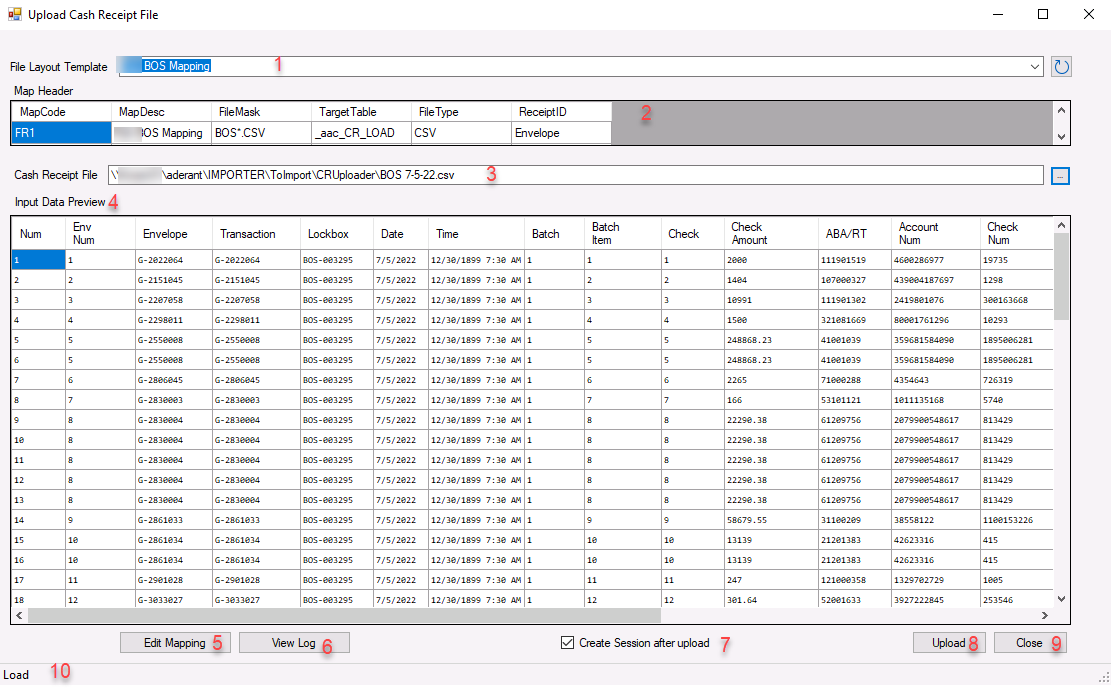
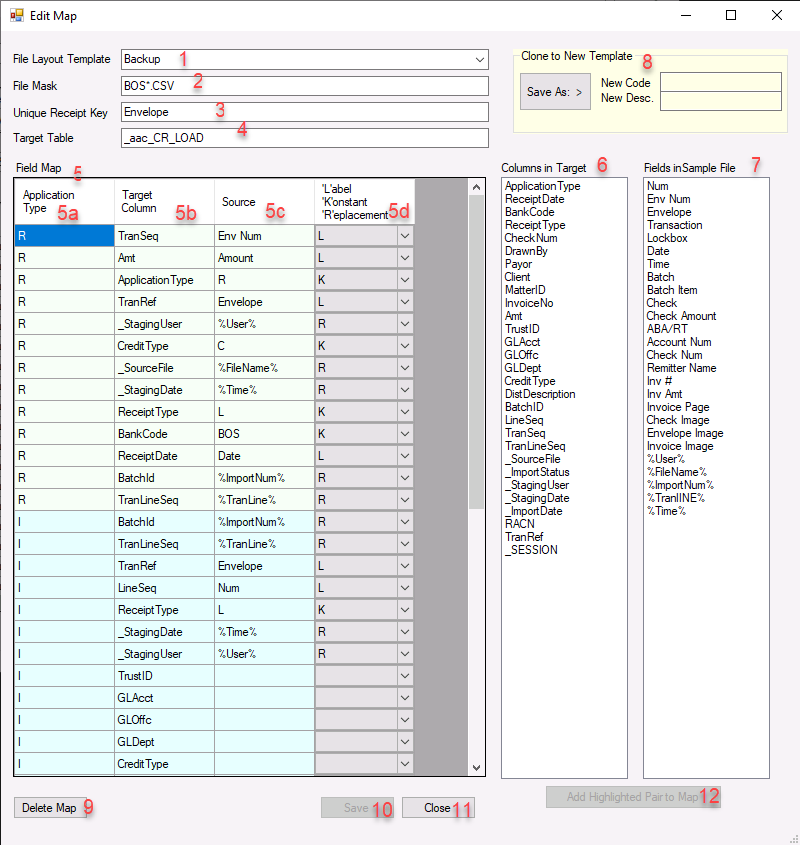
## The User Interface



|  |  |
| --- | --- |
| Screen Item | Usage |
| 1 | Selects the mapping template to be used.  This template defines how the data columns from the input file are mapped to the importer staging table used to create cash receipts.  See Item 5 to maintain the mapping. |
| 2 | Visual display of mapping options for reference. |
| 3 | The import file to be processed.  Use the Ellipsis button to select the file. File choices will be filtered based on the FileMask specified in the Mapping. |
| 4 | Data preview of the file to be imported |
| 5 | Will open the data mapping form to allow creation or maintenance of different mapping definitions.  See section on Map Editing |
| 6 | Opens up the process log file. This is a text file saved in the directory containing the import file..  File name is *{Import file name}*.err |
| 7 | If checked, will create a cash receipt session once the data is uploaded.  If not checked, the data will be uploaded to the staging file, but not processed. |
| 8 | Upload the file to the staging table (And create Cash Receipts session based on checkbox item 7)  If file has previously been uploaded, you will get a warning message that will allow you to continue or stop:    Once the file has been uploaded, you will get a completion message:    If Item 7 is checked, you will also receive a batch number and Cash Receipt session number in the form status Area (Item 10): |
| 9 | Close the form |
| 10 | Status area will display processing status messages. |

## Map Editing



|  |  |
| --- | --- |
| Screen Item | Usage |
| 1 | Mapping template  This is the description of the mapping  There will typically be a separate mapping for each available input file layout |
| 2 | File mask.  This will be applied when opening an import file |
| 3 | Unique Receipt Key  This identifies the column in the import file that will be used to group multiple items within a receipt. For example, if a single check is used to pay multiple bills, there may be a row for each bill application in the input file. This identifies the column used to group those application items together. |
| 4 | Target Table  This indicates the staging table that the data will be imported into |
| 5 | The data mapping.  There are multiple application types that can be mapped depending on the data required for each application type. |
| 5a | Type of application record  The application types are:   |  |  | | --- | --- | | R | Receipt Header – typically one per deposit item (i.e. Check or Wire transaction) | | I | Application to a client invoice (Bill) | | C | Client Credit, e.g. Unapplied Cash | | M | Matter Credit | | G | GL Entry | | T | Trust Receipt | |
| 5b | The target column in the staging table to receive the data.  Values in this column must match a column name in the Target Table specified in Item 4 |
| 5c | The source column in the Import File.  Values in this column must match columns in the input file. |
| 5d | Mapping function   |  |  | | --- | --- | | L | Label – Source column labeled with ‘Source’ value (5c) will be moved into the Target Column (5b) | | K | Constant – the constant value in Source (5c) will be moved into the target Column (5b) | | R | Replacement – System values indicated by the replacement variable in Source (5c) will be moved to the Target Column (5b)  Available replacement variables are listed in 7 with the format %*{value}*%. | | {blank} | No mapping will be done into the Target Column (5b) | |
| 6 | A listing of all available columns.  This list is populated from the database based on the table indicated in item 4. |
| 7 | Fields in Sample File  If a file is selected on the User Interface above, Item 3, the column sin that file will be listed here.  If a cell is selected in 5c, double clicking an item in 7 will update the value in 5c with the selected item from 7. |
| 8 | Clone Template  Entering a code and description, then clicking “Save As: >” will cause the current mapping definition to be cloned the description new code with the new name.  This can be used as a shortcut to begin a new mapping. |
| 9 | Delete Map  Will permanently remove a mapping definition. |
| 10 | Save  Will save any pending changes to the mapping displayed. |
| 11 | Close  Will close this form and return to the main user interface.  If there are unsaved changes, you will be prompted:    Select Cancel to return to Mapping screen and click Save to save pending changes.  Select OK to discard all pending changes (Except Cloning operations) |
| 12 | Add Highlighted Pair to Map  Enabled if a column from item 6 and a column for Item 7 are selected.  Will add a row to the mapping, 5b and 5c, with the selected pair |

## Infrastructure

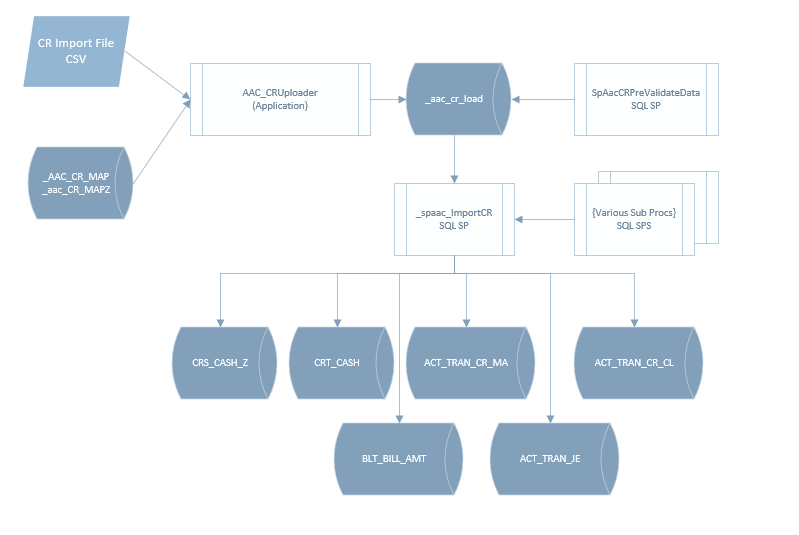
The process works by reading the selected import file and uploading it into a staging table in SQL.

A validation procedure is then run against that staged data and will enforce business rules to handle various situations.

Once the data is validated and business rules applied, the data in the staging table is used to create a cash receipt session, with multiple cash receipts as needed, and multiple applications of cash within each cash receipts.

Each import file should generate one CR Session. Each unique set of data with a common link as indicated by the “Unique Receipt Key” (Mapping item 3) will generate a receipt in that session and each line item with that key will generate an allocation record within the cash receipt.

### Data Flows\*



\*Notes:

Each file will create one Cash Receipt Session

Each group of rows with matching receipt keys*[[1]](#footnote-1)* will create one Cash Receipt

Each individual rows within each group of rows will create one Cash Allocation Detail row.

### Supporting SQL Procedures:

The operation of each procedure is described following this table.

|  |  |
| --- | --- |
| Procedure Name | Purpose |
| SpAacCRPreValidateData | Analyze uploaded batch details in staging table, validate codes and apply default values for missing data according to business |
| \_spaac\_ImportCR | Driving procedure to read batch details from staging table and call sub-procedures to create Sessions, Receipts and Receipt Details |
| \_spaac\_CR\_Create\_Session | Creates a Cash Receipts Session Header |
| \_spaac\_Create\_Receipt | Creates a Cash Receipt transaction |
| \_spaac\_CR\_Bill\_Application | Creates allocation details to a bill within a cash receipt transaction |
| \_spaac\_CL\_Credit\_Application | Creates a Client-level credit within a cash receipt transaction |
| \_spaac\_MT\_Credit\_Application | Creates a Matter-level credit within a cash receipt transaction |
| \_spaac\_CR\_Bill\_Application | Creates allocation details to a bill within a cash receipt transaction |
| \_spaac\_CR\_Import\_Map | Import a mapping file (.map). Mapping files are created with the Export Map option. |

#### SpAacCRPreValidateData

This procedure will Validate and process the raw data uploaded into the staging table. It serves to validate the data being imported as well as to apply business rules as prescribed to enable the creation of a complete cash receipt session.

On first call for any given BatchId, the process will create a backup of the original data that was staged. Subsequent calls for the same BatchId will restore the original uploaded data before re-validating the batch. Backup data is stored in the \_AAC\_CR\_LOAD with a BatchId set to the negative of the original BatchId.

If the batch was successfully uploaded to a CR Session, as indicated by the \_SESSION column having a non-zero value, it cannot be uploaded again. Validation will fail and no additional validation rules will be processed.

##### Validation includes:

|  |
| --- |
| All codes are validated against the database, including clients, matters, bank codes. |
| Verifies that there is enough data is available to enforce business rules. For example, there must be a viable combination of Client, Matter or Bill Numbers provided. |
| If an invoice number and a matter are provided, the invoice must be for the indicated mater. |

##### Example business rules are:

|  |
| --- |
| If Client and Matter are undefined, but a valid bill number is, derive the client and matter from the bill number. Works for single matter bills only. |
| If Client and Matter are undefined, and a valid bill number is for a multi-payor bill, Apply the cash as a client-level unapplied cash receipt.  **Option**: Apply the cash as a matter credit to the specified matter. |
| If the receipt header does not carry the detail totals, update the header total to match the sum of the detail amount. |
| If the header payor is missing, take the payor ID from the first detail row with a valid client specified. |
| If the cash receipt amount for a bill does not match the AR balance on the bill, Apply the cash as a matter level unapplied cash transaction.  **Option**: If Amount is less then AR balance, apply as partial payment  **Option**: If Amount is more than AR, apply AR balance value to bill and save remainder as unapplied cash for the matter. |

#### \_spaac\_CR\_Create\_Session

This procedure will create a cash receipt session header in CRS\_CASH\_Z using data from the “R” rows in the staging table.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Name | Default | Usage |
| @BatchId | Required | The BatchID Value from the staging table |
| @transeqfilter | -1 | Optional filter to process a single transaction item from the staging batch. |
| @Session | OUTPUT | Will return the session number of the created Cash Receipt Session |

##### Required “R” Row Staging Data

* ReceiptDate
* BankCode
* \_SourceFile
* Amt (Session Total)

#### \_spaac\_CR\_Create\_Receipt

This procedure will create a cash receipt item in CRT\_CASH using data from the “R” rows in the staging table.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Name | Default | Usage |
| @BatchId | Required | The BatchID Value from the staging table |
| @Session | Required | The Session to add the receipt to. This will be the value returned from \_spaac\_CR\_Create\_Session |
| @TranSeq | Required | The transaction line sequence within the staging batch. |
| @ReceiptNum | OUTPUT | Will return the receipt number of the created Cash Receipt |

##### Required “R” Row Staging Data

* Amt (Receipt Total)
* InvoiceNo (Will create concatenated string of InvoiceNo values from all detail rows for the same receipt)
* CheckNum (Can be blank)
* ReceiptDate
* TranRef
* DrawnBy (Check Remitter)
* Payor

#### \_spaac\_CR\_Bill\_Application

This procedure will create receipt details to apply payment to a bill using data from the “I” rows in the staging table.

Certain data is retrieved from the system settings to use within this procedure. Specifically, Accounting default Office, department, profit center and employee codes are retrieved from GLM\_PARMS. The account uno for Bank Fees[[2]](#footnote-2) is retrieved from GLM\_CHART.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Name | Default | Usage |
| @ReceiptNum | Required | The cash receipt num to attach the allocation to. This will be the value returned from the \_spaac\_Create\_Receipt procedure. |
| @BatchId | Required | The BatchID Value from the staging table |
| @TranSeq | Required | The transaction line sequence within the staging batch |
| @TranLineSeq | Required | The Sequence within the receipt transaction. This will identify a single “I” row with in the receipt allocation rows. |
| @MatterUno | NULL | Optional override for the matter uno for the receipt.  This parameter is deprecated but included for compatibility. It was previously used to designate a specific matter on multi-matter bills. |
| @AmountToAllocate | NULL | Optional override for a specific amount to apply to the bill.  This parameter is deprecated but included for compatibility. It was previously used to indicate the amount to allocate to the @MatterUno for multi-matter bills |
| @AmtApplied | OUTPUT | Will return the detail amount actually applied |
| @Debug | 1 | Optional parameter to cause display of procedure debugging information |

##### Required “I” Row Staging Data

* Amt (Bill Credit Amount)
* InvoiceNo (Bill Num)
* \_AllocationMethod
* TranRef

The procedure can utilize different methods to allocate the receipt. These allocation methods are used if the Amt does not match the AR total exactly. These are utilized primarily for payments on multi-matter bills to prescribe how to apply the payment across the billed matters.

The supported allocation methods are:

|  |  |
| --- | --- |
| '1' (Default) | Apply amount, shortpay is applied top-down across matters. Will leave Open AR where not paid in full, overpayment amount will be used to create an Unapplied Cash entry on a specified matter. |
| '2' | Apply amount, shortpay prorated across all matters based on Matter AR balance as a ratio to the entire AR on the bill, Will leave Open AR where not paid in full, overpayment amount will be used to create an Unapplied Cash entry on a specified matter. |
| '3' | Apply amount, if shortpay, settle AR in full and record difference to GL account for bank fees |
| ‘4’ | Apply amount, if shortpay, generate writeoff transacitons to clear remaining AR. |

When allocating payment to a bill, for short pay receipts, the system default allocation plan[[3]](#footnote-3) is used. This will apply the receipt in the order listed here: Hard Costs, Soft Costs, Taxes, Fees, Retainer, On Account Disbursements, On Account Fees, Interest then Premium/Discount.

This procedure will create the following data items:

1. insert CR transaction row into BLT\_BILL\_AMT with allocated bucket amounts

2. update BLT\_BILLM to adjust AR amounts by subtracting paid amounts

3. update BLT\_BILLP to adjust AR amounts by subtracting paid amounts

For Hard/Soft costs - separately

4. insert into BLH\_PAID\_DISB CR Trans (Credit Amount) by Cost\_Code, Matter, BLT\_BILLED\_DISB UNO

5. update into BLH\_BILLED\_DISB CR Trans (PAID\_CREDIT\_TOT - allocated amount) by Cost\_Code, Matter

For Fees

6. insert into BLH\_PAID\_FEES CR Trans (Credit Amount) by Cost\_Code, Matter, BLT\_BILLED\_FEES UNO

7. update into BLH\_BILLED\_FEES CR Trans (PAID\_CREDIT\_TOT - allocated amount) by Cost\_Code, Matter

Optionally

8a. Create GL allocation for bank charges if \_allocationmethod = 3 and receipt amount less than total AR

8b. Create Write Off of ar balance if \_allocationmethod = 4 and receipt amount less than total AR

9. Create Transaction for Unapplied Cash under the general matter[[4]](#footnote-4) for the payor if CR Amount > AR Amount on bill

### Supporting SQL Tables

|  |  |
| --- | --- |
| Table Name | Purpose |
| \_AAC\_CRMAPZ | Defines the mapping choices. Each map defines a file source and layout. This table includes an input file mask used for opening the source data, the file type (XLS[[5]](#footnote-5) or CSV), the target table to be loaded and which data item is used to group separate cash receipts.  For XLS data type, the range of the data is also stored here. |
| \_AAC\_CRMAP | Defines the mapping of source data columns to target table columns. See the section on Map Editing above for a discussion of what mapping is supported. |
| \_AAC\_CR\_LOAD | This is the staging table for cash receipts. All data is uploaded into this table. Validation takes place against this table and ultimately, the Expert cash receipt tables are loaded from this data.  Each file is grouped by an unique BatchId. |

## Configuration

The following operating parameters need to be specified:

|  |  |
| --- | --- |
| Setting | Storage |
| Bank Fees Account Code  Specify the GL Account code that will be used to record Bank Charges | ExpertConfiguration  Preference.Applications.FirmCustom.CRUploader. BankChargesAccountCode |
| SQL sequence for Import\_Num | Use SQL command:  "CREATE SEQUENCE [Sequence].[Import\_Num] START WITH 1" |
| Security Resource | AccountsReceivable.Custom.CRUploader |
| Security Policy | Above resource can be granted as needed to a FW security Policy to grant execute rights |

1. Column TRANSEQ used to group transactions for the same receipt [↑](#footnote-ref-1)
2. As of Dec 9, 2022, the Bank Fees account number is hardcoded in the procedure. [↑](#footnote-ref-2)
3. This default order is the generally accepted order. If a different allocation plan is needed, the procedure will need to be customized. [↑](#footnote-ref-3)
4. General matter code is currently hard-coded in procedure [↑](#footnote-ref-4)
5. XLS requires the Microsoft.ACE.OLEDB.16.0 data source [↑](#footnote-ref-5)