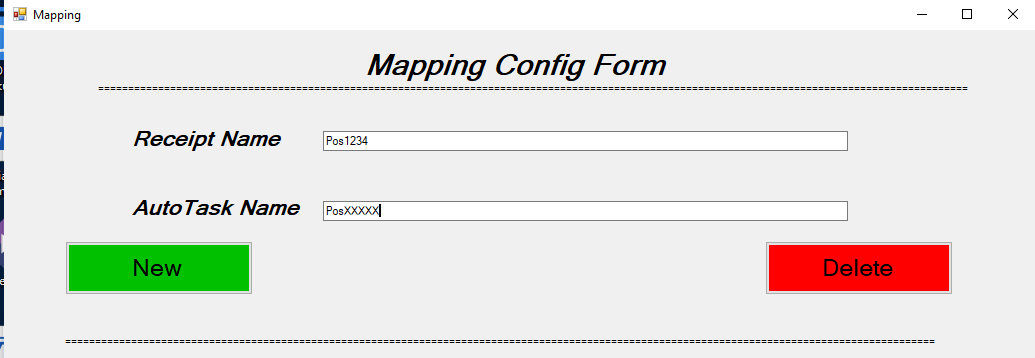
The New Button allows Users to create a new entry in to the grid, and ultimately store this new mapping in the system (see the Export Button section). To create a new entry the user enters the receipt name in the corresponding textbox,



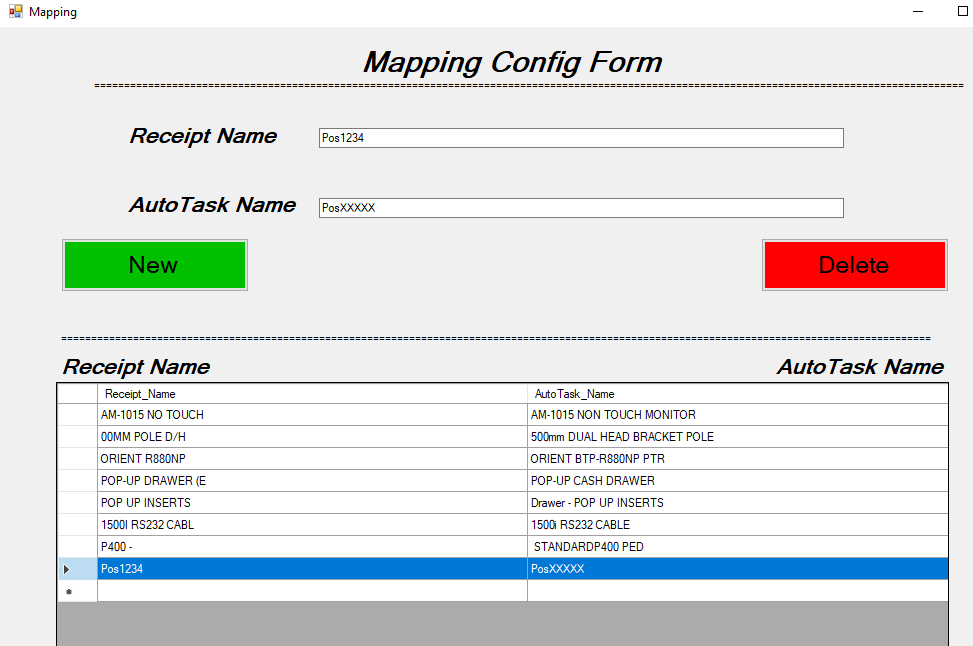
And they then enter the AT name in the corresponding textbox,



Then the user clicks the New Button,



This will then populate the grid below with the new entry (highlighted in blue below).



### 4.12.5 Mapping Config Form Import Function

The form has an import function (Figure 13 Import Function):

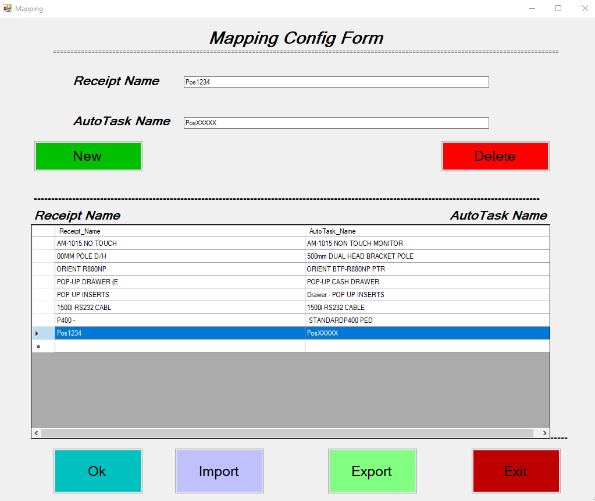
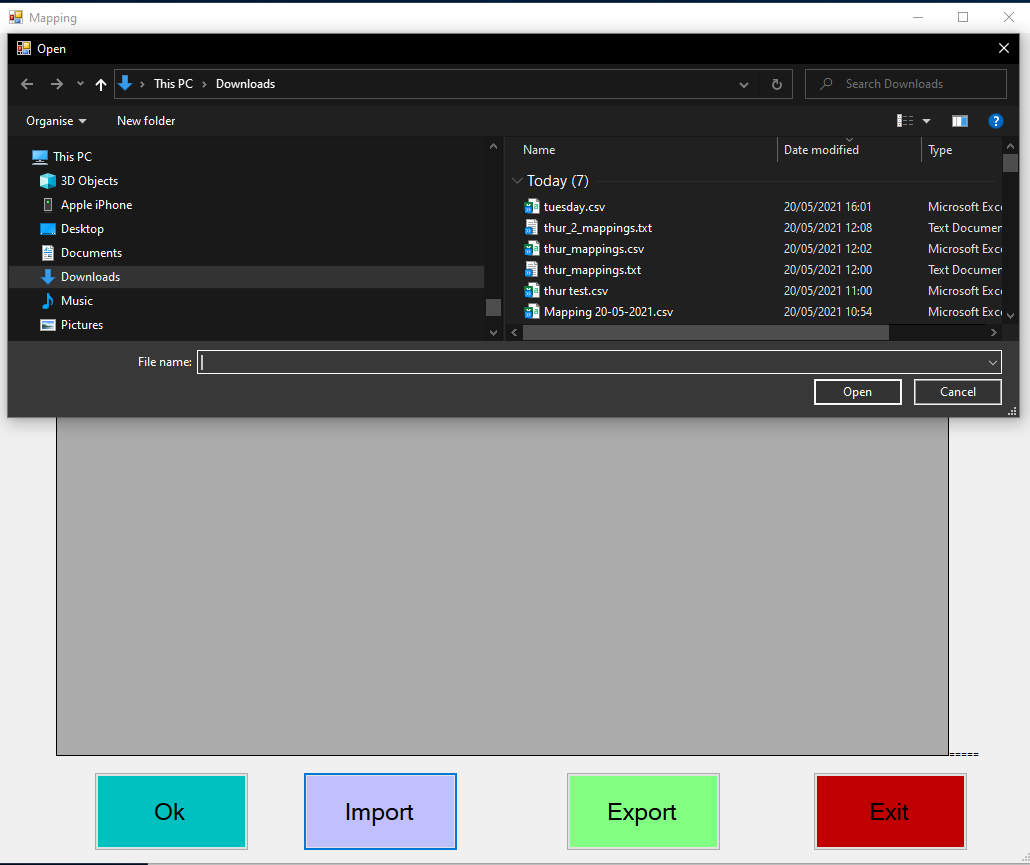


Figure 13 Import Function

When the user clicks on the Import button a File dialog appears prompting them to choose a file for import (Note: the file must be in CSV format).



The user would click Open in the File Dialog and the grid in the Mapping Config Form will populate. An example of what a pre-formatted CSV file would look like is shown below (Figure 14 Example of Mapping File to Import). Please note:

* It will have 2 entries per line.
* The entries must be separated with a comma.
* The left-hand side entry will be the receipt name.
* The right-hand side entry will be the AT name.
* If the format is not correct, it will not load into the grid.

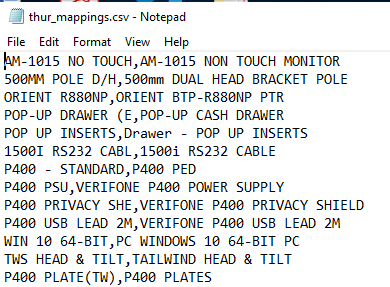


Figure 14 Example of Mapping File to Import

Creating a file to import can be handled by the utilities Export Button (See next Section). By creating the mapping file in this fashion, it will guarantee the format is correct for Import.

### 4.12.6 Mapping Config Form Export Function.

The purpose of the Export Function (Figure 15 Mapping Config Form Export Button) is for the user to export the mappings they are currently working on so that they can be saved in a different location, backed up to an external location or shared with another user. For this function to work, the user must have existing entries loaded in the grid (see previous section on how to import a mapping fille). The utility will load the last mappings used automatically.

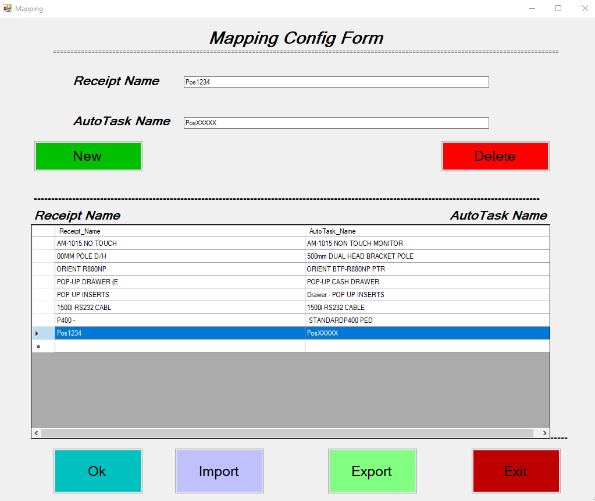
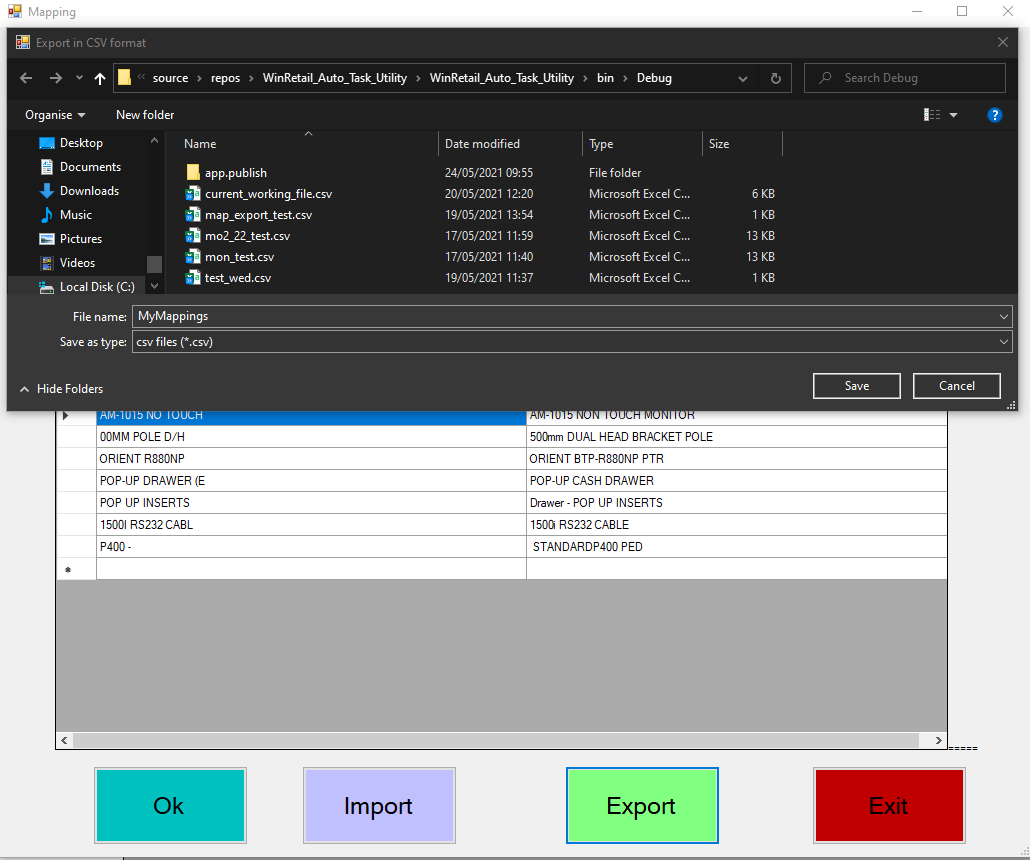


Figure 15 Mapping Config Form Export Button

To use the export function, the user would, provided the grid is populated:

* Click the Export Button (Figure 15 Mapping Config Form Export Button).
* A file dialog will appear.



* The user enters the name they wish to call the file in the File Name textbox.
* User clicks the File Dialog Save button.

### 4.12.7 Mapping Config Form OK Function

The Ok button plays an import role. Once a new mapping is done, or any other updating/editing/deleting of an entry in the grid displayed on screen, once the Ok button is clicked, it updates the grid to the Utilities memory and the Mappings file stored locally with the utility. NB: The user should always click Ok when they are happy with the contents of the grid (Figure 16 Mapping Config Form OK Button).

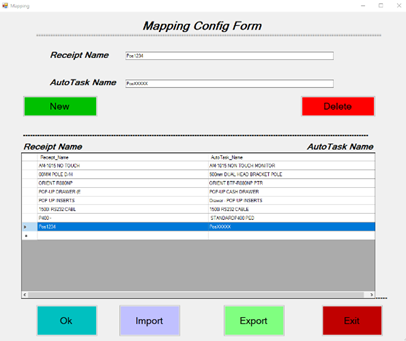
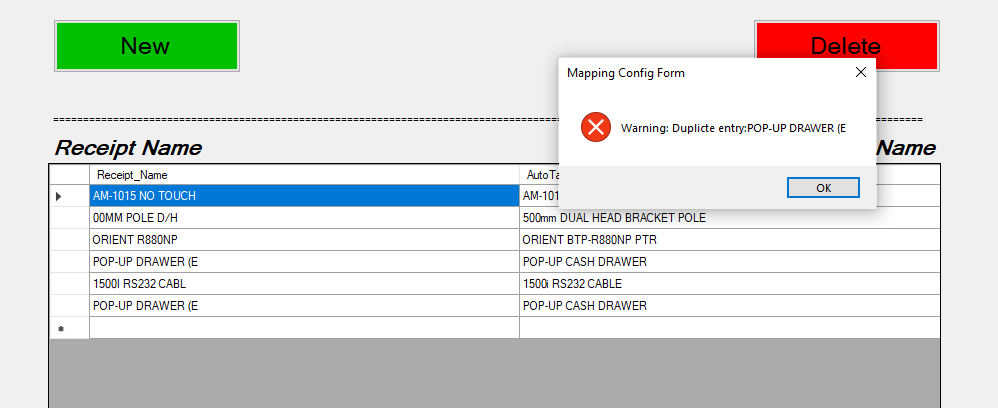


Figure 16 Mapping Config Form OK Button

Note: If there are any duplicates mappings in the grid/system memory, they will be handled when the user clicks ok. An error message will display if a duplicate is detected.



In the diagram above we can see that POP-UP DRAWER (E has been entered twice, the message box alerts the user to this error and also prints the duplicate item.

To handle this error the user needs to

* Take note of the duplicate.
* Click the ok button on the warning message.
* The Mapping Config form re-loads
* The user locates duplicate entry.
* User clicks the Delete button.

### 4.12.8 Mapping Config Form Exit Function.

The Exit Button closes down the Mapping Config form.

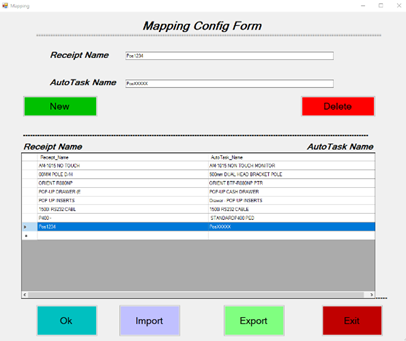
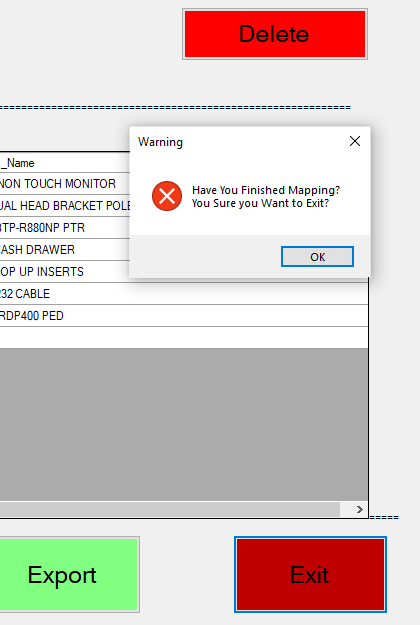


Figure 17 Mapping Config form Exit Button.

The utility will issue a waring in a message box asking the user if they are finished mapping and want to exit, the user should click ok if they are happy to exit.



## Search Function

Users can search the grid in the main GUI by clicking the Search button (Figure 18 Search Button).

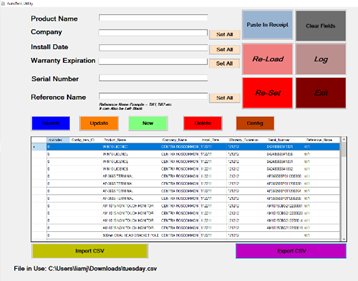
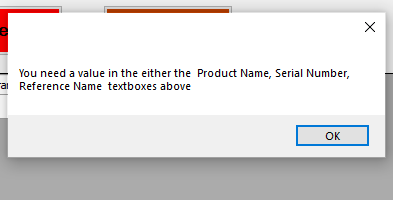


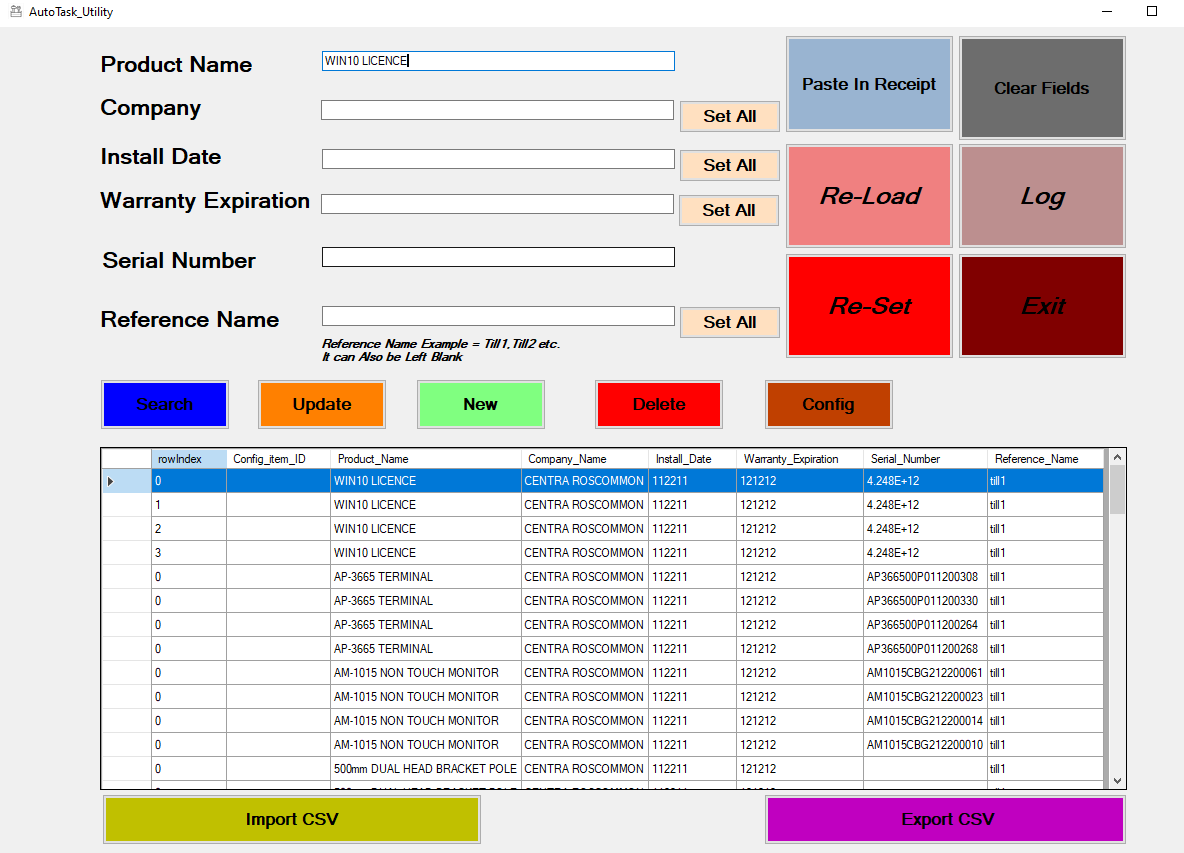
Figure 18 Search Button

To use the search function, the user would enter an existing Product Name/ Serial number/Reference Name into the associated text box and click the search button. The grid will populate with all found entries. NB: Entries must match what is in the grid. If a user enters a search term in any of the other textboxes, a warning message (shown below) will appear.

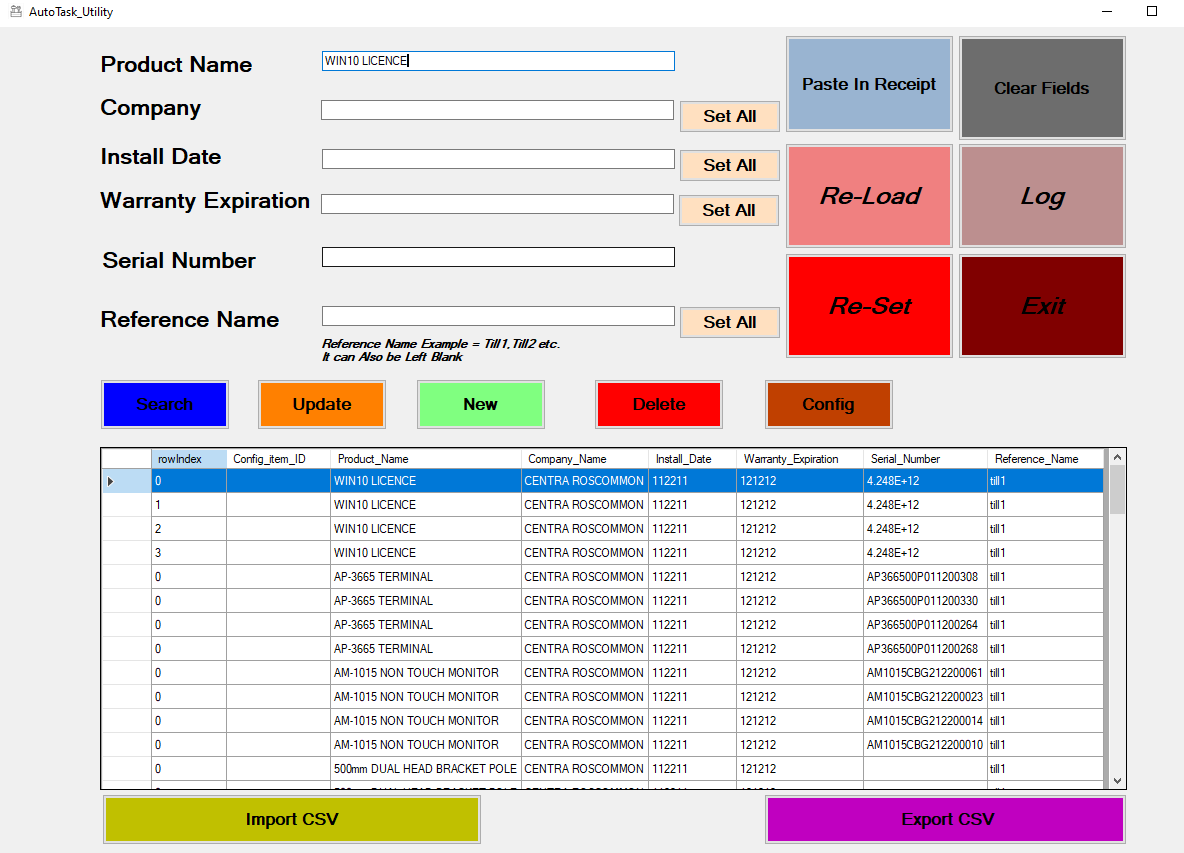


### 4.13.1 Search Via Product Name

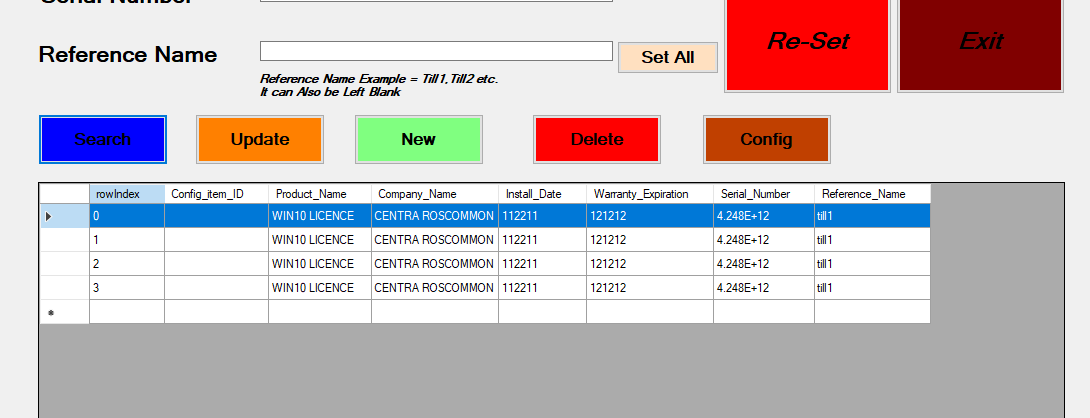
When the user wishes to search via product name, they enter the product name (WIN10 LICENCE in this example) in the associated text box,



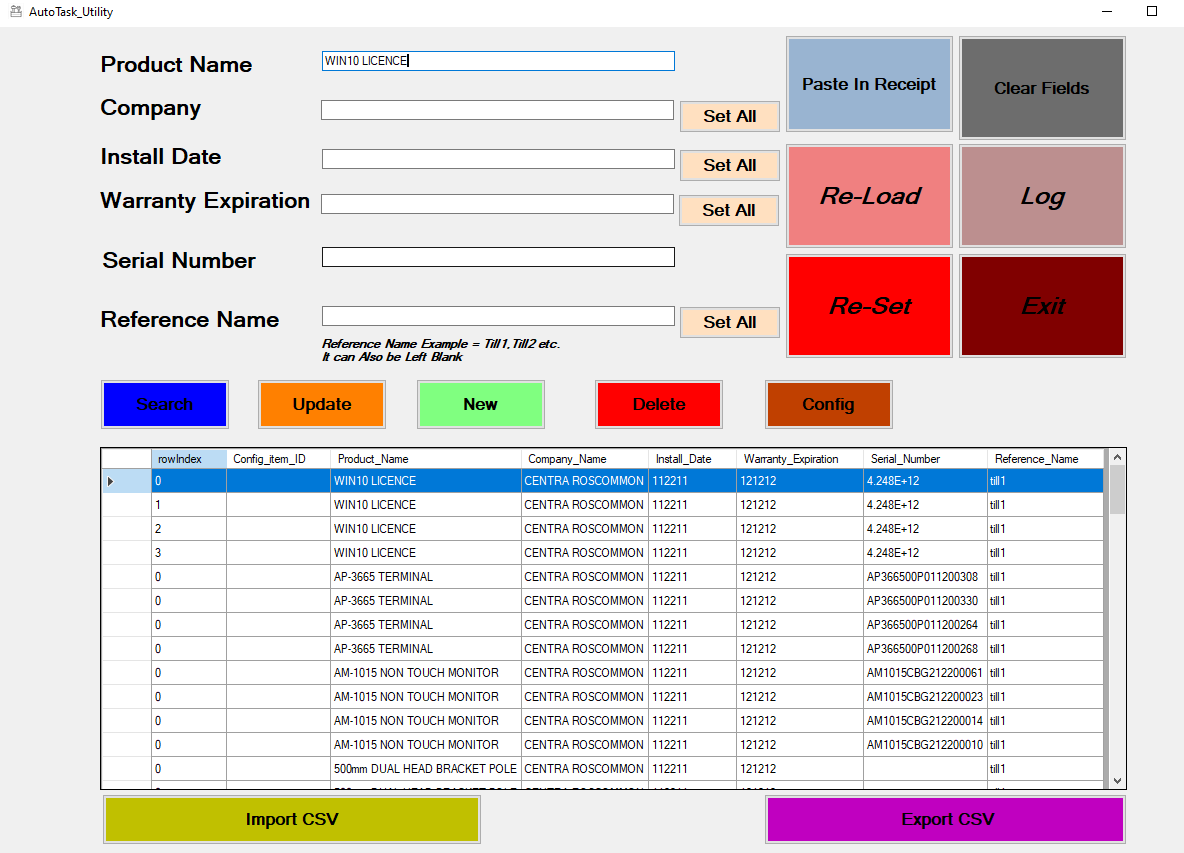
And click the Search Button,



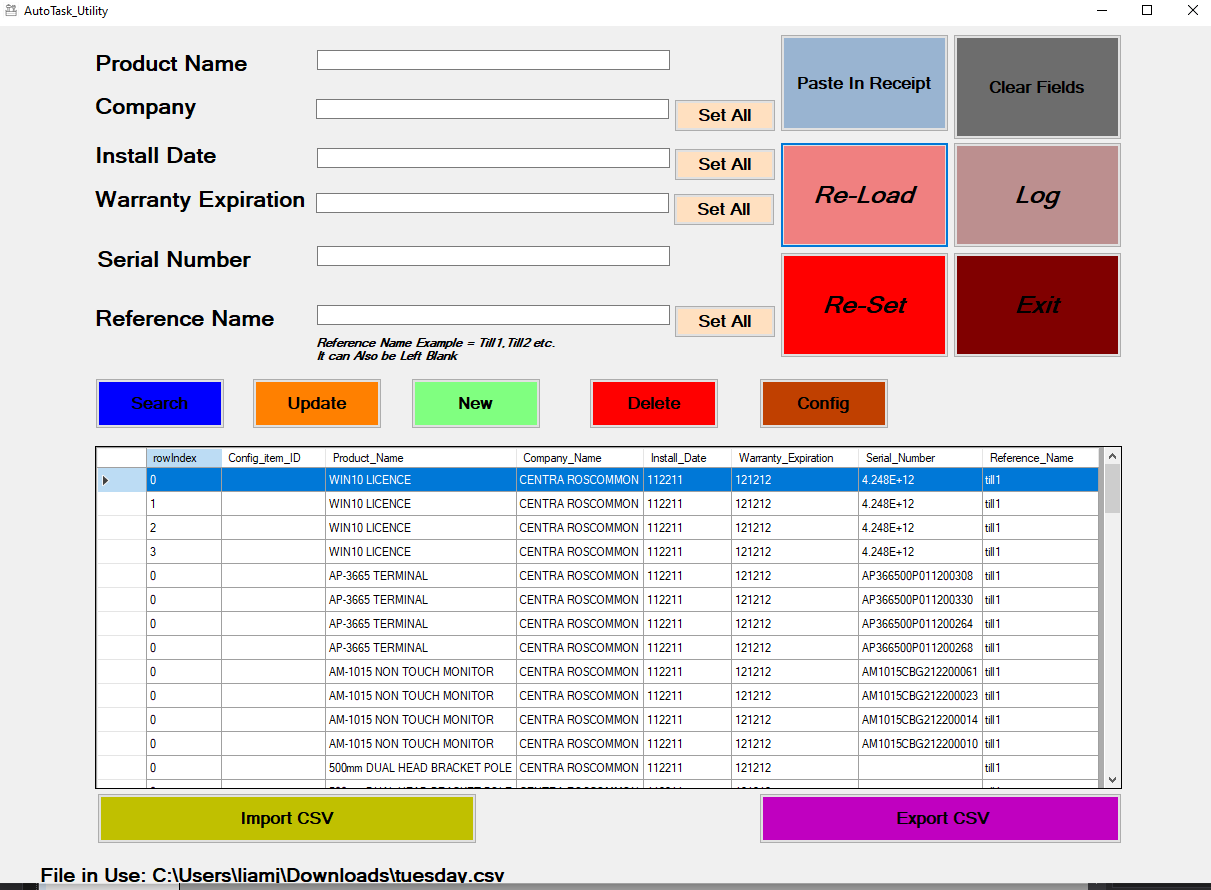
And the grid will populate with all entries found,



To exit the search, the user clicks the Re-Load Button,



Which clears the text boxes and repopulates the grid will all entries, essentially back to where the user was before the search began.



### 4.13.2 Search by Serial Number

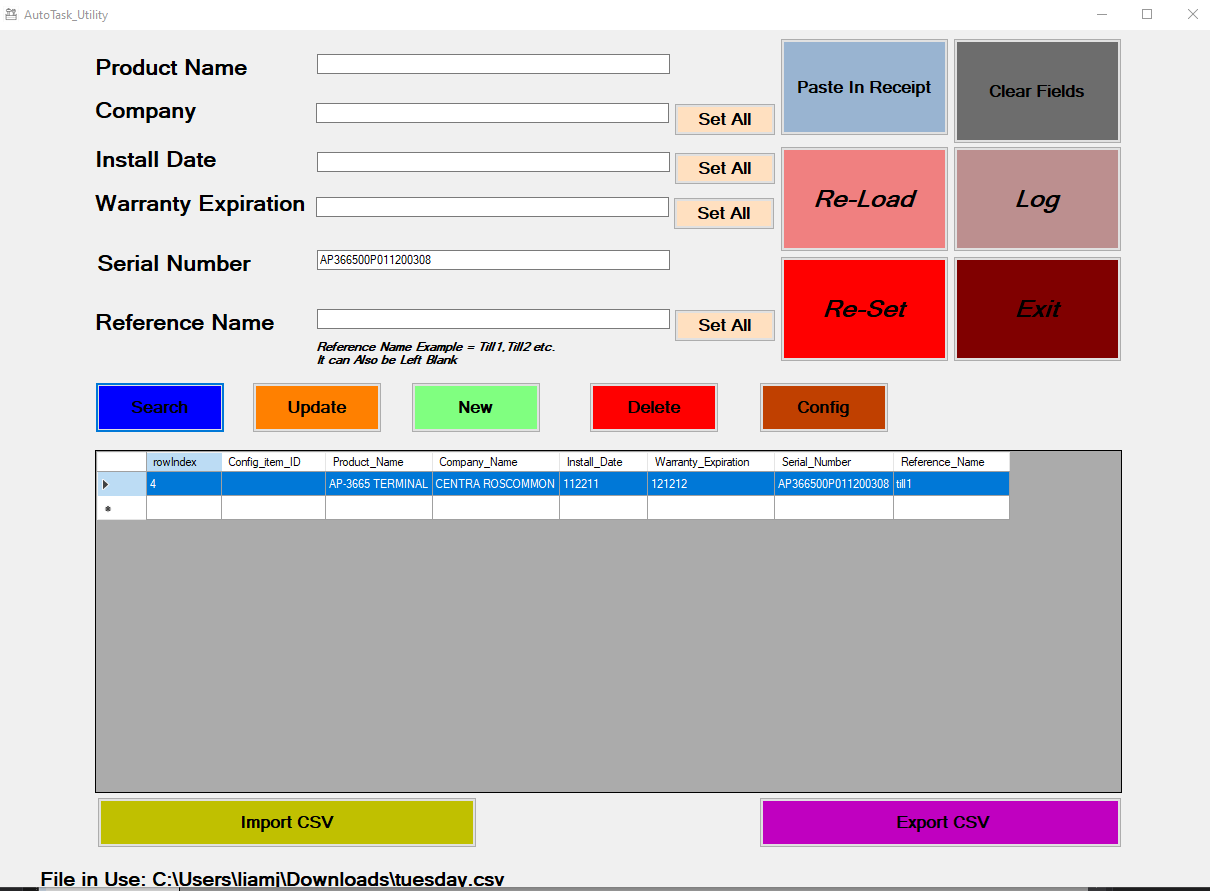
The search by serial number is the same process as above. The user enters a value in the Serial Number textbox (in this example AP366500P011200308) and clicks the Search Button.



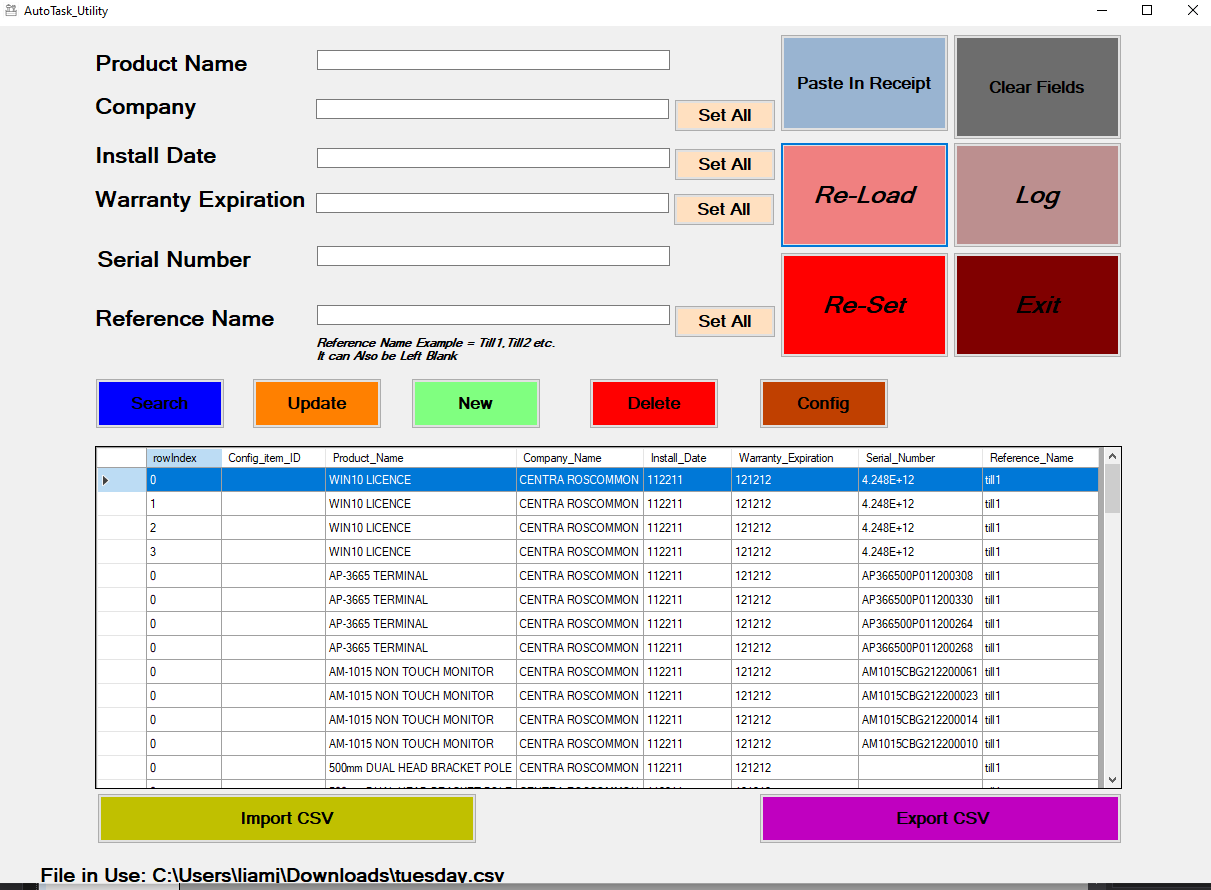
The grid will populate with the search results,



The user clicks the Re-Load button,

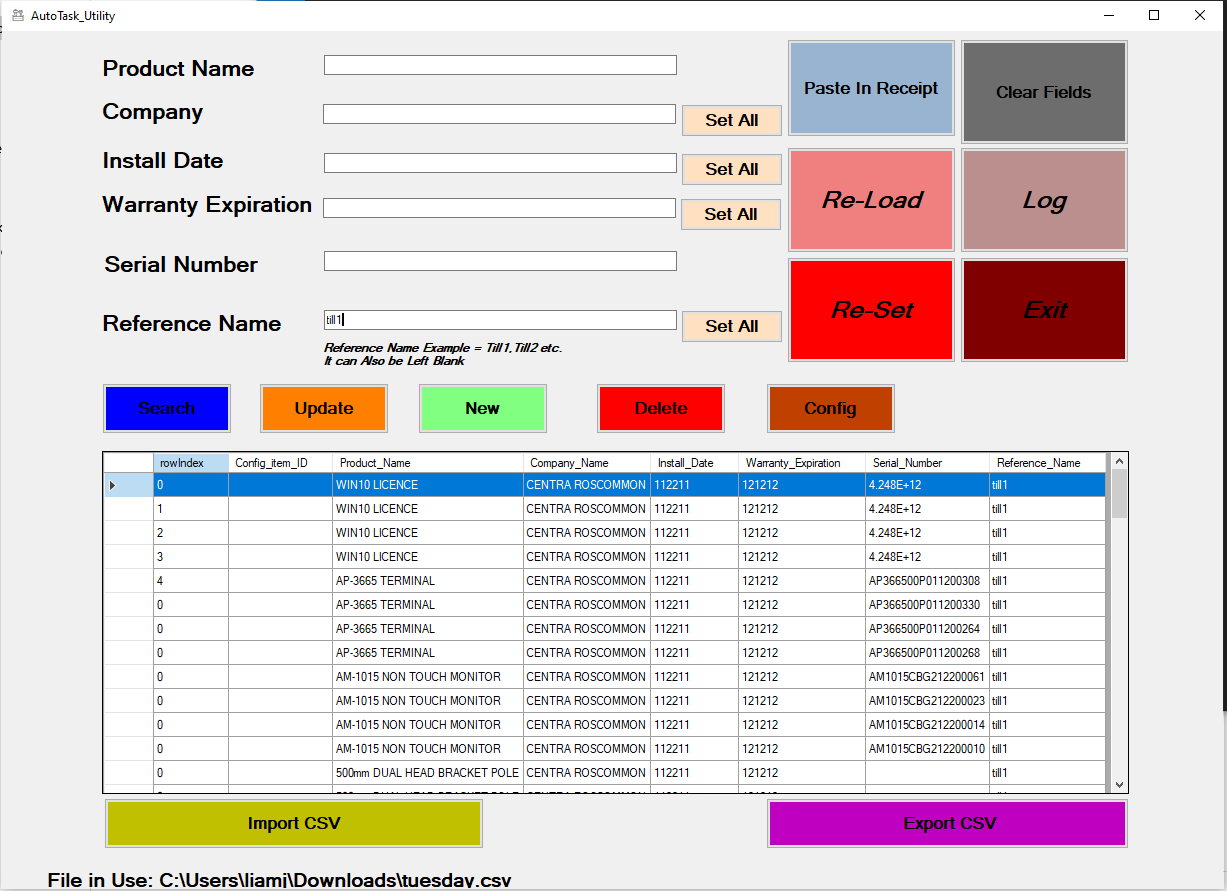


The textboxes clear the grid re-populates.

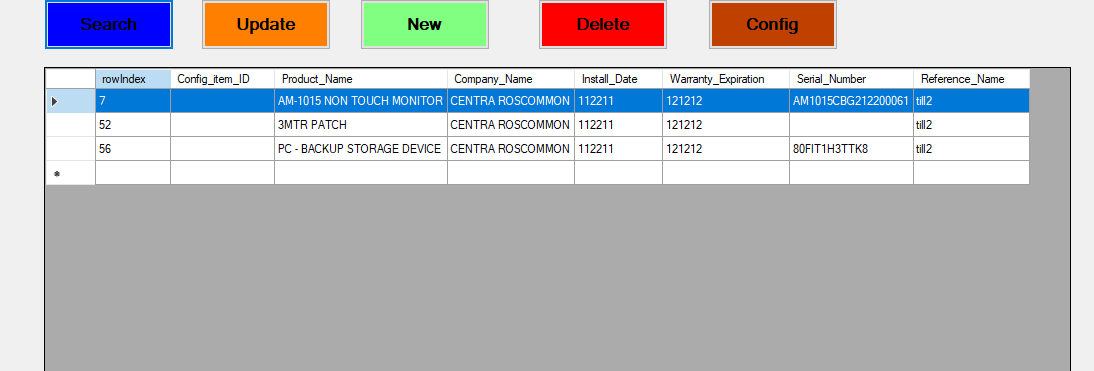


### 4.13.3 Search by Reference Name

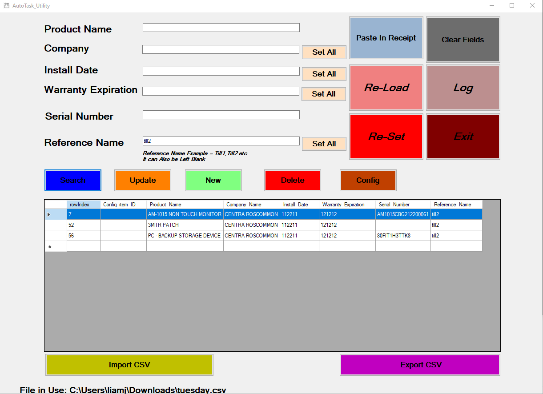
The utility can search by reference name by populating the relevant textbox (Till 2 used as an example value here), and the Search button is then clicked.



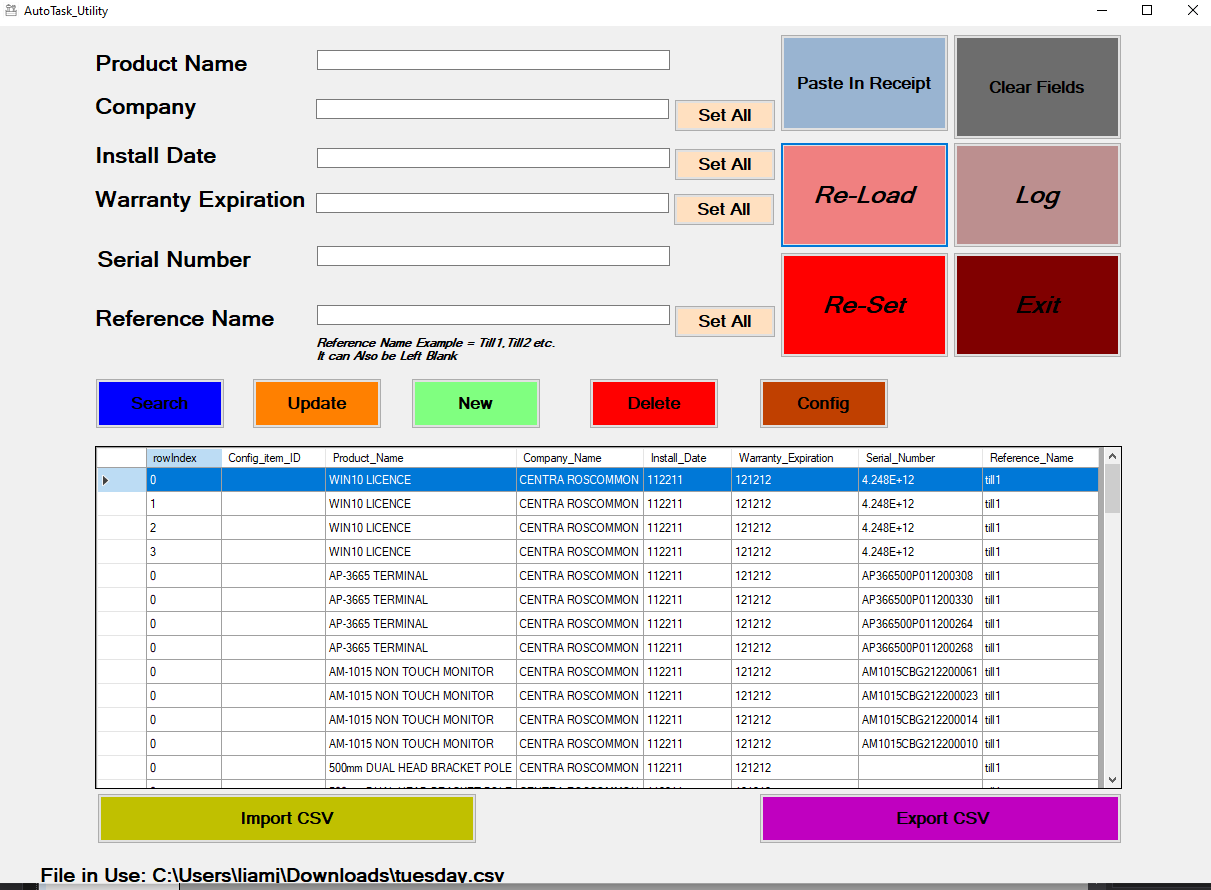
The grid will display the search results.



The user clicks the Re-Load button,



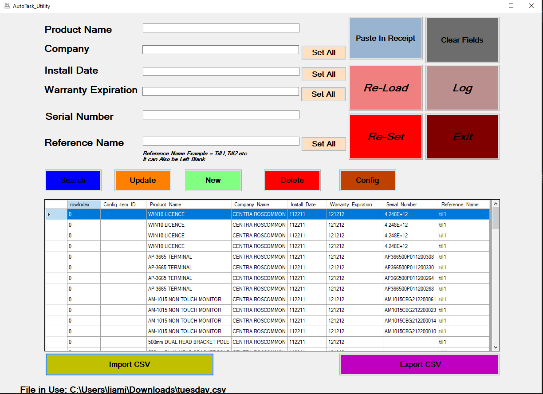
The textboxes clear the grid re-populates.



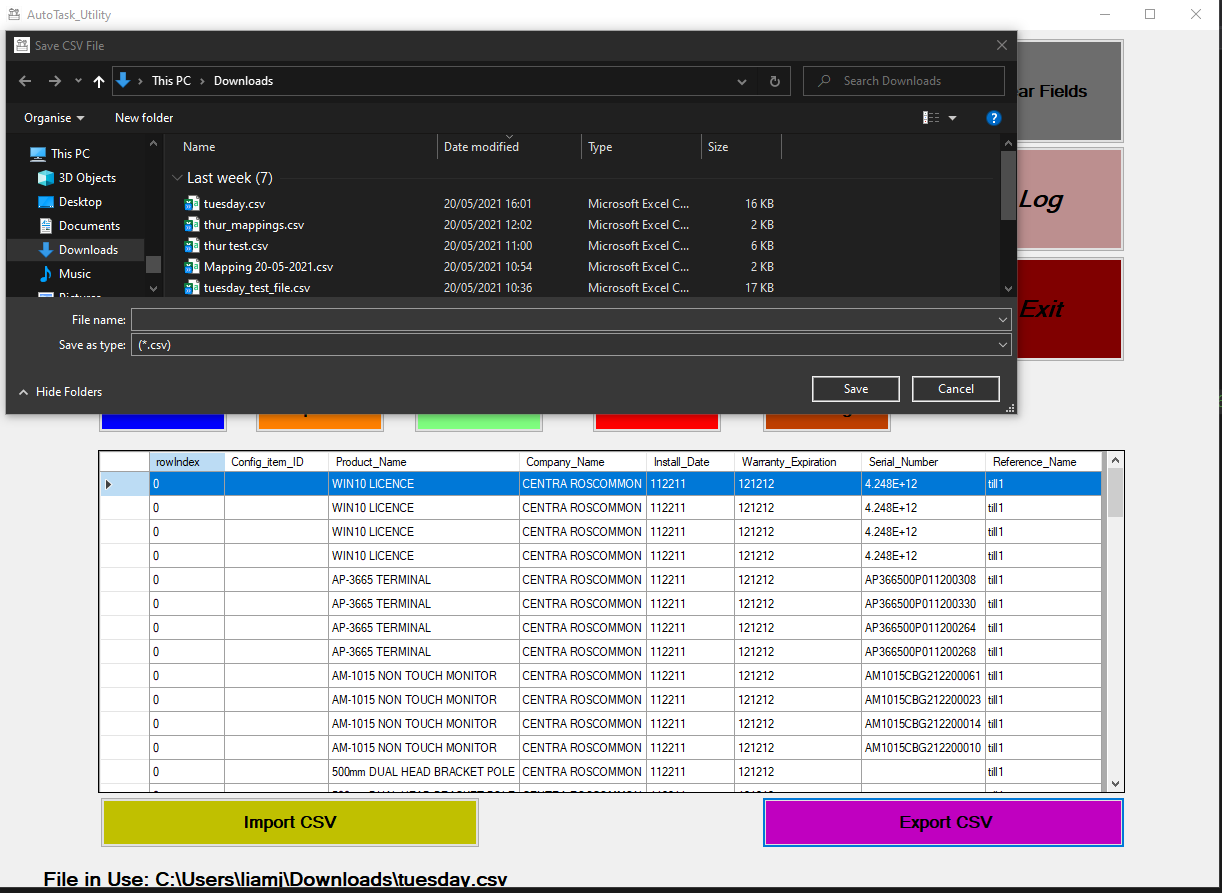
## 4.14 Export CSV

Once the user is happy with all of the updates, new entries, deletions, mappings etc.

They can then click the create *Export CSV* Button*.*



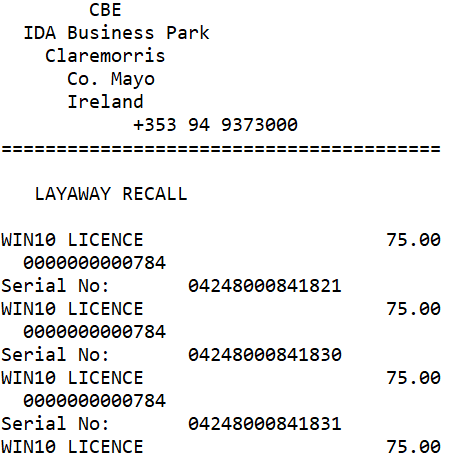
A file dialog window will pop up and the users chooses the name and location of where to save the file, followed by the save button.

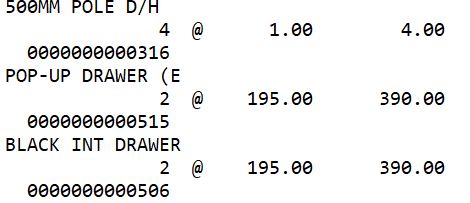


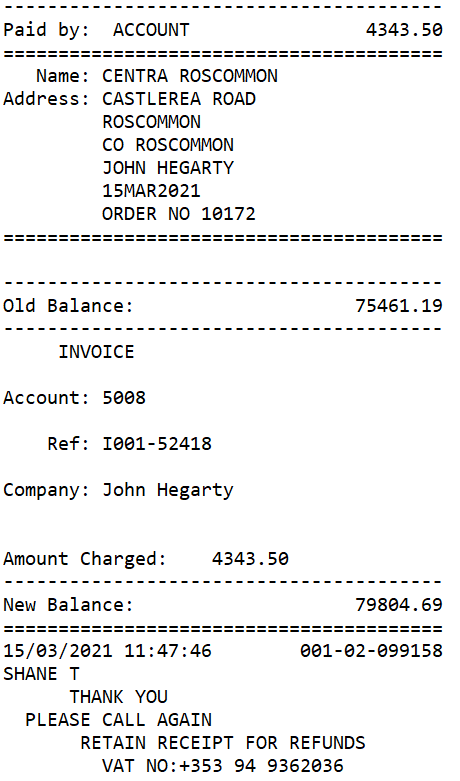
# 5 Receipt Parsing

## 5.1 Example of Receipt

Below is a snippet of a receipt taken from WinRetail Journals (cropped to fit here for demonstration purposes).



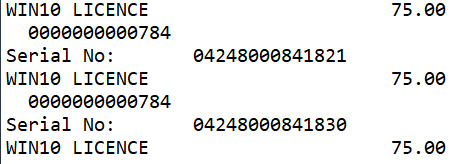




## 5.2 Data Extraction from Receipt

### 5.2.1 Product Description

The receipt has a product description:



The product description is required for CSV file (see section 4.2 for mapping). As mentioned earlier the description does not map correcting into AT and will need to through config stage of the utility. The description above corresponds to a column on the CSV file, see below:

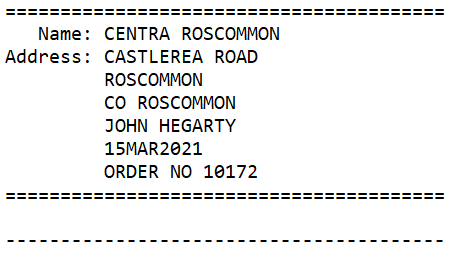
The utility will (1) identify the location of the description on the receipt (2) map the correct description (3) print the correct description to the CSV, which AT will accept. See graphic below:



Receipt Mapping CSV

### 5.2.2 Company Name

The receipt contains a company name:



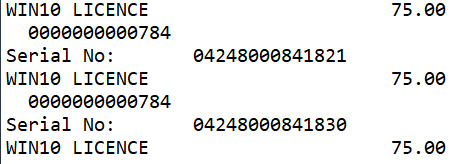
The company name is required for the CSV file and requires mapping, the same procedure as above is required, see graphic below:



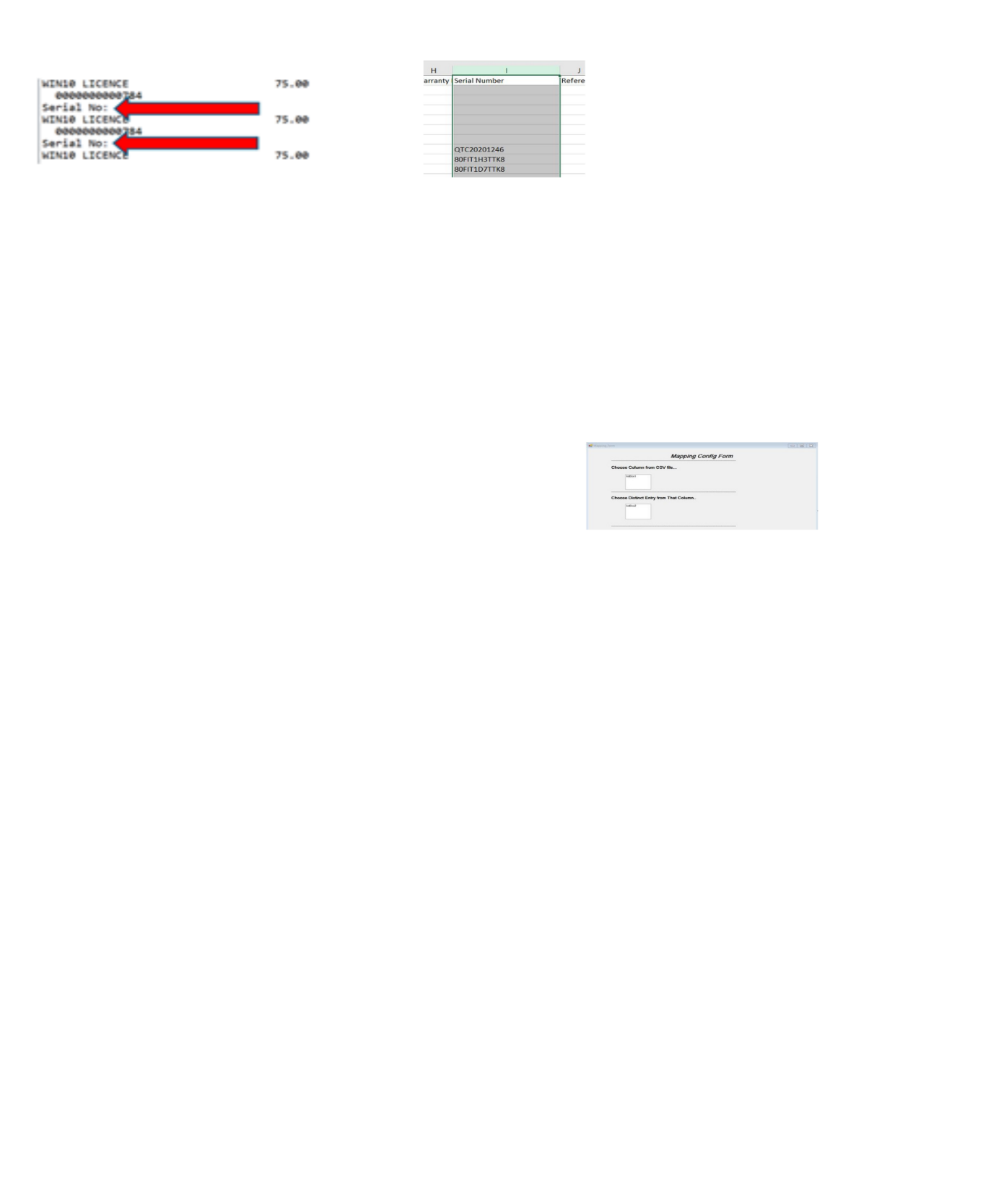
Receipt Mapping CSV

### 5.2.3 Serial Number

The receipt contains serial numbers:



The serial numbers are required for the CSV file. A straight copy and paste from receipt to CSV file should work in this instance:



## 5.3 Example of CSV Column Headers

After the parsing of the receipt the following column headers will be populated:

* [required] Product Name.
* [required] Company.
* [required] Install Date.
* Serial Number.

This leaves 2 more columns that need to be manually or via the config function, namely:

* Configuration Item ID [updates only].
* Reference Name.

### 5.3.1 Configuration Item ID [updates only]

This inputs source is from AT. It can be entered via the config function.

### 5.3.2 Reference Name

This inputs source is from AT. It can be entered via the config function.

### 5.3.3 [required] Install Date

This input is required from AT, user discretion is required here as to the cell value, via config function. The date itself seems to have no bearing, other than a required input from AT.



Figure 19 Required Column Headers

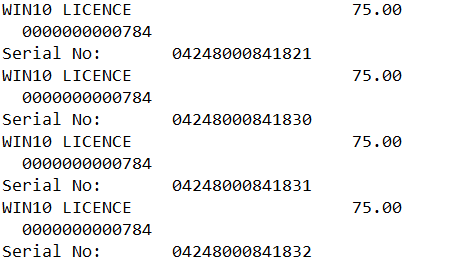
## 5.4 Design Patterns in Receipt

**Important**

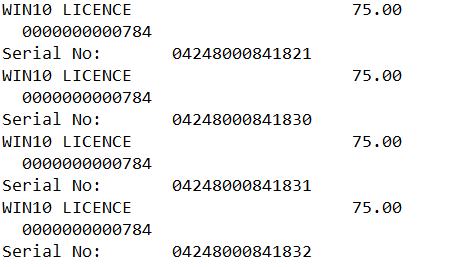
The utility Looks for patterns in the receipt in a bid to identify what type of sale is being conducted, what area of the receipt needs to read in (could be a single line, two lines, three lines etc). It is very important that the utility knows all of the patterns at play, so as not to miss any sales. Below are examples:

### 5.4.1 Serial Number Pattern

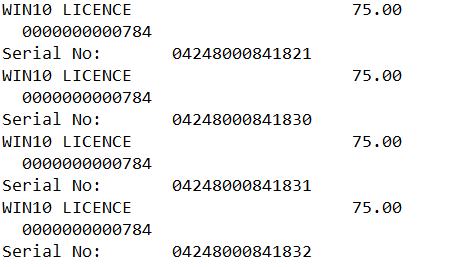
Here we can see a pattern in the receipt which evolves around printed serial numbers. We can see from the graphic below the pattern consists of a description of the product on the first line:



And the associated PLU on the second line:



And finally on the third line, a unique identifier in this instance Serial Number.

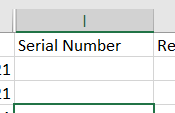
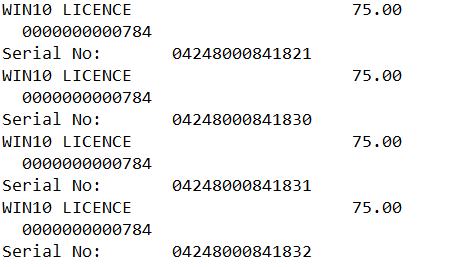


The utility will:

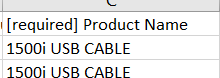
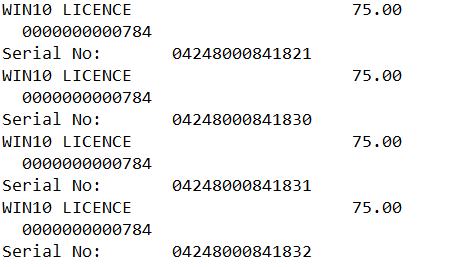
* search the receipt.
* stop when it reaches the unique identifier.
* Hold the value of the identifier (serial number) in memory.
* Grab the previous line (PLU) keep it in memory.
* Grab the line before that and use it for the item description.

At this point the utility will be able to populate the following columns in the CSV with the information above:

The serial number column:

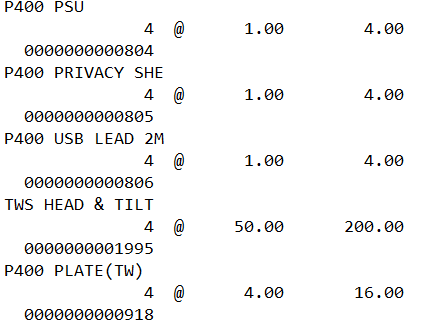
 

The [required] Product Name column:

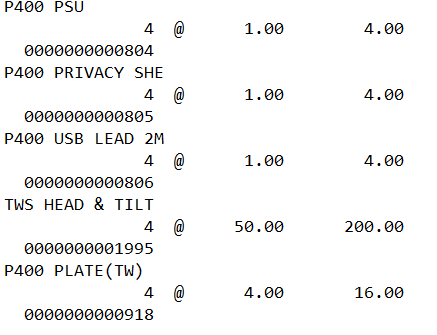
 

### 5.4.2 @ Symbol Pattern

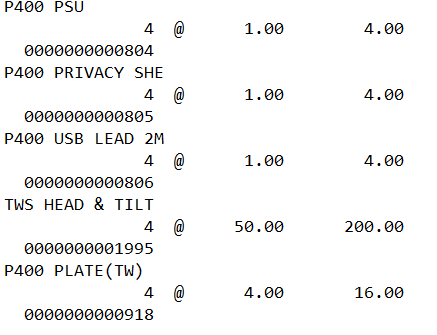
Another pattern identified is using the @ symbol, we can see from the graphics below the pattern consists of a description on the first line:



Following that a line describing the amount using a unique identifier (@):



And a third line providing a PLU:

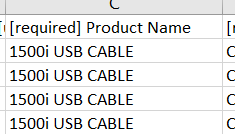
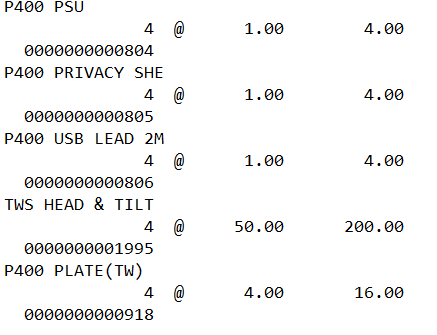


The utility will:

* search the receipt.
* stop when it reaches the unique identifier (@).
* Take number of items, their single unit value and total value and keep them in memory.
* Display each this transaction as single entries in the CSV
* Keep the description in memory.
* Keep the PLU in memory.

At this point the utility will be able to populate the following columns in the CSV with the information above:

The product name.

### 5.4.3 Two Line Entry Patterns

Another pattern found on the receipt is the 2 line pattern, which consists of a product name:



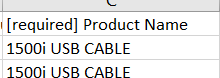
And the associated PLU:



The utility will:

* look through the receipt.
* Identify lines with no unique identifier (like @ or a keyword).
* Check that if the lines in groups of two are not followed by a line with a keyword.
* If not, group those two lines together in a single entity.
* Keep the value of the first line in memory and use it for the Product name in the CSV file.
* Keep the value of the second line in memory as it corresponds to the PLU.

At this point the utility will be able to populate the following columns in the CSV with the information above:

### Single Line Entry

Single line entry has been identified as a receipt entry, containing a description and cost.



The utility will:

* Look through the receipt.
* Check for single line entries not containing a keyword.
* Double checking, they are not part of serial number/symbol/two-line entry patterns described above.
* Keep the description in memory.
* Keep the associated unit price in memory.

At this point the utility will be able to populate the following columns in the CSV with the information above:

## Handling Refunds

The utility will automatically handle refunded items against an existing sale receipt. As an example, see the receipt below, which is of a typical sale (Figure 20 Typical Sale Receipt).



Figure 20 Typical Sale Receipt

When this receipt is pasted in (as explained in 4.2 Paste In Receipt), the grid in the utility is populated, shown below(Figure 21 Typical Sale Imported).

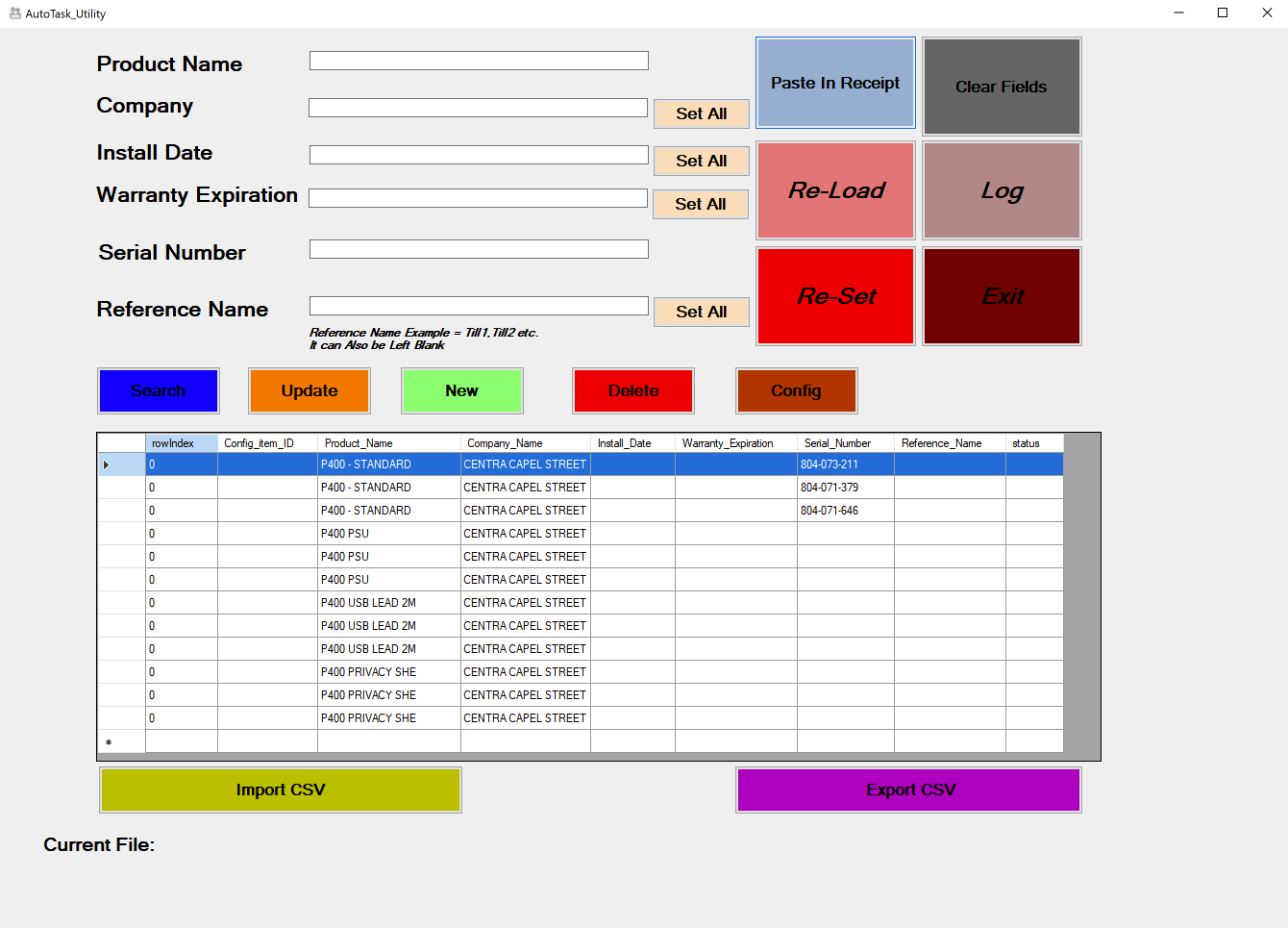


Figure 21 Typical Sale Imported

When a refund receipt is issued, like the one below (Figure 22 Refund Receipt),

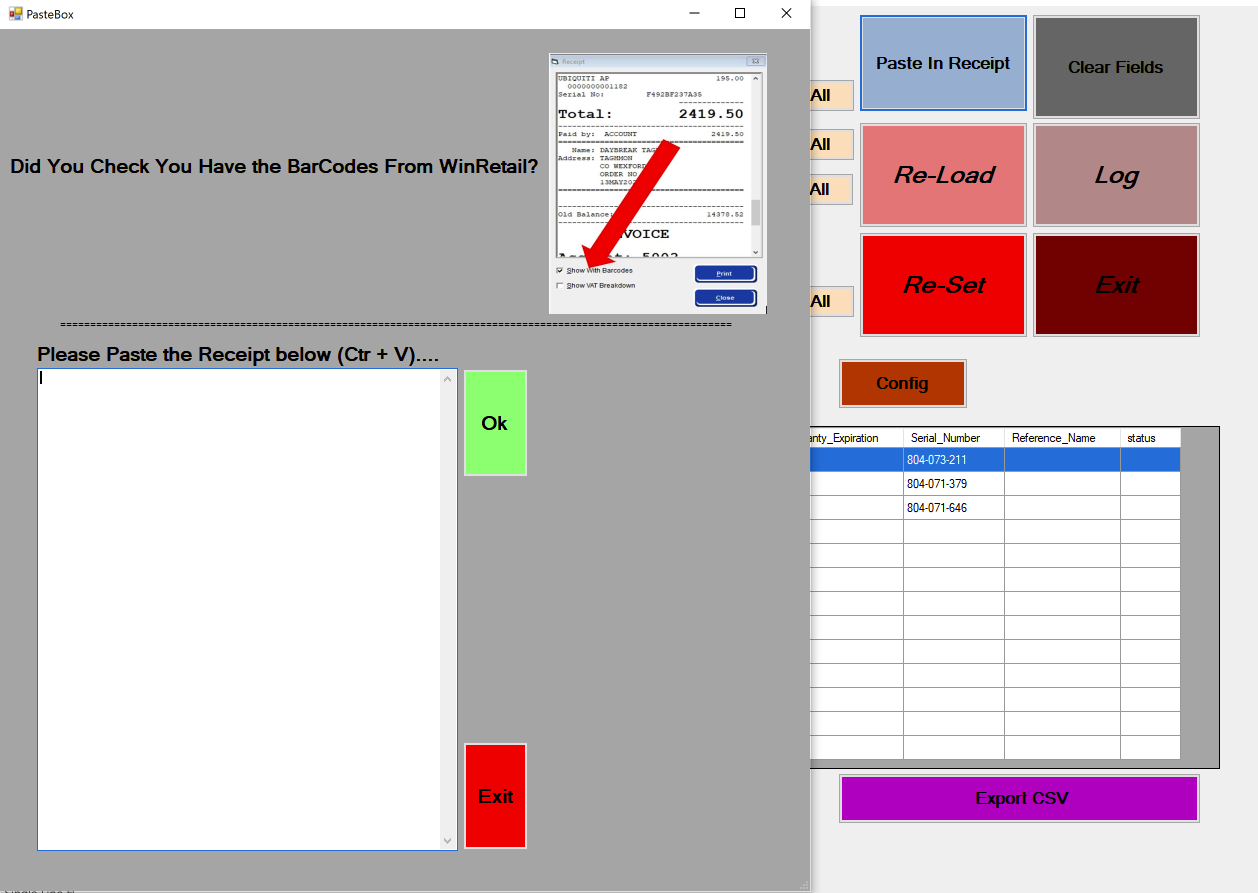


Figure 22 Refund Receipt

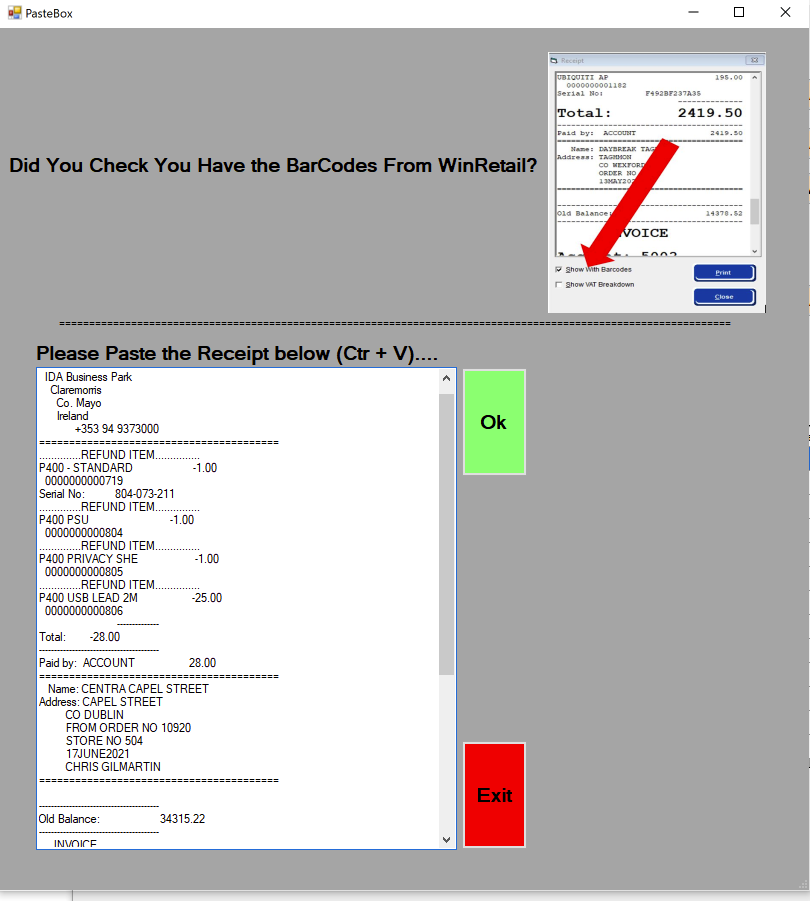
the user does not exit the application, instead keeps the sale receipt in the grid as pictured above (see Figure 21 Typical Sale Imported), and pastes in the refund receipt, by clicking *The Paste In Receipt* Button.



Which reopens the paste box form,



Now the contents of the refund receipt is pasted in and the Ok button is clicked (same process as 4.2 Paste In Receipt).



The resulting grid in the utility will have the refunded items clearly marked (Figure 22 Grid Updated After Handing Refund).

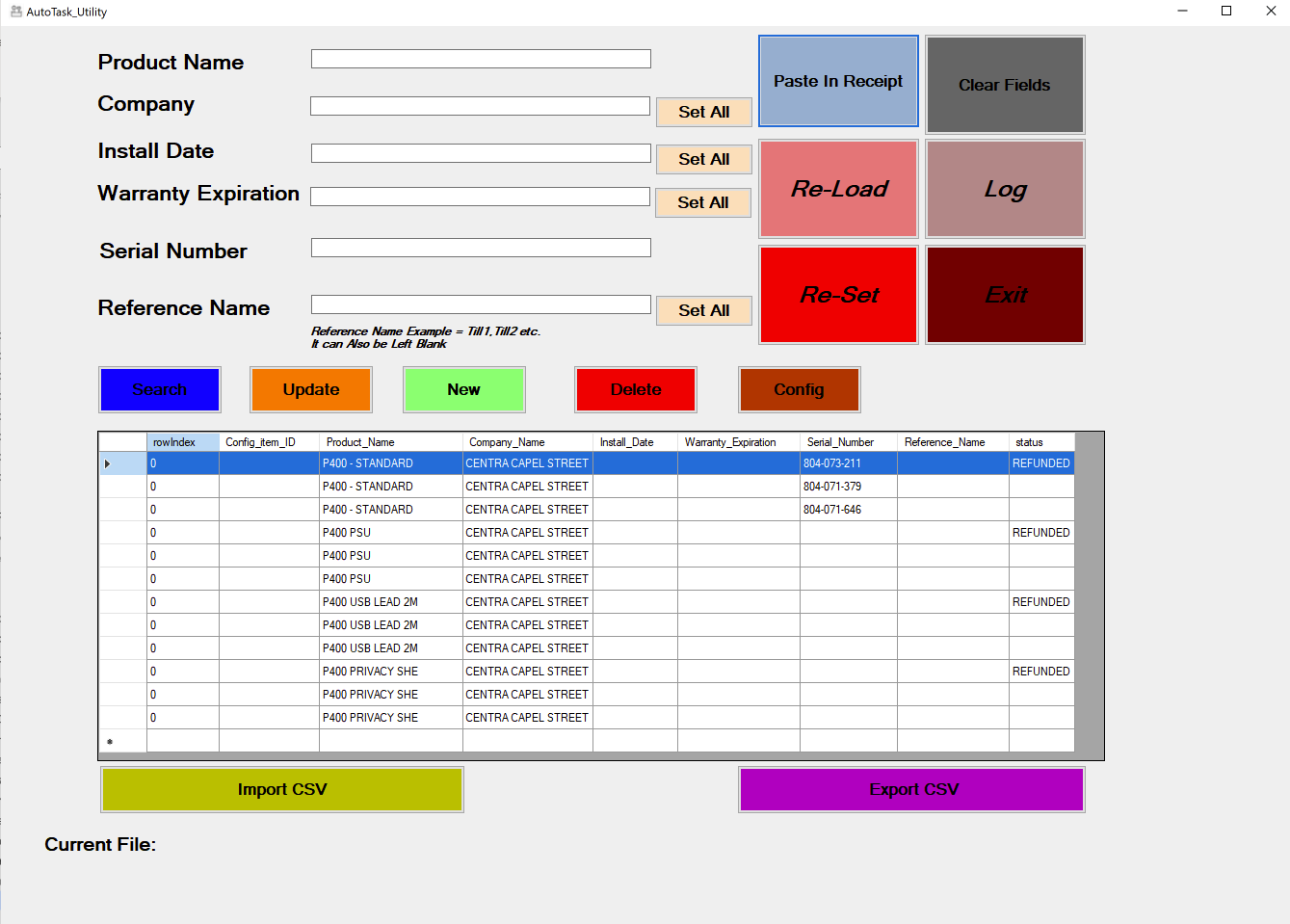


Figure 23 Grid Updated After Handing Refund

NB; When the users exports the data via the *Export CSV* button,



the refunded items will not be part of the export process (see).

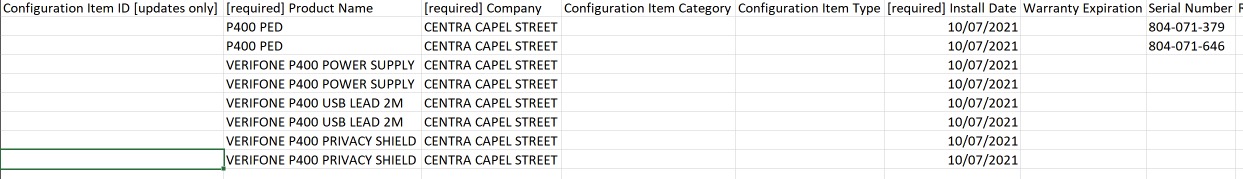


Figure 24 Refund Export Example

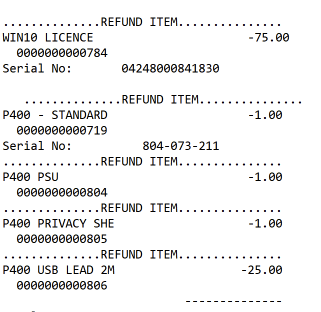
Note: there are now 8 entries in the export file, as opposed to 12 entries in the grid (Figure 23 Grid Updated After Handing Refund). The entries in the exported file have gone through the mapping process (see Config Function), hence why the description of the items looks different.

### 6.1 Error in Handling Refund

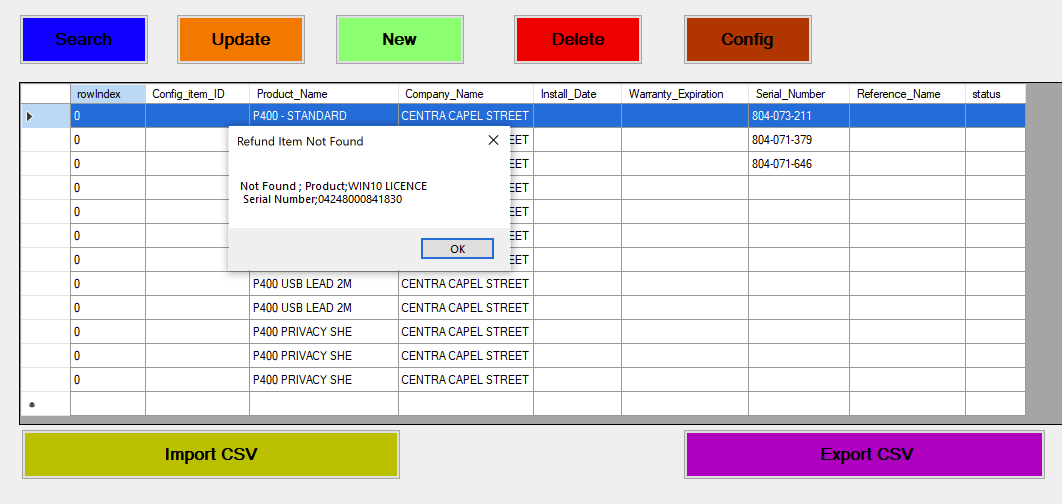
Let us look at the same process but introduce an intentional error. As mentioned, the refund receipt must correspond to the sale receipt, if not the utility will notify the user. At this point, the user needs to double check they have the correct refund/sale receipt. An example is below using the same sale receipt as above:



But, with a refund receipt that doesn’t correspond; a WIN10 LICENSE product is intentionally added to the refund receipt for demonstration purposes,



This item did not exist in the original sale receipt and is flagged by the utility, see below.

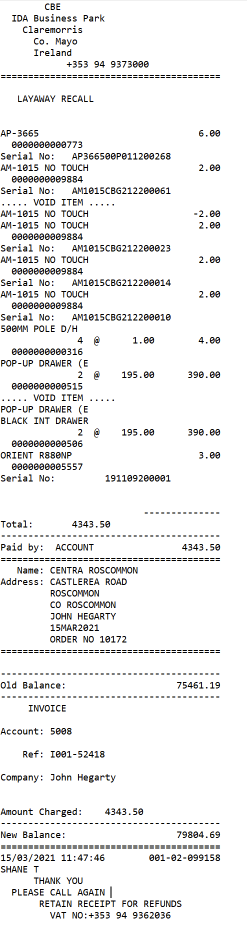


The user should make note of the “Refund Item Not Found”, and double check they have the correct receipt(s).

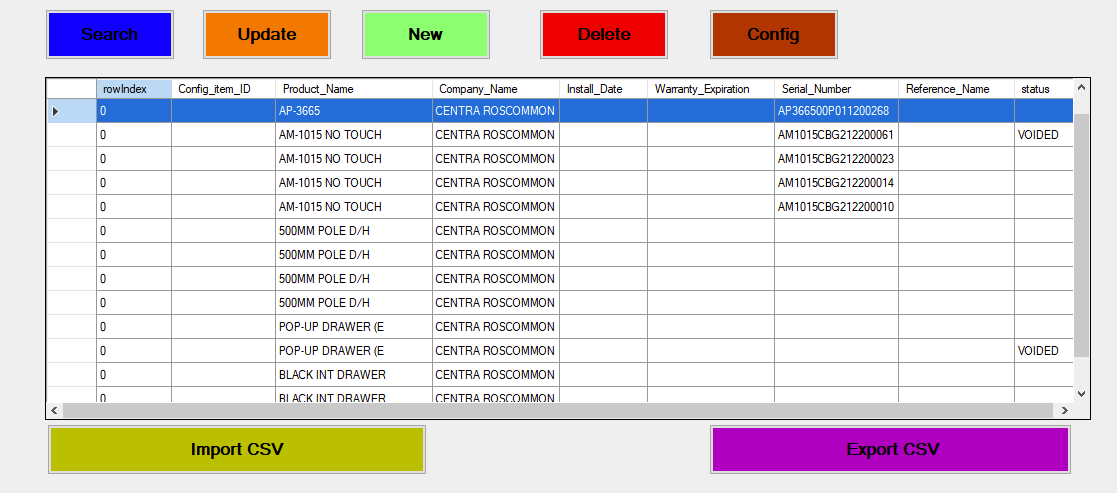
## 7 Handling Voided Items

The utility will automatically handle voided items. An example of a receipt that has voided itesm is below

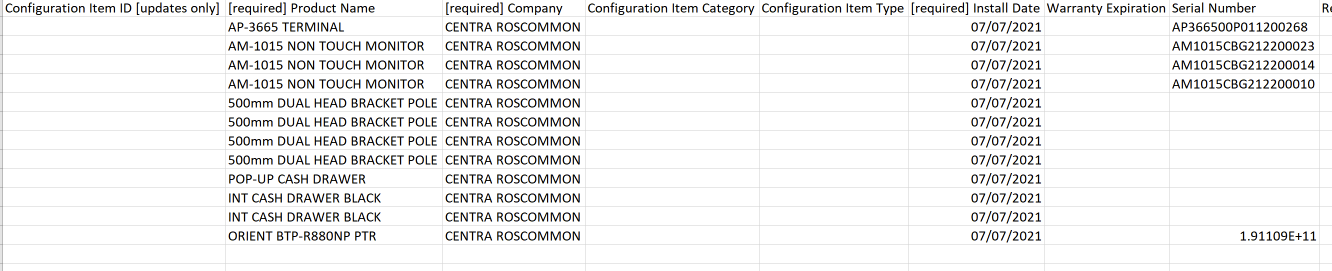
(please note it has been slightly modifed to fit).



The receipt is pasted in the same way as the previous(4.2 Paste In Receipt). And the resulting grid in the utilty clearly highlights the voided item(s).



As with the refunded items, voided items will not be exported, see below.



Note: there are 12 entries in the CSV file, as opposed to 14 in the grid above, this accuratly reflects the omission of the 2 voided entires.

## 8 Exceptions and Error Handling.

* Install date cannot be a future date from today’s date.
* Character limit in textboxes (tbc)
* Receipt totals will be checked after parsing, and an error message will advise user something is a miss

## 9 Configurations

Active directories hook up, should only be needed to be done once, on site.

The list box on the config file will be populated from a text file. Instructions on how the user can update this text file will need to be clearly explained.

.

# 10 Other Functionality

This section covers other areas of functionality.

## 10.1 Log

The log will be maintained with the records of the following:

* Login/Off with user credentials.
* Timestamp of all activities.
* Update (before and after).
* New record created.
* Deleted items.
* Search criteria (number of items returned).
* Any Set all values entered.
* All import/export csv file activity.
* Mappings Performed/config activity.

The utility will save a Log for each day. The current log is easily accessible when the user presses the Log Button,



A text document will pop up, see sample below. Descriptor lines start with a #, and the following line will have the values. For example the ##LOGIN:, User,Time line, is fowllowed by the line with the values LOGIN:LIAM-WIN-VM\liamj,9:55:04 AM. The values start after the :, and are separated by a comma. The first value is the username and the second value is the timestamp. This will continue throughout the txt file, recording all interaction of the user(s) that day.



## 10.2 Privacy & Security.

Security and privacy issues have been considered in the following aspects:

Utility will be loaded on in house machines.

Utility will record via log user credentials.

## 10.3 Deployment, Support & Maintenance

* Local only installation
* Support via Developer

## 10.4 Installation Files

The installation files that would be deployed are.

* Application files folder.
* Executable file.
* Application manifest.

# 11 Open Questions & Issues

Currently open questions and issues related to the following:

# 12 Contingency and Unknowns

none

# 13 Estimate Development Schedule

Below is a list of tasks and estimate duration to hand over code for testing

|  |  |
| --- | --- |
| **Tasks** | **Estimate Hrs** |
|  |  |
| Address Unknowns | 2 |
| **Login Form** |  |
| Active Directory code and test | 3 |
| Form creation | 1 |
| Login with password function | 1 |
|  | (7) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **AutoTask Utility form** |  |
| Form design | 2 |
| Config Function | 8 |
| Paste in Receipt Function | 8 |
| Delete Function | 1 |
| Search Function | 2 |
| Update Function | 2 |
| New Function | 2 |
| Delete Function | 1 |
| Logging (code behind) | 4 |
| SDS Review | 1 |
| Paste from receipt including form design | 3 |
| Code Review | 2 |
|  | (36) |
|  |  |
|  |  |
|  |  |
| **Mapping Form** |  |
| Form design | 1 |
| Choose column function | 1 |
| Choose distinct entry function | 1 |
| Choose what to replace function | 1 |
|  | (4) |
|  |  |
| ***TOTAL Hours*** | ***47*** |
| ***Total Days*** | 6 |

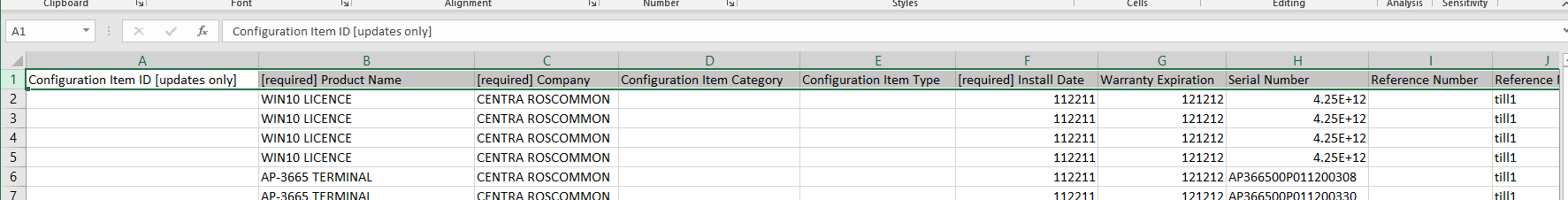
# 14 Testing

Developer testing/service manager/group director

## APPENDIX 1

Import CSV Format.

The csv format is to large to take a screenshot of , it will look like the graphic below,



The entire list of column headers that AT expects is listed below. Note: [Required] are necessary columns and expect an input.

Configuration Item ID[updates only]

[required] Product Name

[required] Company

Configuration Item Category

Configuration Item Type

[required] Install Date

Warranty Expiration

Serial Number

Reference Number

Reference Name

Number of Users

Contact

Location

Area

Contract

Service

Service Bundle

Billing Product

Billing Product Effective Date

Billing Product Expiration Date

Vendor

Service Level Agreement

Parent Configuration Item Serial Number

Description

Hourly Cost

Monthly Cost

Daily Cost

Per - Use Cost

Setup Fee

Company Link

Material Code(required if creating product)

Active / Inactive

Subscription Name

Reviewed for Contract

Subscription Description

Subscription Period Type[required if creating subscription]

Subscription Effective Date[required if creating subscription]

Subscription Expiration Date[required if creating subscription]

Subscription Period Price[required if creating subscription]

Subscription Material Code[required if creating subscription]

Subscription Purchase Order Number

Subscription Period Cost

Subscription Active

Subscription Vendor

Domain (Required if Category = Domain)

SSL Source(Required if Category = SSL Certificate)

UDF: 29682852 Username

UDF: 29682853 Password

UDF: 29682854 IP Address

UDF:29682861 OS

UDF: 29682862 Name

UDF: 29682864 Roles

UDF: 29682865 WAN IP:

UDF: 29682866 LAN IP:

UDF: 29682867 Brand

UDF: 29682868 SSID

UDF: 29682869 Security

UDF: 29682870 Location

UDF: 29682871 Make & Model

UDF: 29682872 Battery Life

UDF:29682873 Version

UDF: 29682874 URL

UDF: 29682875 Registrar

UDF: 29682913 AEM\_DeviceID

UDF: 29682914 AEM\_DeviceUID

UDF: 29682915 AEM\_Description

UDF: 29682916 AEM\_Manufacturer

UDF: 29682917 AEM\_Model

UDF: 29682918 AEM\_OperatingSystem

UDF: 29682919 AEM\_IPAddress

UDF: 29682920 User - defined field 3

UDF: 29682921 User - defined field 2

UDF: 29682922 User - defined field 1

UDF: 29682923 User - defined field 10

UDF: 29682924 User - defined field 7

UDF: 29682925 User - defined field 6

UDF: 29682926 User - defined field 5

UDF: 29682927 User - defined field 4

UDF: 29682928 User - defined field 9

UDF: 29682929 User - defined field 8

UDF: 29682981 User - defined field 19

UDF: 29682982 User - defined field 17

UDF: 29682983 User - defined field 18

UDF: 29682984 User - defined field 15

UDF: 29682985 User - defined field 16

UDF: 29682986 User - defined field 13

UDF: 29682987 User - defined field 14

UDF: 29682988 User - defined field 11

UDF: 29682989 User - defined field 12

UDF: 29682990 User - defined field 21

UDF: 29682991 User - defined field 20

UDF: 29682992 User - defined field 22

UDF: 29682993 User - defined field 23

UDF: 29682994 User - defined field 24

UDF: 29682995 User - defined field 25

UDF: 29682996 User - defined field 26

UDF: 29682997 User - defined field 27

UDF: 29682998 User - defined field 28

UDF: 29682999 User - defined field 29

UDF: 29683000 Server Type -PixelPOS

UDF: 29683001 User - defined field 30

UDF: 29683002 Bit - Locker

UDF: 29683003 PixelPoint - Backup

UDF: 29683004 Wholesaler

UDF: 29683005 Store ID

UDF:29683006 Symbol Group