Receipts Batch Import v.2.0

Configuration and Processing

Copyright

Author: Ken Evans

Contents

Introduction

Star Receipts Batch Import provides a mechanism by which cash receipt files can be imported and receipts allocated against invoices, with minimal user interaction beyond the initial set-up and configuration.

Once set up, files can be imported simply by placing them in a specified folder.

Not Supported

The following operations are not supported:

- Single ledger.
- Multi-currency.

  Receipts are imported as functional currency. Attempts to allocate against an invoice in currency will fail.
- Profit/loss.

Installation

The Receipts Batch Import runs as a Service, and can be installed either to a Server or to an individual user's machine.

Requirements

.NET Framework 4.0 must be available on the machine running the service.

Star PDM is not required on the machine running the service.

Once the Service is installed it must be run under a specified user account, which requires the following privileges as a minimum:

- Local System rights to the machine on which the service will run.
- Full Rights to the folder that the service will monitor for file drops.

  Dependent on the location of this folder and the location of the service, this may require rights to a share on another machine.

Star advise that a dedicated account be created, in preference to running the module under the account of an existing user.

The service operates with the regional settings of the user under which it runs, as defined on the machine upon which it runs. This must be taken into account when changing regional settings: logging in as a different user and changing the server's region may not be sufficient to effect the change required.

Backwards Compatibiity

The new service is not backwards-compatible with the scripts or data structures of the old service. It requires an update to the latest version of Star PDM.

The latest scripts for the Batch Import, however, are backwards-compatible and will work correctly with the older service. As a consequence, there is no need to update the batch import service in order to use the latest version of Star PDM.

Configuration

The service must be configured by modifying its configuration file, as described below. An incorrect configuration will prevent the service from starting.

The configuration file — Star.Nominal.Receipts.BatchImport.exe.config — can be found in the application folder.

Database Settings

The following settings govern access to the database into which the import will write receipts and allocations.

- UseManualSettings
- T2SFile
- SQLServer
- UserName
- Password
- DatabaseName
- ConnectTimeout
- CommandTimeout.

These settings match equivalents in the Star PDM configuration file, and in most situations they settings should be configured exactly as their counterparts in Star PDM configuration are configured. It may, however, be necessary to adjust CommandTimeout to allow longer intervals, as after the reading of a file the posting and allocation of all receipts in the file is performed in a single transacted SQL command.

When the service is started these settings are validated. Start-up will fail if they do not point to a valid database.

Process Settings

The process settings define the general behaviour of the service once it has been started:

- **ProcessPath** – Defines the path where the service will expect files to be placed for processing. This may be a local folder, a mapped network folder, or a UNC path.
- **ProcessMode** – Defines the processing mode. This may be one of two values:
  - **Interval** – Starting from a fixed time of day (on process start-up) and then at specified intervals (e.g. every 24 hours), the service will process any files found in the process path.
  - **Immediate** – The service will perform checks for files every minute, and will initiate processing if any are found. The time between a file being dropped and processing being started could be anywhere between 0 to 60 seconds. The time between a file being added to the process queue and it actually being processed may vary, depending on the total number of files in the queue and their size.

    If **Immediate** is missing or invalid, **ProcessMode** defaults to **Interval**.

  If **ProcessMode** is **Immediate** then the following settings - **ProcessTime** and **ProcessInterval** - are ignored.
- **ProcessTime** – If **ProcessMode** is **Interval**, ProcessTime defines the time at which the Service is initially started. It is entered in hours and minutes in 24-hour format (e.g. 01:30 is half past one in the morning and 13:30 is half past one in the afternoon).
  If **ProcessTime**is missing or invalid, its default value is 01:00.

  If the start-up time is imminent or already passed for the day then the **ProcessInterval** is used to determine the next processing time.

- **ProcessInterval** – If **ProcessMode** is **Interval**, ProcessInterval defines the size of the interval between processing executions (e.g. every hour, once a day). It is entered as the number of whole minutes between the last execution time and the next (e.g. 60 for hourly intervals). The minimum permitted interval is 10 minutes. If you require a smaller interval, set **ProcessMode** to **Immediate**.
  If **ProcessInterval**is missing or invalid, its default value is 1440 (i.e. once every 24 hours).

Bank Accounts

In order to identify the bank account to be used for the import, the Receipts Batch Import service requires that the Bank Code be included in the filename, with the start and end marked by an identifying character.

- **BankCodeMarker** – This setting defines the character used to mark the start and end of the bank code in the file name.

  if missing, the default value for **BankCodeMarker**is #.

The **BankCodeMarker** character must be valid for use on filenames, and should not be a commonly used character (this would be an issue if the bank code is not the first element in the name, or if the bank code contains the character).

For further details, see Filenames).

Service Start-up

During the start-up of the Service the configuration is read.

As part of the start-up the Service will:

- Test for the existence and access rights to the ProcessPath
- Test for the existence and access rights to the following sub-folders in the ProcessPath: Failed, Processed, and Queued
- Create those sub-folders if they do not already exist, and if the Service has sufficient access rights to the ProcessPath.
- Test for the existence of, and access to, the specified Star PDM Database.

If any of these actions fail the Service will not start.

Once the Service is running, in order to change the configuration you must stop the Service, edit the configuration file, and then restart the Service.

If a setting is missing or invalid, then if configuration setting defaults are defined those defaults are used, otherwise the Service is stopped.

Filenames

Filenames must be unique.

The requirement for unique filanames makes it impossible to import the same file twice in error.

The design of the Service permits separate bank accounts to have different file formats (e.g. fixed width, CSV). As a consequence, both the code that is used to identify the bank account (Bank Code) and the relevant file format must be included in the filename.

Bank Codes are recorded in the Code column for in the Star PDM database Nominal Accounts table, tblNominalAccs.

Processing expects the Bank Code to be the second block in the filename.

When processing begins, the Service splits each filename according to the placement of the character defined in the BankCodeMarker configuration setting (see Configuration: Bank Accounts).

In the following examples, the default Bank Code Marker # is used.

- #<Bank Code>#<Rest of Name>
  Split gives:
  Block 1 – Empty String
  Block 2 – <Bank Code>
  Block 3 – <Rest of Name>
- <Prefix>#<Bank Code>#<Rest of Name>
  Split gives:
  Block 1 – <Prefix>
  Block 2 – <Bank Code>
  Block 3 – <Rest of Name>

There can be any number of instances of the Marker after the Bank Code with no impact on processing. For example, the following example creates more splits, but since the Bank Code remains the second item there is no impact on processing.

- <Prefix>#<Bank Code>#<Rest of Name Part 1>#<Rest of Name Part 2>
  Split gives:
  Block 1 – <Prefix>
  Block 2 – <Bank Code>
  Block 3 – <Rest of Name Part 1>
  Block 4 – <Rest of Name Part 2>

The Marker cannot be used to split a filename prefix, as this changes the location of the Bank Code.

The following example would result either in a processing failure or, potentially, an import under the wrong bank account with <Prefix Part 2> incorrectly used as the Bank Code.

- <Prefix Part 1>#<Prefix Part 2>#<Bank Code>#<Rest of Name>
  Split gives:
  Block 1 – <Prefix Part 1>
  Block 2 – <Prefix Part 2>
  Block 3 – <Bank Code>
  Block 4 – <Rest of Name>

Except for the above constraints, you can name the files as you wish.

Bank Account Settings

In order for a Bank Account to be usable in the import it must be bound to an entity. This is achieved by setting the ContNominalID for the Bank Account in Star PDM database table tblNominalAccs to the ID of the relevant entity.

An entity can have many bank accounts, but each bank account can have only one entity.

Import formats may differ from one bank account to another, as each bank account has its own settings defining formats and rules. These are configured in the tblReceiptBatchImportSettings.

There may be only one set of settings per bank account.

There is currently no maintenance interface for these settings. They must be configured in SQL Server by editing database table tblReceiptBatchImportSettings directly.

tblReceiptBatchImportSettings contains the following information:

- **BankAccountID** – The ID of the bank account the settings are for.
- **UserID** – The ID of the user with which posts and allocations are to be associated.
- **FileFormat** – A value that indicates the format of the file. Valid values are:
  0 – Delimited File (e.g. CSV)
  1 – Fixed Width File.

The specified **FileFormat** determines which of the following settings are supported.

- **Delimiter** – Specifies the separator to use in delimited files (e.g. comma for CSV).
- **TrimWhitespace** – Trim white space from text while reading the file.
- **TrimLeadingZeros** – Remove leading zeros from the client reference when it's read.
- **EnclosingQuotes** – Enclose columns in a delimited file in quotes. Commonly used to allow the information in a column to contain the delimiter without it being misidentified as a splitting point.
- **ColumnWidths** – A comma-delimited list of column widths for processing Fixed Width files. Each delimited value indicates the number of characters in that column (e.g. 40,20,16 defines three columns of 40, 20, and 16 characters in width).
- **ClientRefIndex** – A 0-based index denoting the client reference column in the file (e.g. 0 would be the first column, 1 the second column, and so on).
- **ReceiptRefIndex** – A 0-based index denoting the receipt reference column in the file.
- **ReceiptAmountIndex** – A 0-based index denoting the receipt amount column in the file.
- **InvoiceRefIndex** – A 0-based index denoting the invoice reference column in the file. This column is optional in the import file. A value of -1 denotes that the column is not in the import file.
- **InvoiceAmountIndex** – A 0-based index denoting the invoice amount column in the file. This column is optional in the import file. A value of -1 denotes that the column is not in the import file.
- **TransactionDateIndex** – A 0-based index denoting the transaction date column in the file. This column is optional in the import file. A value of -1 denotes that the column is not in the import file.
- **PayPortalPaymentIDIndex** – A 0 -based index denoting the column in the file that holds the receipt ID of a third-party receipts posting module. This column is optional in the import file. A value of -1 denotes that the column is not in the import file.

  Even when the use of this column is defined it remains optional, allowing import of files from both a bank and third-party software under the same format.

- **PayPortalAllocIDIndex** – A 0 -based index denoting the column in the file that holds the allocation ID of a third-party receipts posting module. This column is optional in the import file. A value of -1 denotes that the column is not in the import file.

Even when the use of this column is defined it remains optional, allowing import of files from both a bank and third-party software under the same format.

- **PaymentTypeID** – The ID of the payment type to use in the receipt post.
- **DummyClientID** – The ID of the dummy client to use if the client cannot be identified.
- **DummyJobID** – The ID of the dummy job to use if the receipt is posted to a dummy client.
- **AutoApprove** – A Boolean flag, which indicates whether the allocation batch is to be posted as approved or unapproved.

Processing

As files are dropped into the ProcessPath they are added to a processing queue. When the configured point in time for processing is reached the queue is checked, and if it contains any files then processing begins.

Processing removes files from the queue one at a time, importing each one before moving onto the next, and continuing until the queue is empty.

Since the queuing of files for processing and the actual processing of files are separate, it is possible to drop files into the ProcessPath and have them included in a currently executing process run, rather than wait for the next run. This is possible so long as the queue does not empty before the dropping of the files.

When processing a specific file, the following actions are performed:

- The file is moved to the Queued sub-folder.
- The filename is checked for the bank code, and relevant bank account settings are loaded.
- The contents of the file are loaded.
- The file contents are passed to the SQL Server, to be written to the database and allocated.
- The file is moved to either the Failed or Processed sub-folders.

Each of these steps is described below.

Movement to Queued Folder

This action is to make it clear that the file is being processed, and that it can no longer be edited or deleted by the user.

Up to this point, users can remove any file from the ProcessPath, even if that file is already logged for processing. Processing checks for the existence of the next file to move to the Queued folder, and will skip any file no longer in the ProcessPath.

Bank Code and Account Settings Check

Once the file has been moved to the Queued folder, its name is split to obtain the bank code (according to rules explained above: see Filenames). The bank account ID is obtained the bank code, and settings are loaded and validated.

The import of a file might fail for any of the following reasons:

- The file header does not contain a bank code.
- The bank code does not tie to the Code column of a bank account in Star database table tblNominalAccs.
- The bank account is not tied to a single entity.
- No import settings are defined for the bank account.
- The UserID does not link to a valid StaffID in tblStaff.
- The user has no access to the entity to which the bank account is bound.
- A PaymentTypeID is required but is not defined.
- A PaymentTypeID is not required but is defined.
- A PaymentTypeID is required and defined but does not link to a valid ReceiptPaymentTypeID in tblPaymentTypes.
- The DummyClientID does not link to a valid client in tblClient.
- The DummyJobID does not link to valid job for the dummy client and bank account entity in tblJob .
- The position of the Client Ref in the import file has not been defined.
- The position of the Receipt Ref in the import file has not been defined.

- The position of the Receipt Amount in the import file has not been defined.
- The FileFormat is not a valid format.
- The FileFormat is delimited, but no delimiter is defined.
- The FileFormat is fixed width, and the number of columns defined is less than the minimum column position defined.

File Reading

Once the file format has been obtained the file is read.

Check are made that the file has a header record, and that the transaction date can be obtained (2nd column in the header). If no header is found, or if there is no date, then the system date will be used.

When reading the transaction lines, if a Transaction Date column has not been defined or the date is invalid, then the header transaction date will be used.

If any transaction line has fewer columns than are required by the import format's column positions, then the file will not be imported.

Writing and Allocating

At this point, all read transactions are passed to a transacted stored procedure, to be written to the database and allocated.

As each receipt is processed it is posted, either to an Allocated Batch or to an Unallocated Batch.

The allocated batch is used for any receipt for which all the information required to perform an allocation has been correctly provided by the import file.

A receipt can be placed into the unallocated batch for any of the following reasons:

- The client reference provided does not tie to a client record in the database.
- The client does not have any open jobs (jobs may be suspended).
- The invoice reference indicates that it should allocate against finance charges, but there are no finance charges for the client.
  Allocation against finance charges is indicated by the invoice reference column containing 'FinCh' (case insensitive).

  From this point forward, the term Finance Charge is used to mean both Finance Charges and Finance Charge Adjustments.
- The invoice reference indicates an invoice to allocate against, but that invoice is not valid.

  For an invoice to be valid it must exist, be of the correct entity, not be fully allocated, and not be multi-currency.
- The invoice and receipt must be of opposing signs.
  The script does not impose which should be positive or negative, just that they must be opposing in order to allocate.
- The invoice/finance charge must not be involved in any other draft allocations.
- The client record must be successfully locked.

Two client failures will also trigger the use of the Dummy Client and Dummy Job to post the unallocated receipt.

Batches are created when required, so by the end of processing there are no batches without receipts in them. The type of batch created depends on its intended final state. Where receipts are to be left unapproved, a Standard Receipts Batch is created. For an allocation batch which is to be approved, an Individual Receipts batch is created.

The line/transaction log records the reasons for the placement of receipts into the unallocated batch (just as the file log records reasons for file rejections).

Receipt posting follows the rules used by the main receipt posting module rather than the existing import (e.g. where the selected client is a child client, the receipt is posted to the parent client unless the ChildOwnFeesLedger setting is turned on). But the allocation sequence of individual splits may differ, since there is no grouping by job as occurs in the receipt posting UI.

Once a receipt has been posted: if it was posted to the allocated batch then allocations are calculated and posted; if it is to be posted against Finance Charges then allocations continue until either there is either nothing left to allocate against or a finance charge that cannot be allocated against is reached. Allocation rules for Finance Charges are the same as for invoices with one additional rule: the finance charge must fully allocate.

Since finance charges are allocated as they are posted, it is possible for allocation to cease even if there are finance charges later in the sequence that could have been allocated against due to validation rules.

Processing supports first-in, first-out (FIFO) and pro rata modes of allocation. For FIFO, splits are allocated in sequence, allocation ceasing once nothing remains of the receipt to allocate against subsequent splits. For pro rata (which can be activated using the Firm level switch ReceiptsProRata), the amount to be allocated against the invoice is proportioned between all of the splits.

Processing to pro rata allocations only occurs if the amount to be allocated against an invoice is less than the total remaining balance on the invoice. At all other times FIFO is used. Additionally the pro rata mechanism operates by adjusting invoice balances while using FIFO processing.

This may yield minor differences to pro rata in the Star PDM application.

Process Completion

The final step is to move the processed file into either the Failed or Processed folder, dependent on the outcome.

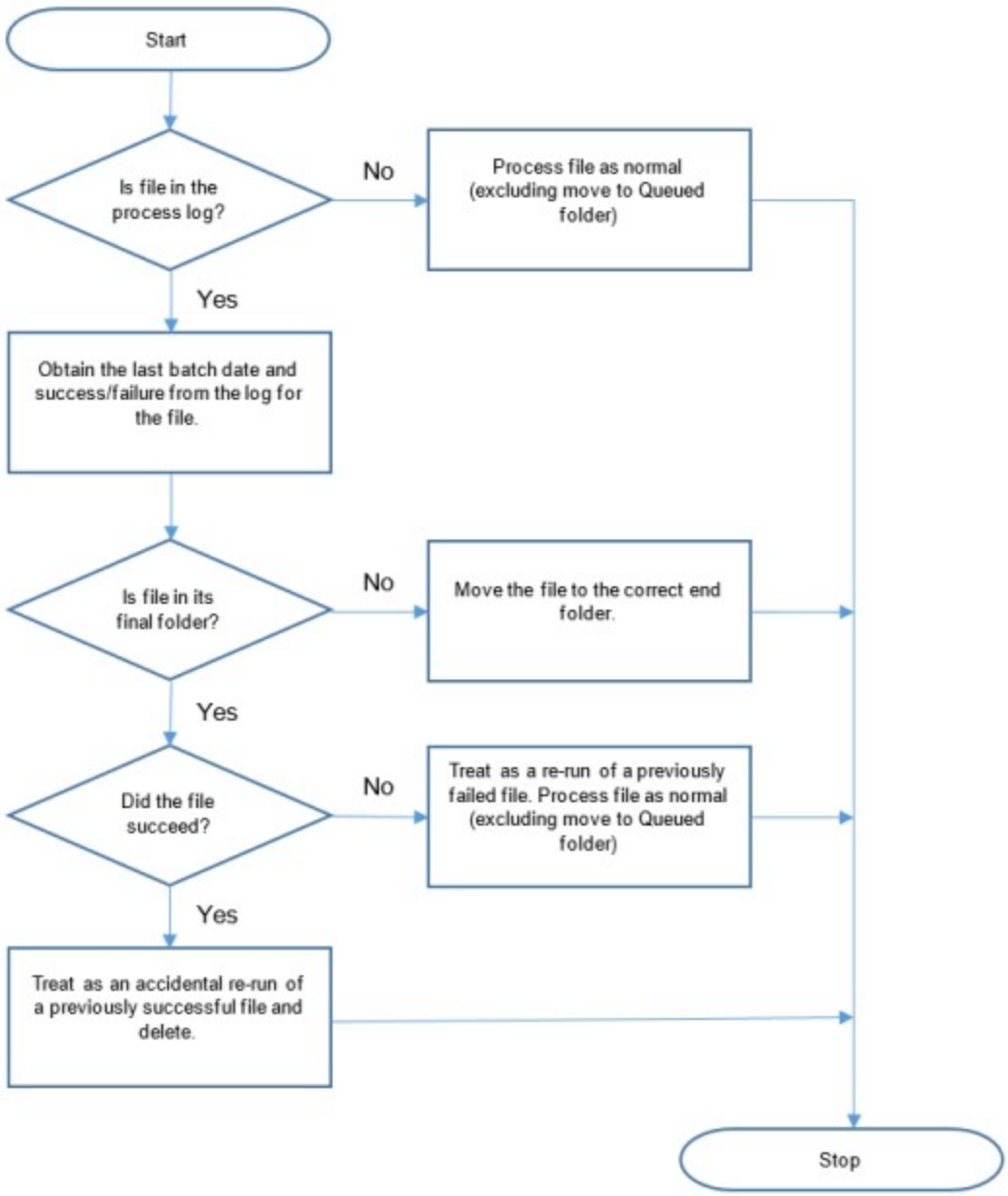
Within the relevant folder a sub-folder is created, named after the date on which the batch was run. All files processed on the same day are placed into the same folder, collecting processed files into manageable groups. If a file fails processing multiple times across separate runs on different days, this allows a file history to be maintained.

Processing then moves on to the next file.

Failure Recovery

In the event of a service failure during a file import, the service is restarted with the following steps taken to recovery:

1. Detection of any file currently in the queued folder.
2. Reinstating those files to the head of the process queue.
3. On processing of the re-queued file: detection of each file's original process status, and determination of whether to re-process, move, or delete the file.



Third-Party Receipt Software

Importing

Support for the import of receipts from a third party receipts module is provided by the ability to import the module's unique identifiers for the receipt and the allocation.

The third party module is expected to be able to generate an import file of the correct format for the target bank account, including the additional identifiers. For the import to utilise the columns, their index must be configured under the format settings (as must their column widths, if the file is fixed width format).

Unlike other column configurations, these columns are treated as optional, allowing banks to supply files that do not contain them. (The two columns are best placed at the end of the file.)

The third party receipt ID, when found, is recorded against the receipt record created in tblNominal, allowing the external module to identify the receipts it has created. Likewise, the third party allocation ID, when found, is recorded against the allocations in tblReceiptAllocationHistory and tblReceiptSplitAllocationHistory.

Both identifiers are also recorded in the Batch Imports log table, for reference.

Data Structure

- **tblNominal** - New column **PayPortalPaymentID**

  Varchar(50). Records against the imported receipt the contents of the specified 'PayPortalPaymentID' in the import file. If no payment ID is supplied it will record a zero length string.

- **tblReceiptAllocationHistory** - New column **PayPortalAllocID**

  Varchar(50). Records against the allocations the contents of the specified 'PayPortalAllocID' in the import file. If no allocation ID is supplied it will record a zero length string.

- **tblReceiptSplitAllocationHistory** - New column **ParentPayPortalAllocID**

  Varchar(50). Records against the allocations the contents of the specified 'PayPortalAllocID' in the import file. If no allocation ID is supplied it will record a zero length string.

The Receipts Module

The Star module acknowledges third party receipt import by preventing certain actions, even if the batch is unapproved.

- The batch cannot be deleted, amended, or have new receipts posted to it.
- Receipts cannot be deleted, amended, or reallocated.

As a consequence of these restrictions, if an issue occurs during import and is posted to the dummy Client, the only action that can be taken is to approve it against the dummy Client, and then create contras to move the amount to the correct location for allocation.

Audit Data Structure

The Star database tables tblReceiptBatchImportFiles and tblReceiptBatchImportLines record the audit of imports performed, transactions imported, and receipts created.

If import fails at file validation, then only a file is recorded, with no import line records in tblReceiptBatchImportLines.

tblReceiptBatchImportFiles

This table records the following details of each file that the import service attempts to import.

- **FileID** – A unique ID for the import attempt.
- **Filename** – The filename of the file that was processed.
- **BankAccountID** – Bank Account with which the file was associated.
- **BatchDate** – The date and time at which the module began processing the batch of files that included this file. Not the transaction date of receipts in the file.
- **TransactionDate** – The transaction date found in the file header or, if there is no header, the system date.
- **RecordCount** – The total number of rows in the import file.
- **TotalAmount** – The total value of the receipts imported.
- **Failed** – A bit flag to indicate that the import of the file failed.
- **Comment** – A description of any error that occurred.

tblReceiptBatchImportLines

This table records the following details of individual rows imported from the file.

- **FileID** – The ID of the file import the record belongs to.
- **LineID** – The number of the line in the file from which the record came.
- **ClientRef** – The value of the client reference column for the row.
- **ReceiptRef** – The value of the receipt reference column for the row.
- **ReceiptAmount** – The value of the receipt amount column for the row.
- **InvoiceRef** – The value of the invoice reference column for the row.
- **InvoiceAmount** – The value of the invoice amount column for the row.
- **TransactionDate** – The value of the transaction date for the row. If a transaction date is defined at row level, and it contains a valid date, then this date is used; otherwise the value in the TransactionDate field in the relevant file header record is used.
- **ClientID** – The ID of the client to which the receipt was posted.
- **ReceiptID** – The ID of the receipt record in tblNominal.
- **PayPortalPaymentID** – The ID of the receipt in a third party receipt posting module. If not in use, this will contain a blank string.
- **PayPortalAllocID** – The ID of the allocation in a third party receipt posting module. If not in use, this will contain a blank string.

Appendix:- Version History

Version 1.0 - Initial release.

Version 2.0 - Added support for third party receipt posting.