

USER MANUAL

& TECHNICAL MANUAL

Table of Contents

Getting Started 5

Before you begin 5

Brief overview 5

Get familiar with the interface 5

Navigation 5

Dashboard 6

Reconciliation Area 8

Match Transactions 8

Reporting 14

Matched Transaction Details 17

To un-match transactions. 18

Search For Transactions 18

Staging Area 20

Closing Balances 20

Import Transactions 21

Bank and Ledger Staging Screens 23

Maintenance Area 25

Accounting Period 25

SS Clearance File 26

Adjustments 27

View Process History 28

Administrator Area 29

Bank and Ledger Groups 29

Custom Processes 31

Custom Reports 32

Users and Access 33

Bank Accounts 35

Advanced Configuration 38

Database Overview 38

Account Related Tables 38

Stored Procedures 39

Application Settings 39

Technologies & Resources 40
.Net Blazor 40
MudBlazor 40
Closedxml 40
Closedxml.Reports 40
Dapper 40
Signal R 41
Excel Data Reader 41
Microsoft SQL Server 41
Installation 42
Requirements 42
Sample Installation 42
Database Schema 47
APPENDIX 1 – Scripts for Main Tables 48
account_bank_file_mapping 48
account_ledger_file_mapping 48
processes 49
roles 50
user_accounts 50
users 50
APPENDIX 2 – Scripts for Account Tables 52
accounting_periods 52
bank_transaction_groups 52
bank_transactions 53
bank_transactions_staging 53
closing_balances 54
ledger_transaction_groups 55
ledger_transactions 55
ledger_transactions_staging 56
process_logs 57
user_bank_columns 58
user_ledger_columns 58
custom_processes 59

reports_history 60
APPENDIX 3 – Scripts for Views 61
ledger_closing_balance_vw 61
ledger_trans_sum_vw 61
bank_trans_sum_vw 61
APPENDIX 4 – Scripts for Stored Procedures 62
bank_staging_process 62
import_ledger_transactions 62
ledger_staging_process 64
generate_ss_fILe 65
matching_process 65

Getting Started

Before you begin

Before you begin using the software, please ensure that the software is installed and configured properly. (See **Installation Section at End of this Guide**) Once, installed, use your browser and navigate to the url.

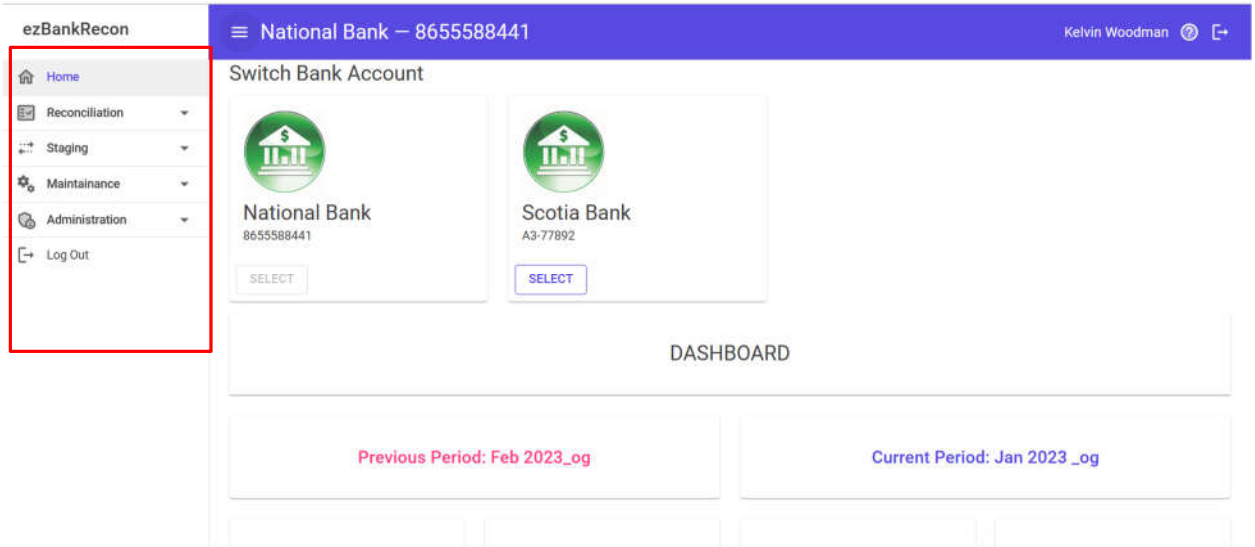
Brief overview

The EzBankRecon software provides a simple way to manage the reconciliation of accounts. Primarily it was designed to work with the SmartStream system. This manual will assist you with getting familiar with the functionality of the program.

Get familiar with the interface

Navigation

The main navigation of the program is located to the left of the page.

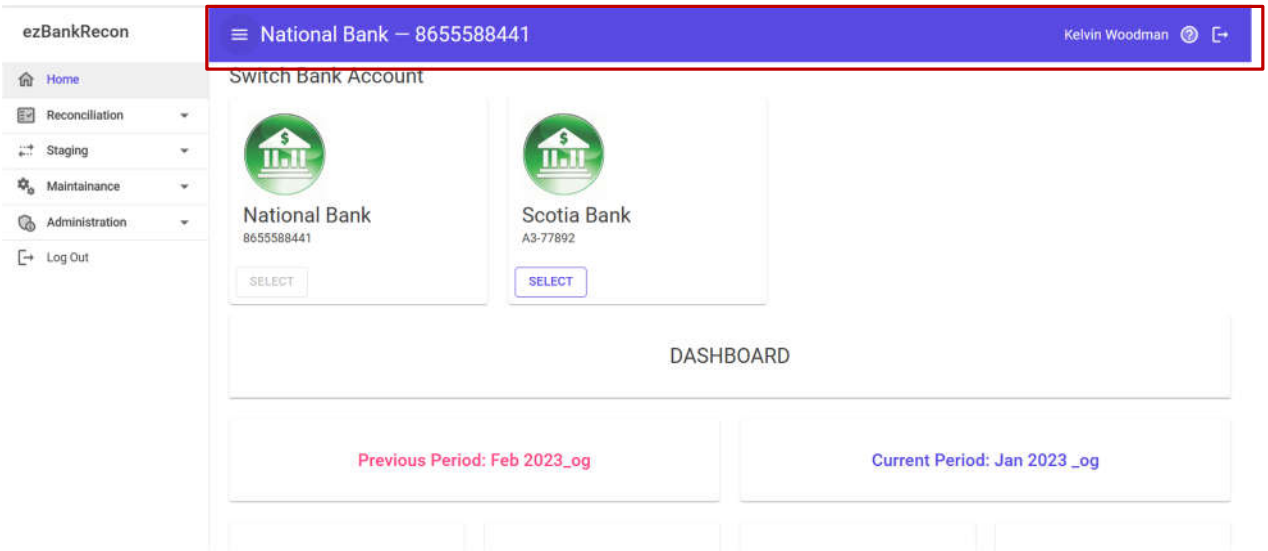


Note. The menu items that appear is based on the user security.

Menu	Role
Reconciliation	Regular User
Staging	Super User
Maintenance	Super User
Administration	Administrator

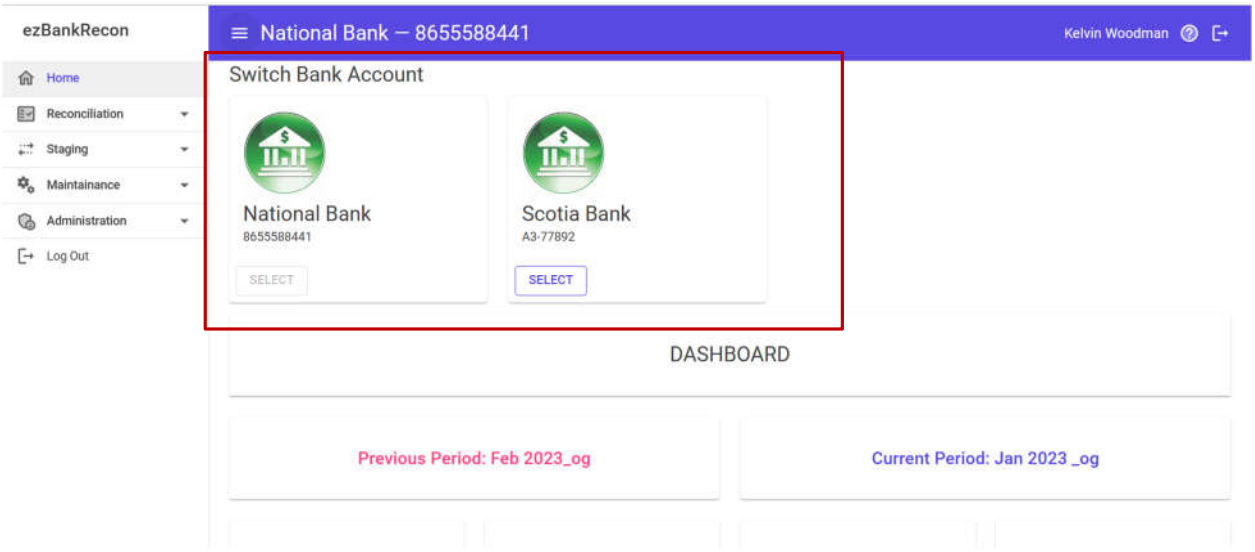
Next area of interest is the top bar.

The top bar shows the active account, the user who logged in, and also a shortcut to log out. The hamburger menu closes or opens the main navigation of the program



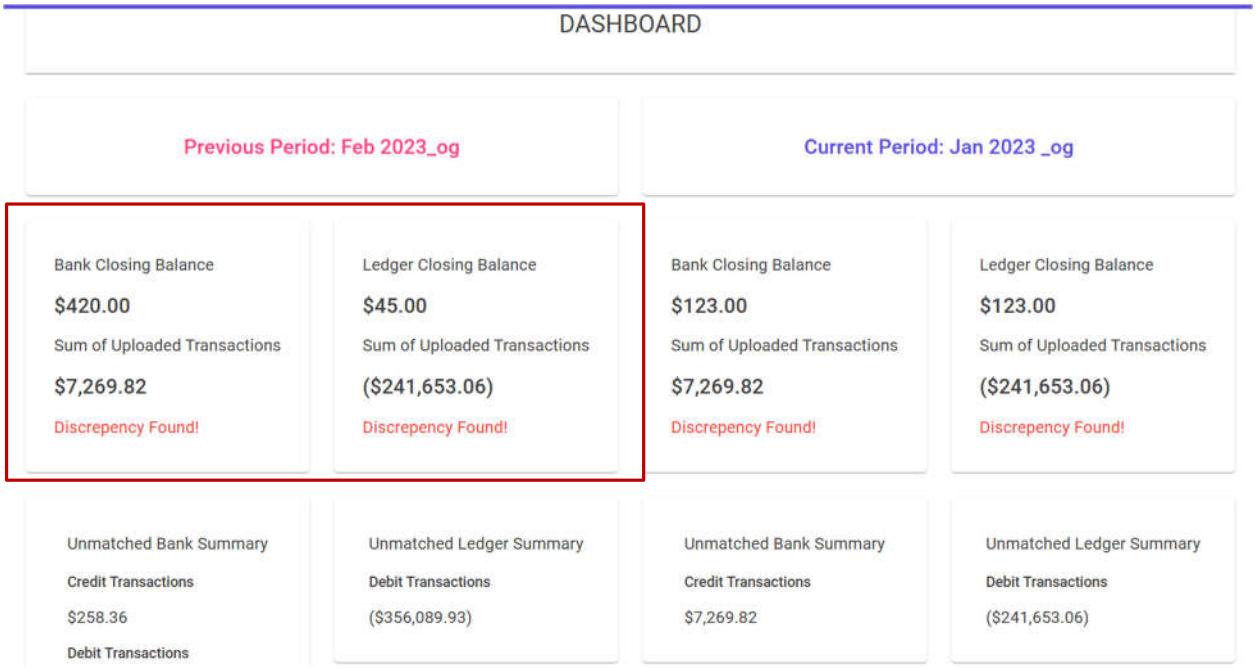
Dashboard

To Access the Dashboard click on the Home menu. The dashboard shows information on the last two periods of reconciliation. It also serves as the place where you can switch accounts if you have access to multiple bank accounts.



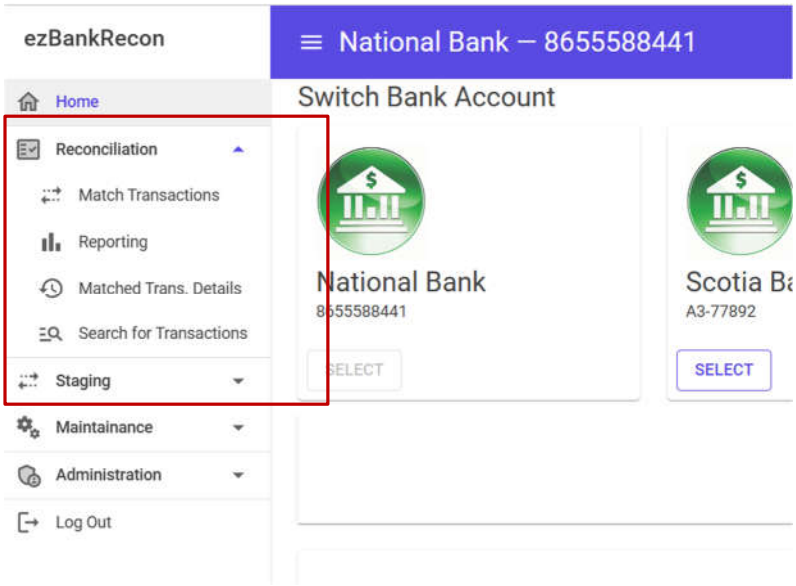
If you only have access to one account, the "Switch Bank Account" option will not show on the dashboard. Also, if you have multiple accounts, the **menu items** will not show unless you select an account to work with.

The dashboard also provides information on whether all the imported transactions match with the balance. A "Discrepancy" message is issued if the balances do not match. Discrepancies may arise due to not all transactions being uploaded, or wrong closing balance amount were entered.



Reconciliation Area

Let's look at the Reconciliation Menu.

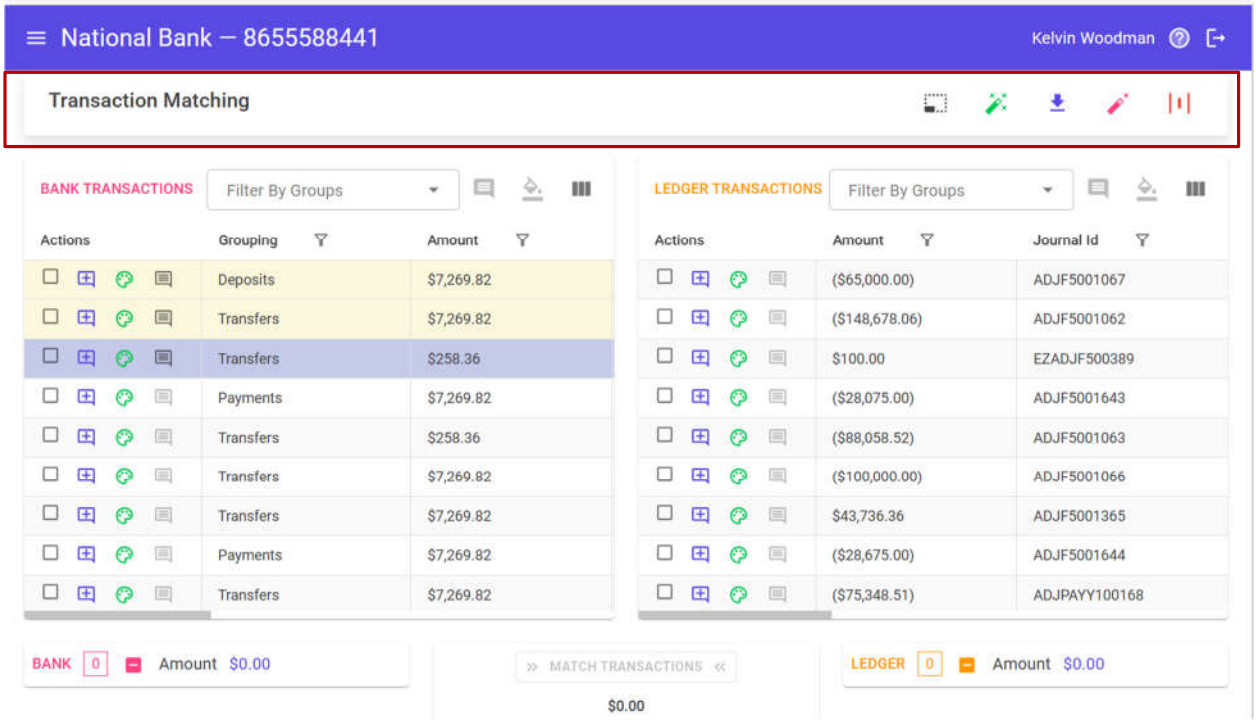


There are 4 sub-menu items.

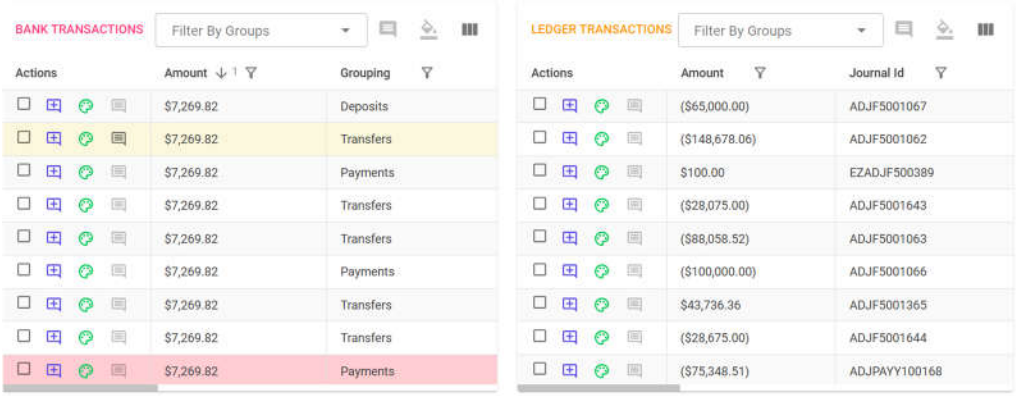
- Match Transactions
- Reporting
- Matched Trans. Details
- Search For Transactions

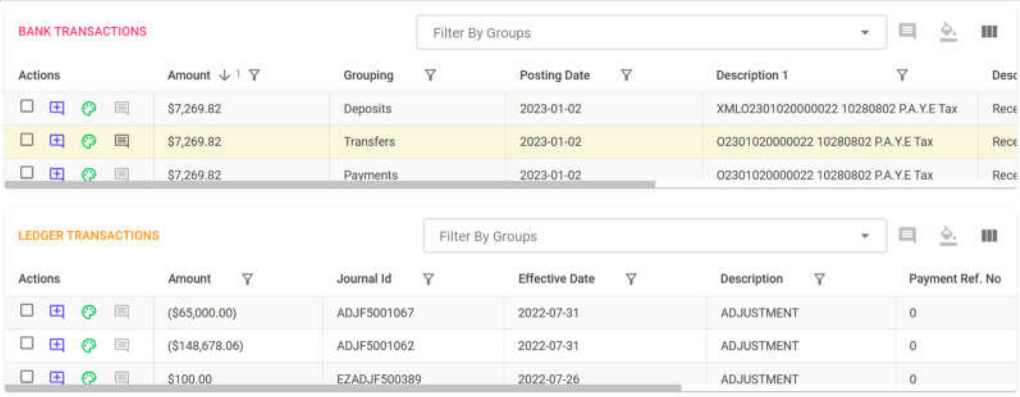
Match Transactions

This is where majority of the work will be done. Let's familiarize ourselves with the interface.

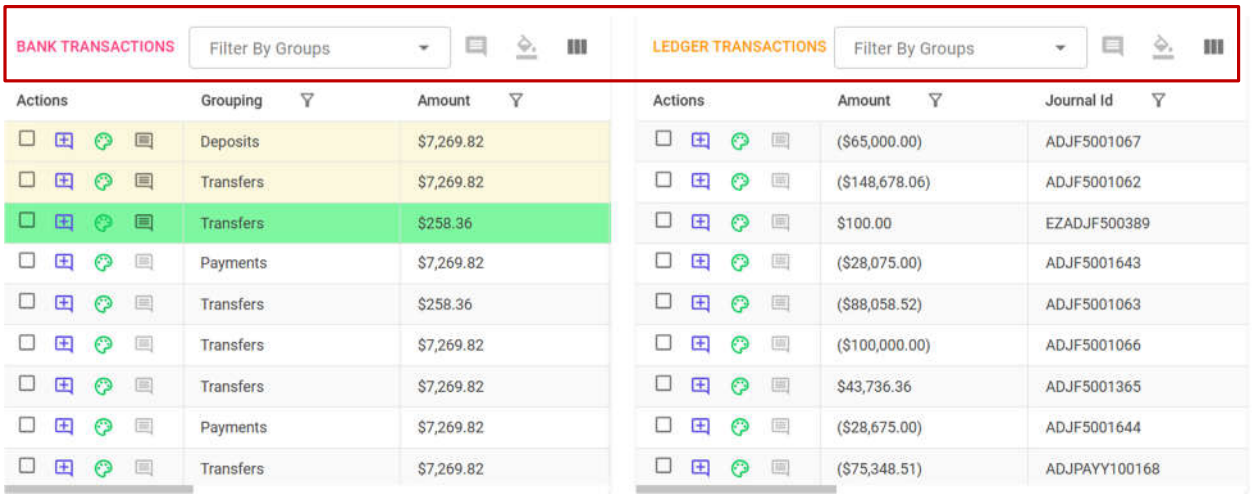


The top bar with the title "Transaction Matching" contains the following functionality:

Icon	Function
	Adjust the grid size to fit the screen. (Only necessary if the system failed to adjust it automatically)
	This runs the auto-matching process
	Pressing this button downloads the unmatched bank and ledger transactions to an excel sheet
	This runs any customs processes that the administrator developed
	This toggles between horizontal and vertical layouts for the bank and ledger grids. *Horizontal View (Side by Side)*  *Vertical View (Stacked)*

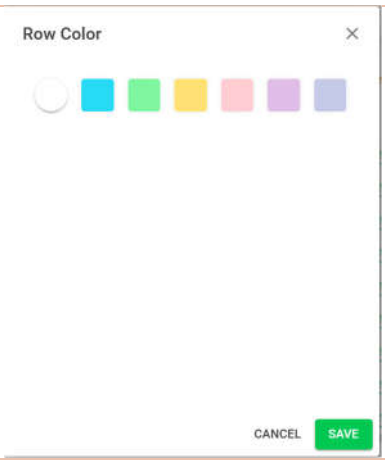


Now, let's move to the Bank and Ledger data grids.



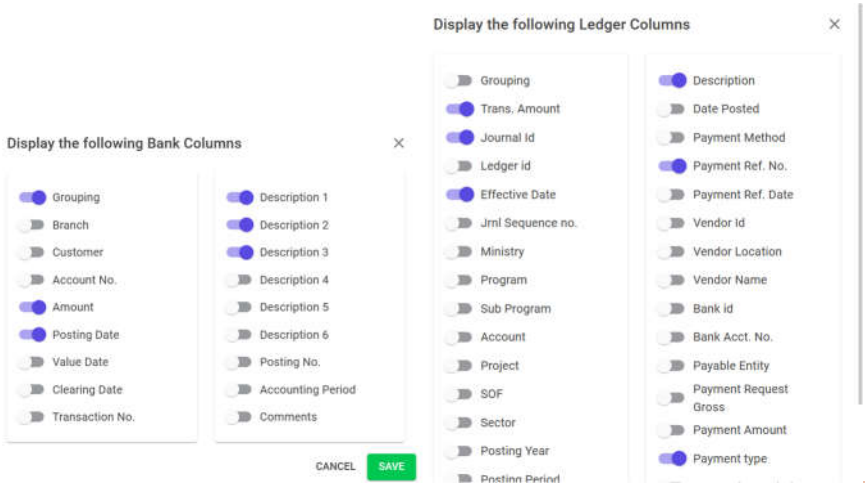
The top of the grids provide a quick filter, whereby the list will be filtered by a group selected. The icons provide the following functionality.

Icon	Functionality
	Insert user comment on all selected rows. This button will remain disabled until one or more transactions are selected. 
	This changes the row color of all selected rows. This button will remain disabled until one or more transactions are selected.

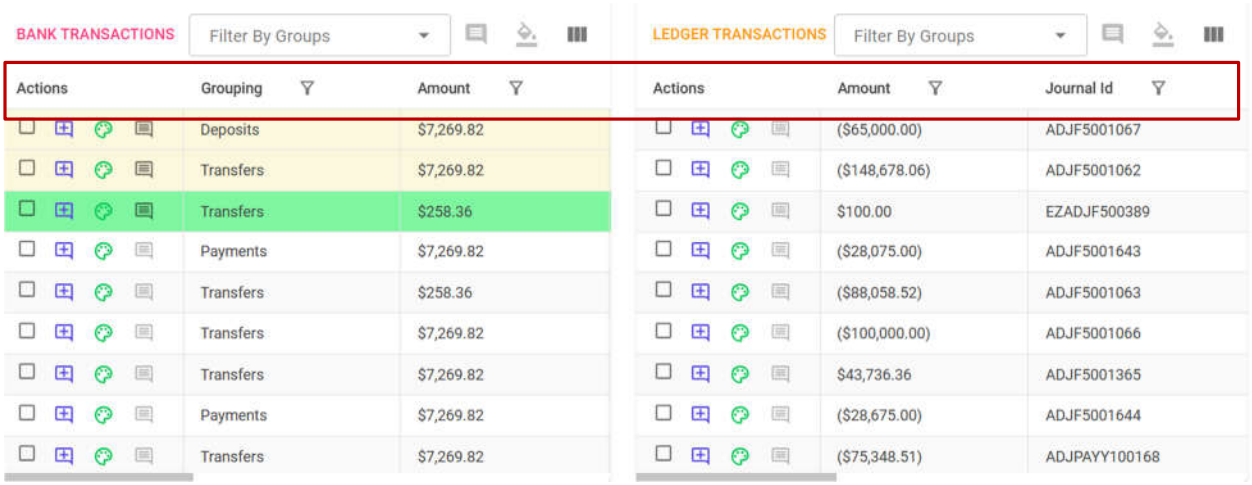


Change the columns being displayed. (Note. Your preference is saved and will affect all grids)

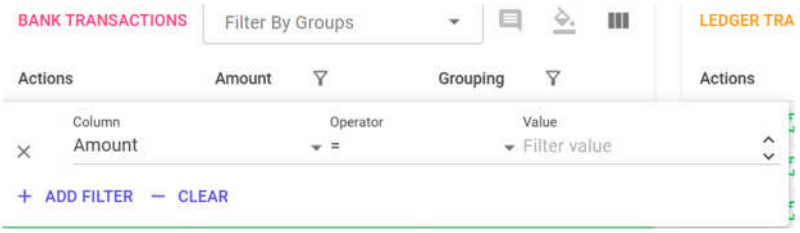
The column selection screens is as follows:



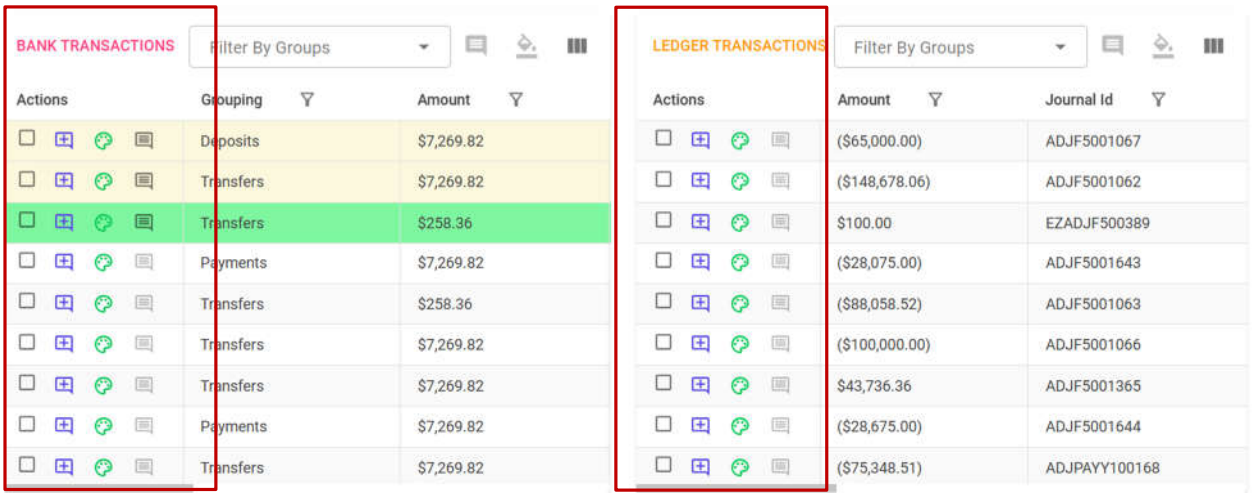
Next, let's look at the grid header.



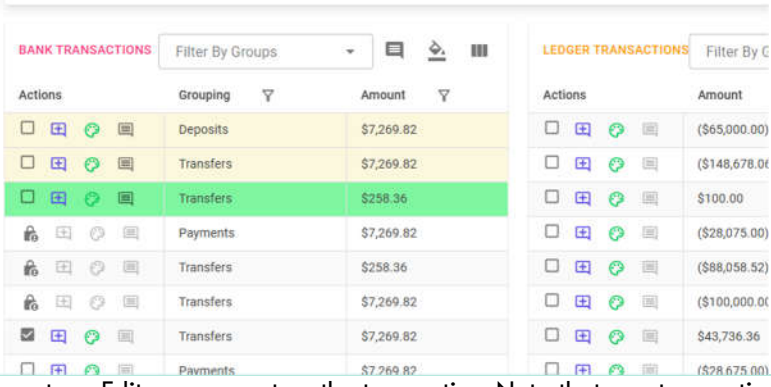
The header allows you to sort, filter and move columns around as you see fit. To sort, you can just click on the column name. There is robust filtering option as seen in the following image.

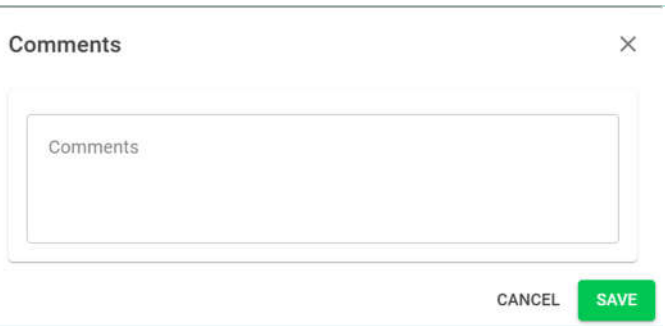


Next let's look at the row functionality.

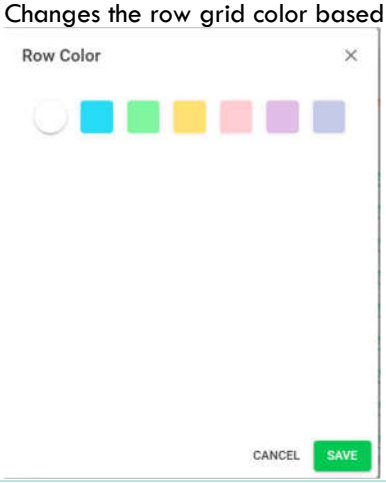


The Actions in the grid provide the following functionality.

Icon	Functionality
	If there are multiple users working on an account, the another user selects a record, the lock icon will show. You will not be able to select that record. If you hover your mouse over the icon it will show you who has locked the record. NOTE: locked records are cleared on log off. Also, an administrator can unlock items 
	Inserts or Edits a comment on the transaction. Note that any transaction that has a comment, the row color will change to a light yellow.

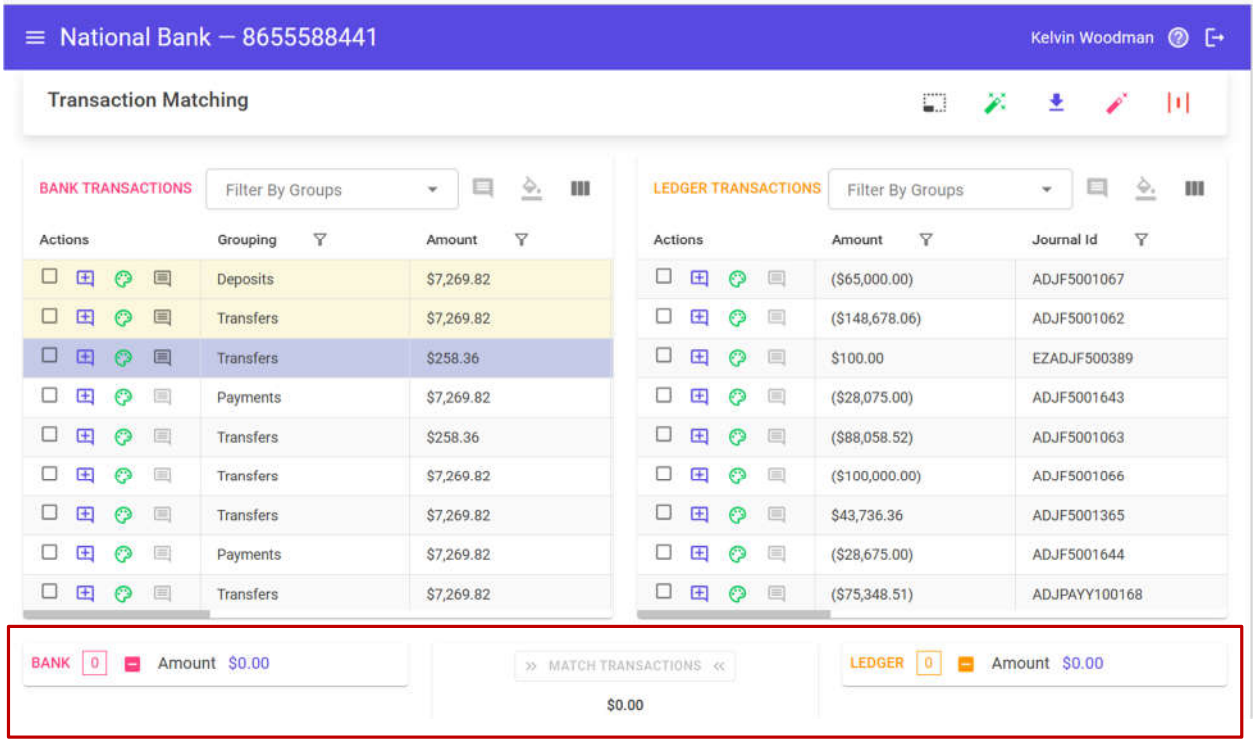


Changes the row grid color based on a preset list of colors



Hovering over this provides a quick view of the user comment on the transaction. (Note. This will be disabled if no comments are present)

Now let's look at the last area, the summary and transaction matching.



The number in the box shows how many transactions are selected.

The – button next to the "Amount" clears all selected transactions.

The Amount shows you the sum of the selected transactions.

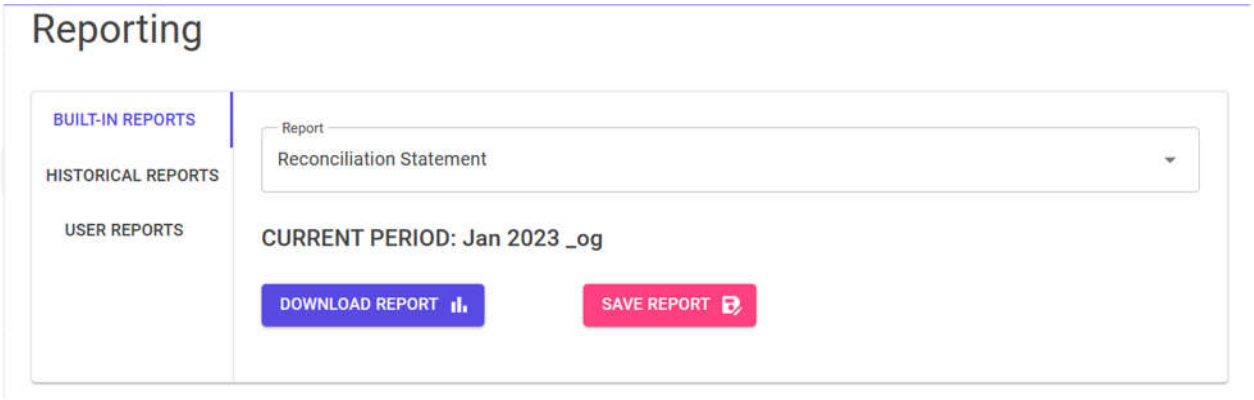
The "Match Transactions" button, allows you to match the transactions. The button will only be enabled when the sum of the selected transactions are zero. Once transactions are "matched", they are removed from the list.

Reporting

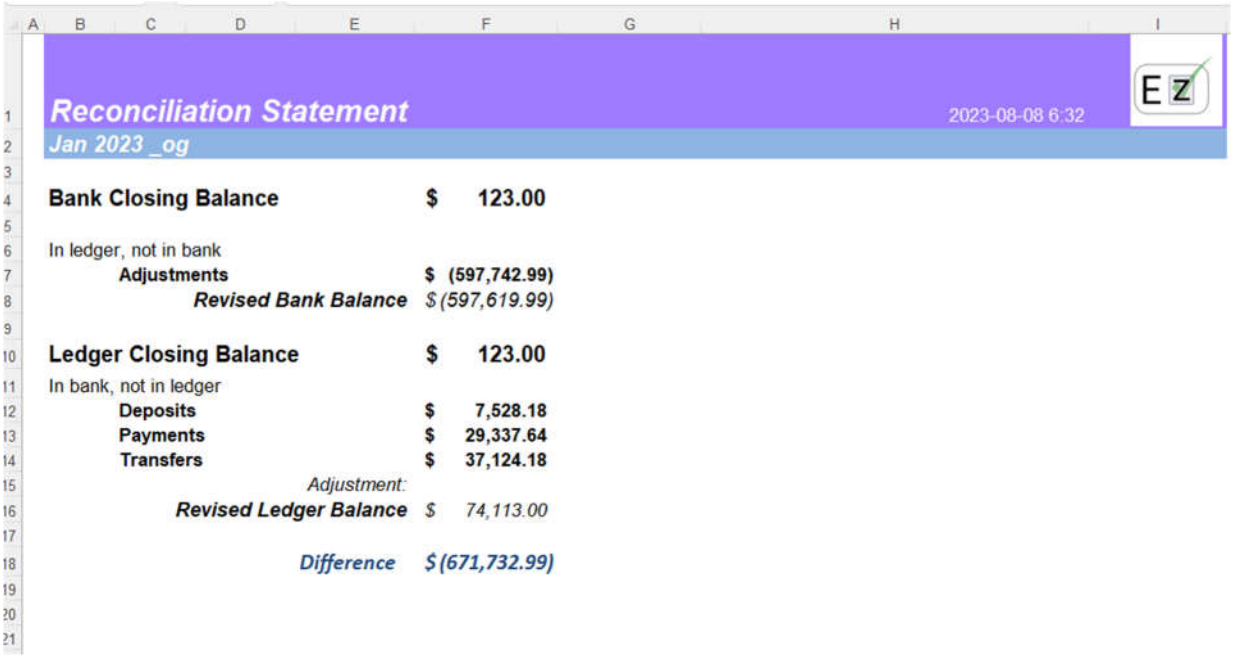
Select Reporting under the menu. This screen handles all the reporting needs of the program.

Under the reporting section, there are three options. Built-In Reports, Historical Reports and User Reports.

The Built-in Reports allows you to run the Reconciliation Statement, along with items that have not been reconciled. This is the only report that can be saved.



All Built-in reports are Excel based.



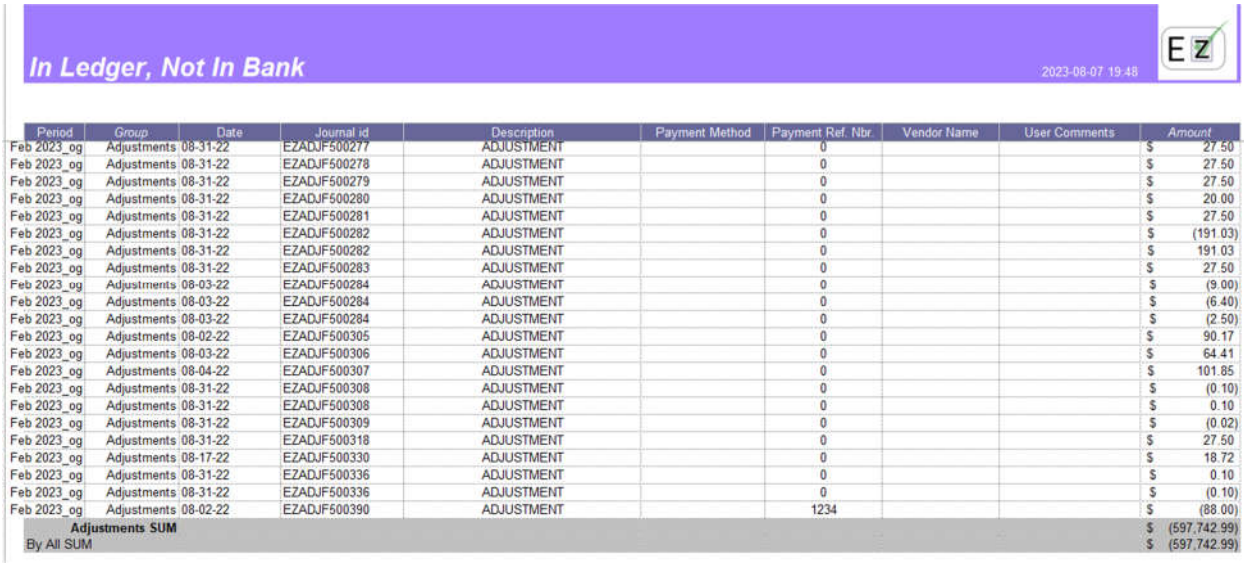
Reconciliation Statement

2023-08-08 6:32

Jan 2023 _og

Bank Closing Balance	**$**	**123.00**
In ledger, not in bank		
Adjustments	**$**	**(597,742.99)**
Revised Bank Balance		*$ (597,619.99)*
Ledger Closing Balance	**$**	**123.00**
In bank, not in ledger		
Deposits	**$**	**7,528.18**
Payments	**$**	**29,337.64**
Transfers	**$**	**37,124.18**
Adjustment:		
Revised Ledger Balance	*$*	*74,113.00*
Difference		***$ (671,732.99)***

Above is the reconciliation statement report.



In Ledger, Not In Bank

2023-08-07 19:48

Period	Group	Date	Journal id	Description	Payment Method	Payment Ref. Nbr.	Vendor Name	User Comments	Amount
Feb 2023_og	Adjustments	08-31-22	EZADJF500277	ADJUSTMENT		0			$ 27.50
Feb 2023_og	Adjustments	08-31-22	EZADJF500278	ADJUSTMENT		0			$ 27.50
Feb 2023_og	Adjustments	08-31-22	EZADJF500279	ADJUSTMENT		0			$ 27.50
Feb 2023_og	Adjustments	08-31-22	EZADJF500280	ADJUSTMENT		0			$ 20.00
Feb 2023_og	Adjustments	08-31-22	EZADJF500281	ADJUSTMENT		0			$ 27.50
Feb 2023_og	Adjustments	08-31-22	EZADJF500282	ADJUSTMENT		0			$ (191.03)
Feb 2023_og	Adjustments	08-31-22	EZADJF500282	ADJUSTMENT		0			$ 191.03
Feb 2023_og	Adjustments	08-31-22	EZADJF500283	ADJUSTMENT		0			$ 27.50
Feb 2023_og	Adjustments	08-03-22	EZADJF500284	ADJUSTMENT		0			$ (9.00)
Feb 2023_og	Adjustments	08-03-22	EZADJF500284	ADJUSTMENT		0			$ (6.40)
Feb 2023_og	Adjustments	08-03-22	EZADJF500284	ADJUSTMENT		0			$ (2.50)
Feb 2023_og	Adjustments	08-02-22	EZADJF500305	ADJUSTMENT		0			$ 90.17
Feb 2023_og	Adjustments	08-03-22	EZADJF500306	ADJUSTMENT		0			$ 64.41
Feb 2023_og	Adjustments	08-04-22	EZADJF500307	ADJUSTMENT		0			$ 101.85
Feb 2023_og	Adjustments	08-31-22	EZADJF500308	ADJUSTMENT		0			$ (0.10)
Feb 2023_og	Adjustments	08-31-22	EZADJF500308	ADJUSTMENT		0			$ 0.10
Feb 2023_og	Adjustments	08-31-22	EZADJF500309	ADJUSTMENT		0			$ (0.02)
Feb 2023_og	Adjustments	08-31-22	EZADJF500318	ADJUSTMENT		0			$ 27.50
Feb 2023_og	Adjustments	08-17-22	EZADJF500330	ADJUSTMENT		0			$ 18.72
Feb 2023_og	Adjustments	08-31-22	EZADJF500336	ADJUSTMENT		0			$ 0.10
Feb 2023_og	Adjustments	08-31-22	EZADJF500336	ADJUSTMENT		0			$ (0.10)
Feb 2023_og	Adjustments	08-02-22	EZADJF500390	ADJUSTMENT		1234			$ (88.00)
	Adjustments SUM								$ (597,742.99)
By All SUM									$ (597,742.99)

Above is a sample of the unreconciled items report.

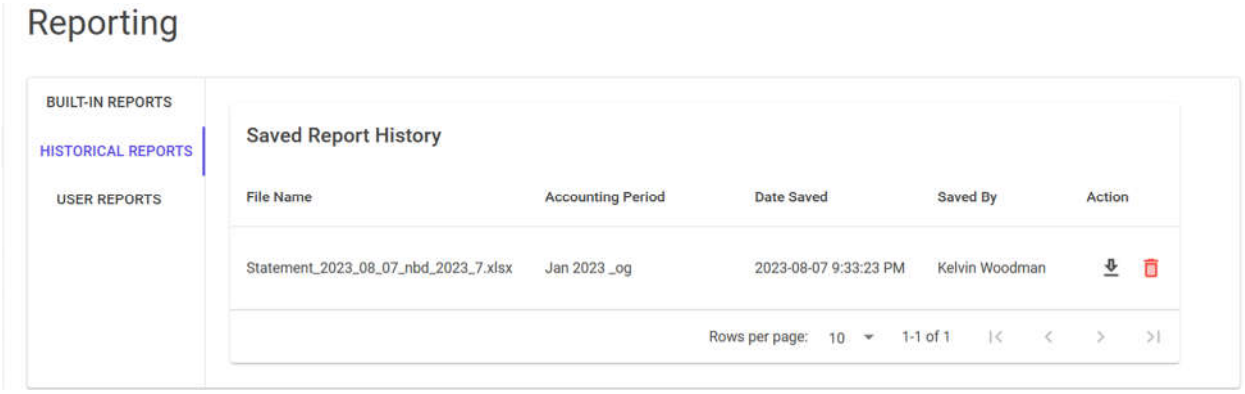


Transaction Verification 2023-08-19 9:50

	Bank				Ledger			
Period	Closing Balnce	Trans. Should Total	Trans. Summary	Result	Closing Balnce	Trans. Should Total	Trans. Summary	Result
Jan-23	$420.00	$420.00	$66,820.18	-$66,400.18	$45.00	$45.00	$66,820.18	-$66,775.18
Feb-23	$123.00	-$297.00	$7,269.82	-$7,566.82	$123.00	$78.00	$7,269.82	-$7,191.82
Mar-23	$500.00	$377.00	$0.00	$377.00	$400.00	$277.00	$0.00	$277.00
Apr-23	$150.00	-$350.00	$0.00	-$350.00	$200.00	-$200.00	$0.00	-$200.00

Above is a sample of the Transaction Verification Report

The Historical Reports contains the list of reports that were generated and were saved in the system. Only super users can delete reports that were saved.



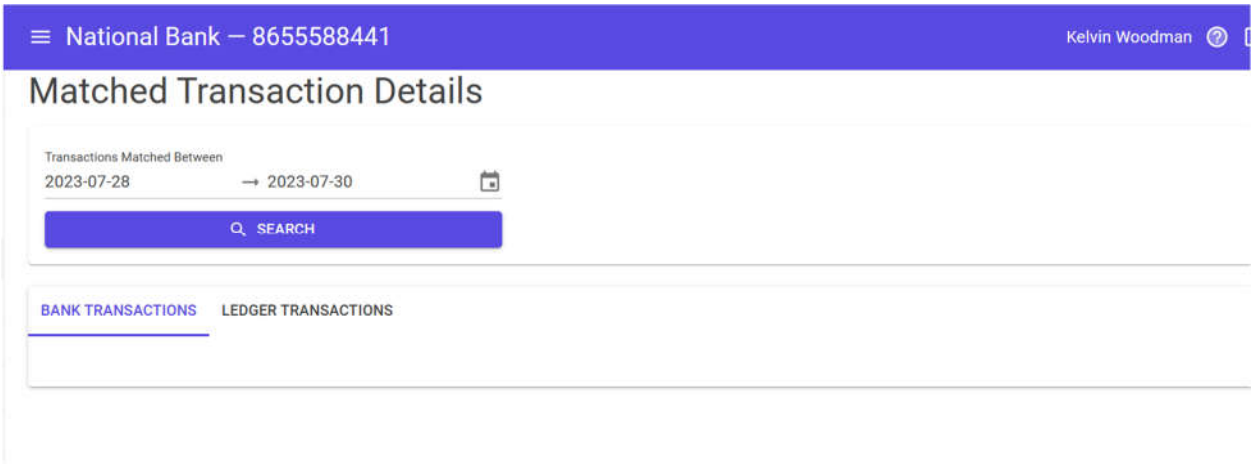
Only Reconciliation statement is available to save to the system.

The user reports contain any user generated report. These reports can be configured and they must be Sql Reporting Services.

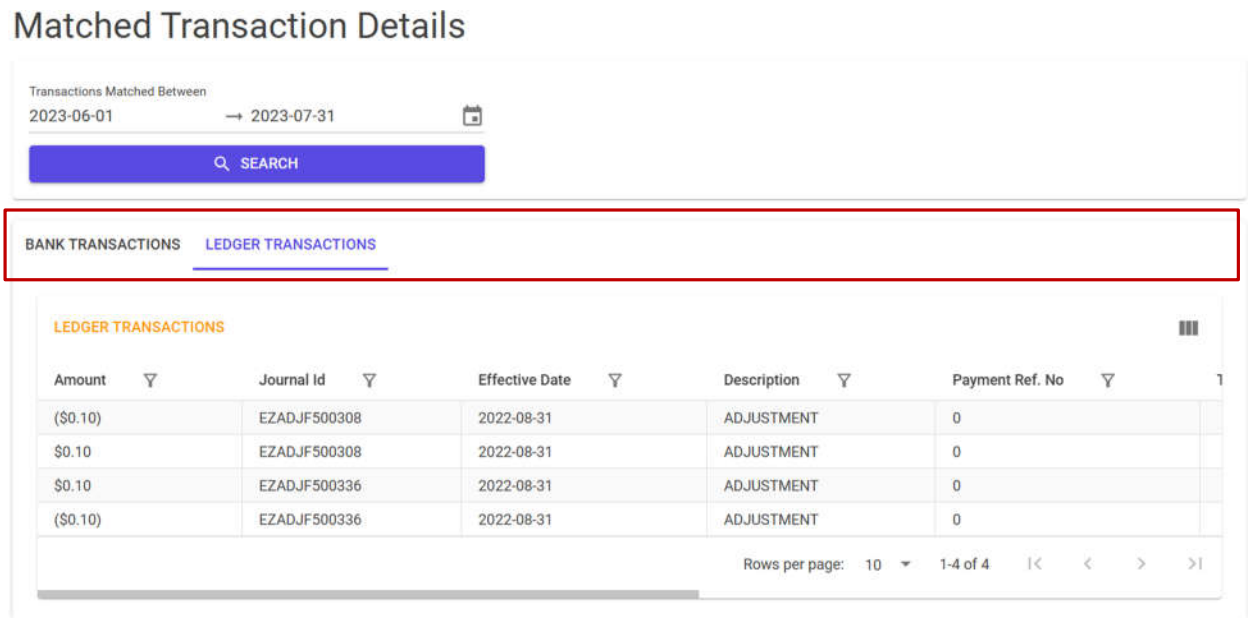


Matched Transaction Details

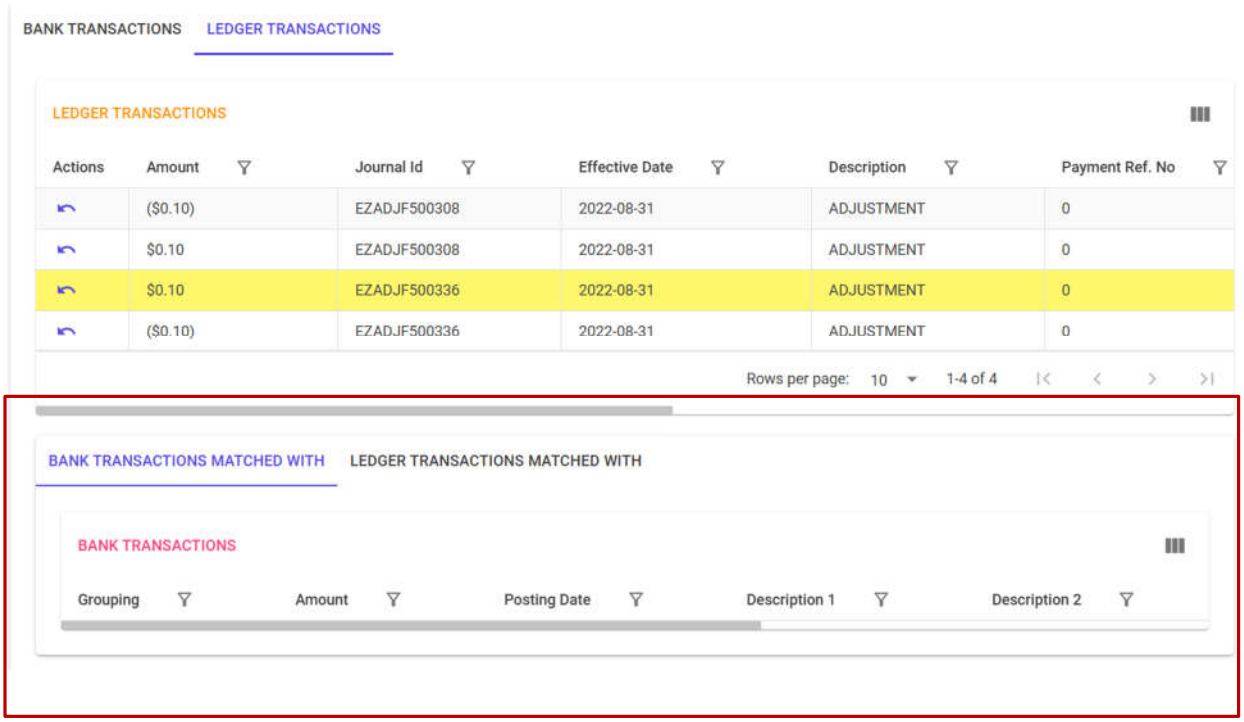
Select the Matched Trans. Details menu option.



This screen allows you to see the transaction which were matched between two given dates. Once the dates are entered, and the search button pressed, the system will list all transactions from both bank and ledger sides.



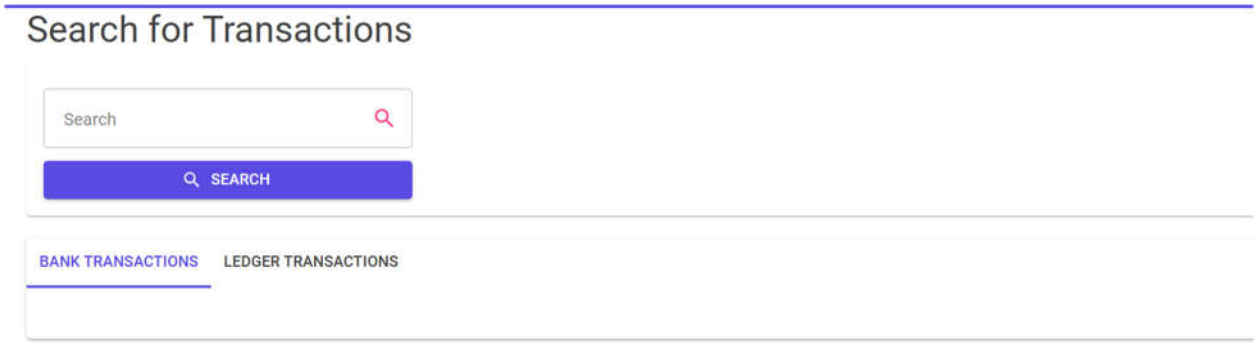
The tab BANK TRANSACTIONS & LEDGER TRANSACTIONS allows you to see the respective transactions that are within the date range. Once a transaction is selected, the transactions which were matched with selected transaction is displayed at the bottom of the tab.



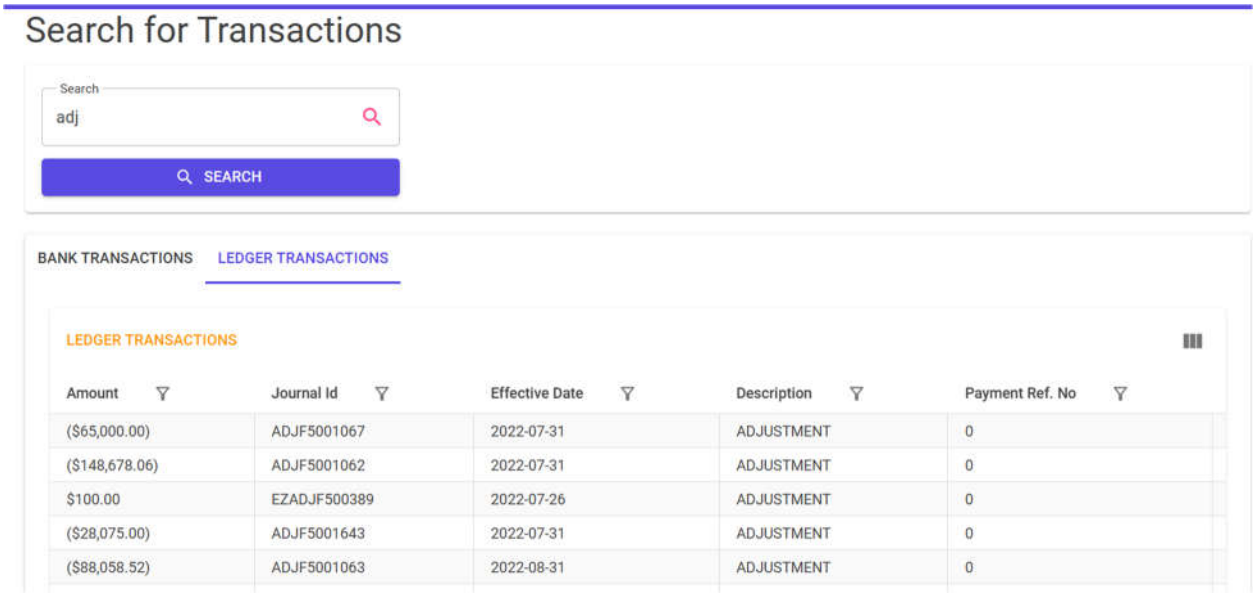
To un-match transactions. If your user is a Super User, you will see a button ( ) next to the transaction. Pressing this button will un-match the transaction and all other transactions matched along with that transaction if needed.

Search For Transactions

This screen allows you to search for a transaction. The system searches the description fields of the transactions. All matching transactions are returned.

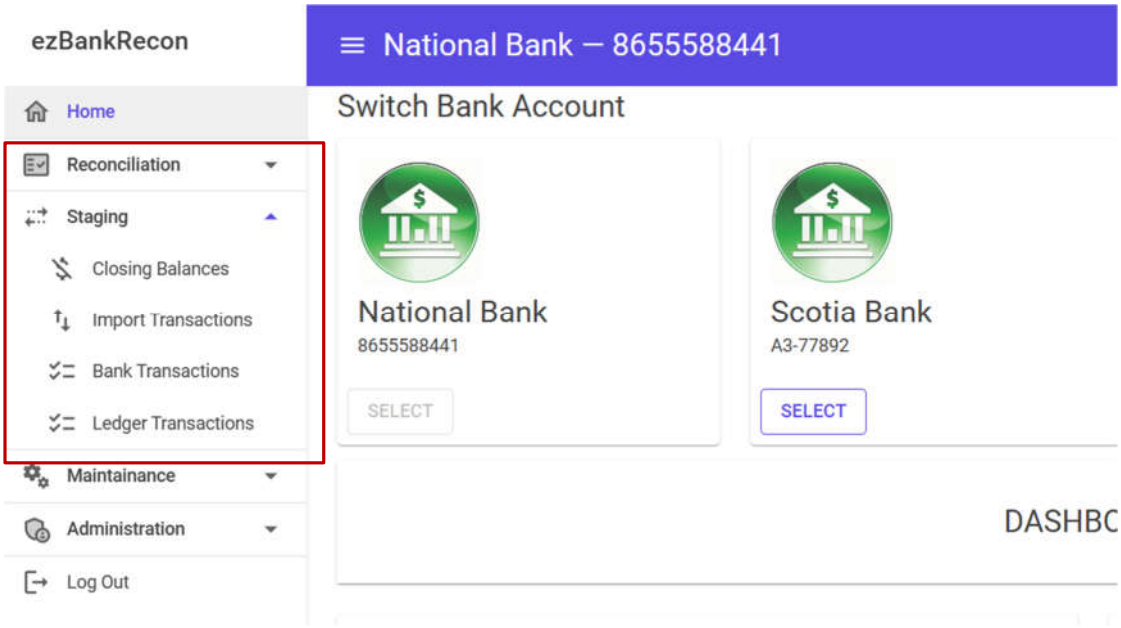


Enter a term and press the search button. The system will return all matching transactions on both bank and ledger side irrespective of whether or not they are matched.



Staging Area

Let's look at the Staging Menu. Before any transaction get's uploaded for processing, it goes to the staging area for "pre-processing"

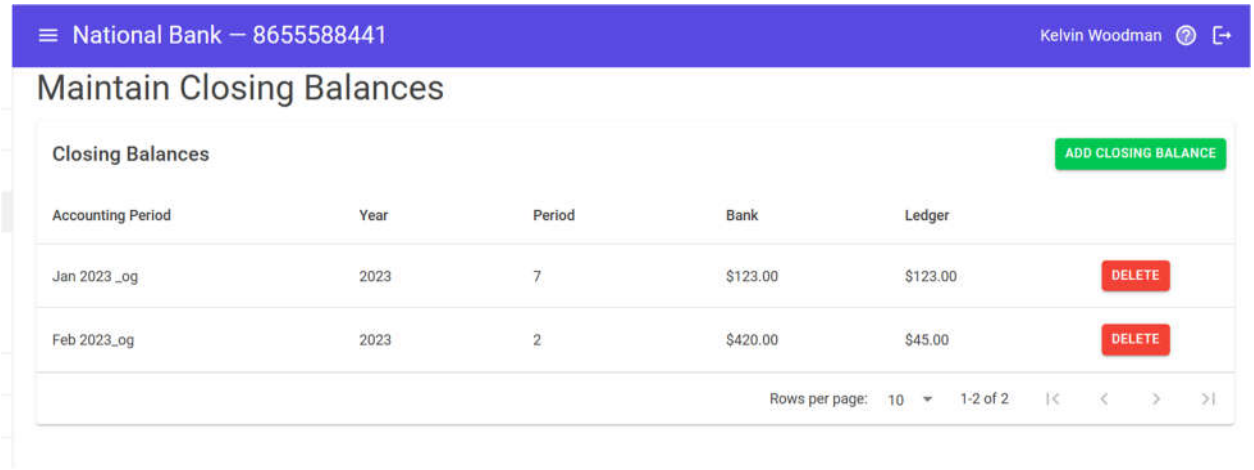


There are four menu items in the staging menu, which are available only to super users.

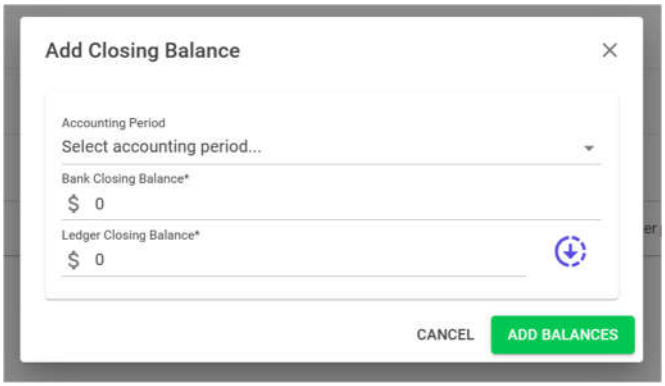
- Closing Balances
- Import Transactions
- Bank Transactions
- Ledger Transactions

Closing Balances

This screen is used to maintain the closing balances for the bank account and the ledger account. Balances cannot be edited once entered. However, they can be deleted. These balances are necessary as they will allow for the reconciliation to be accurate.



To add a new period closing balance, press the ADD CLOSING BALANCE button. You will be presented with the following screen:

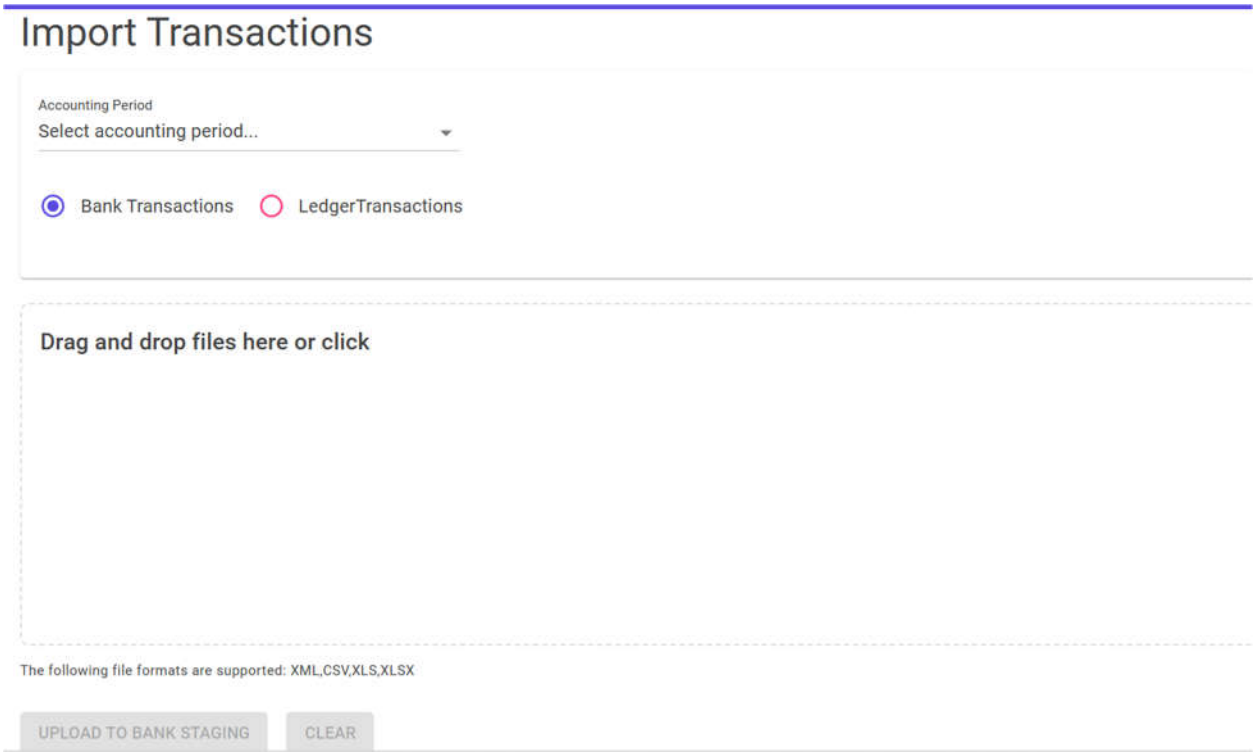


Select the accounting period. Enter the figure on the bank statement. You may enter the figure from the ledger or press the blue arrow button to load the balance directly from Smartstream.

Once done press the ADD BALANCES button to save.

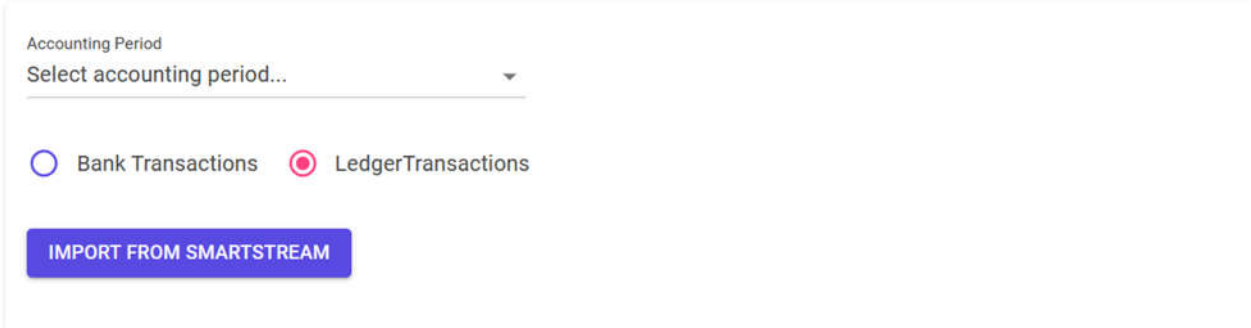
Import Transactions

This screen allows you to upload the file from the bank, from the ledger program or directly from Smartstream. It supports the following file types XML,CSV,XLS,XLSX.

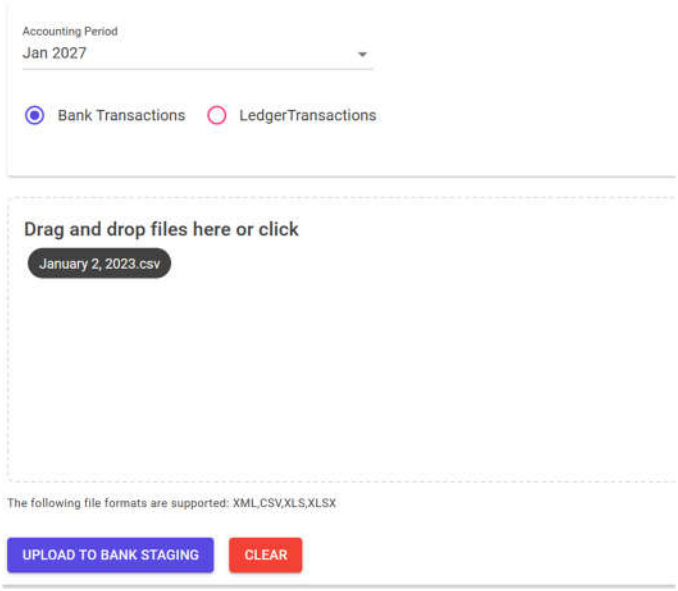


To begin, select the accounting period you are importing the transactions into.

Then select the type of transactions, whether bank or ledger. Note: if you select Ledger, you will have the option to import the transactions directly from Smartstream as shown below.



You may drag and drop the file into the drop zone or clock in the drop zone to open the file browser to select the file. Once the file is ready to be uploaded it will appear in the drop zone as follows and the buttons at the bottom of the screen will be enabled allowing you to upload the file or clear the file from the system.

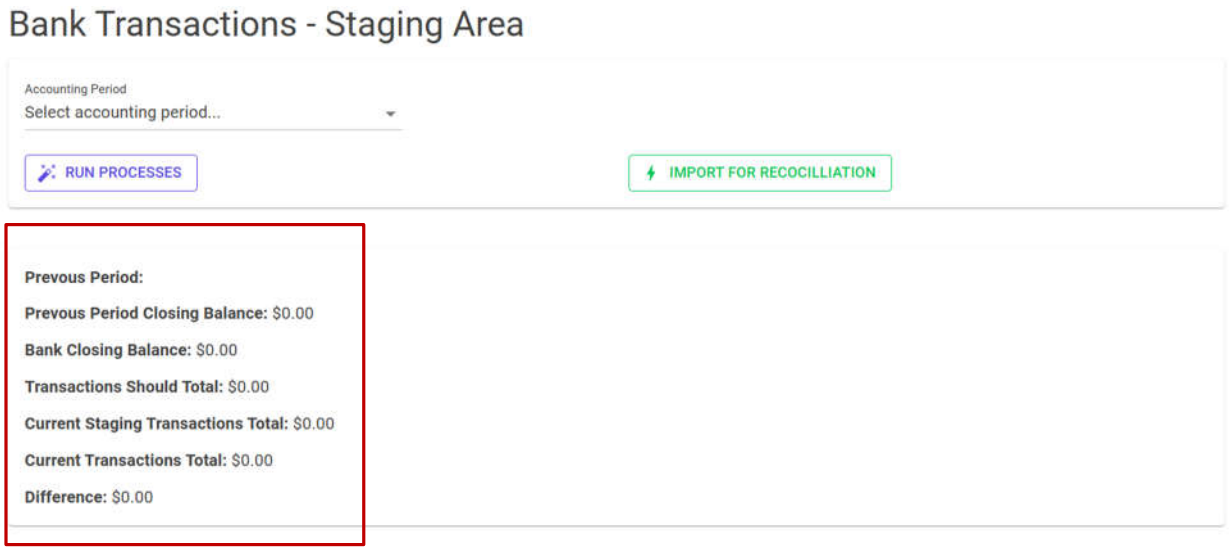


One the transactions are uploaded they are in the staging area, Bank Transactions and Ledger Transactions waiting for further processing.

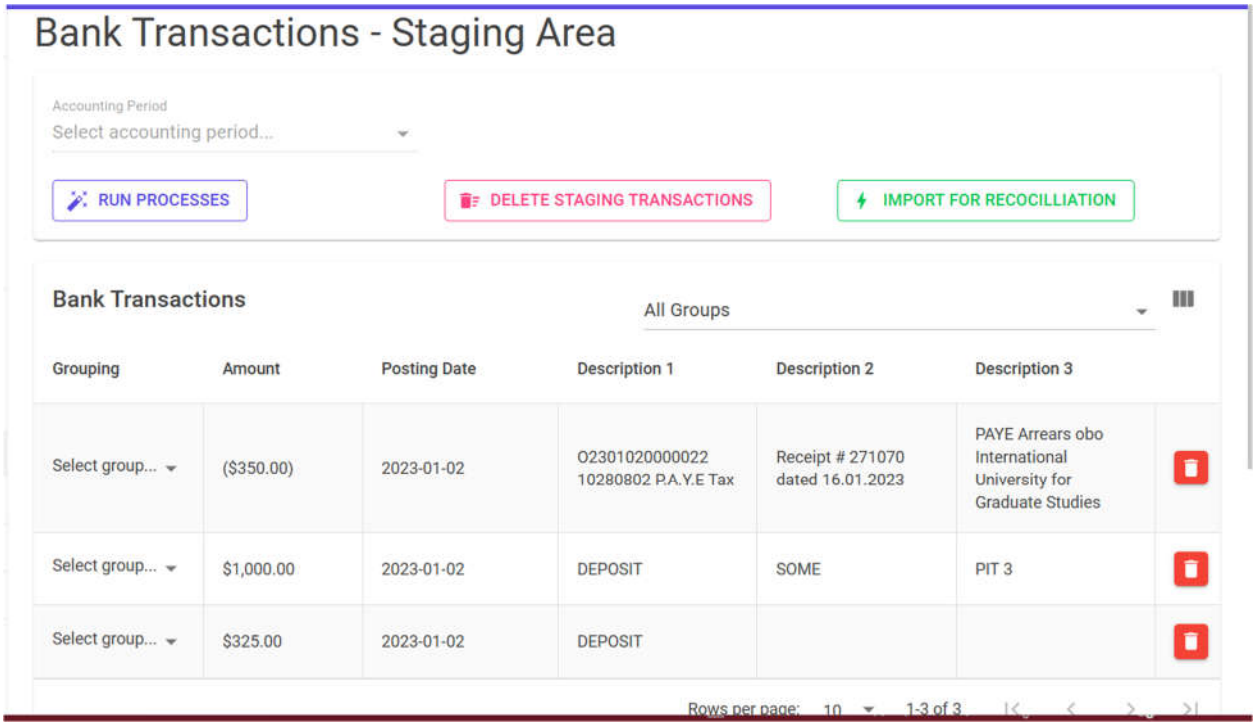
Bank and Ledger Staging Screens

These two screens are identical in function. They allow you to double check and verify the transactions before it moves into the work area. It also allows you to group the transactions.

Note: The accounting period will only be enabled if there are multiple periods in the staging area. i.e. if you imported files from two different periods (eg. You imported files from January 2021, and February 2021)



There are totals at the bottom of the screen to assist with the verification of transactions and balances.



The RUN PROCESSES button runs any custom process that may be necessary. For example, you may set up a process to convert the currency, or automatically set the grouping based on a field description.

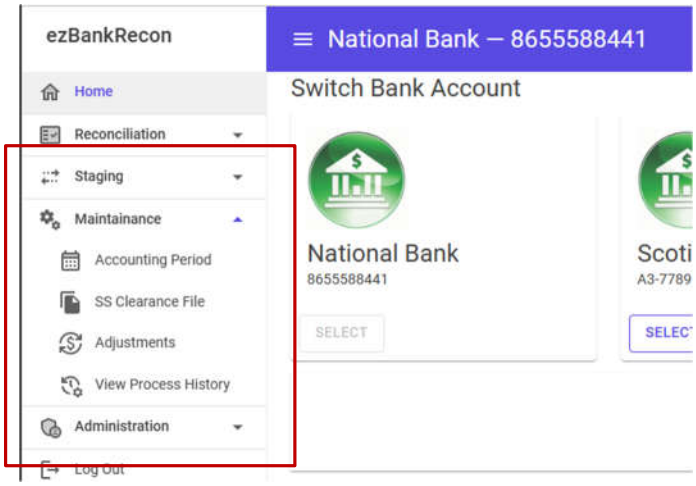
Once your verification is complete, press the IMPORT FOR RECONCILIATION button. This will import the transactions for reconciliation. NOTE. Any transaction without a grouping will not be imported. Also, note that this button will be disabled if the sum of the imported transactions match the closing balance for the period.

If transactions were imported into the wrong period, you may delete the transactions by pressing the button "DELETE STAGING TRANSACTIONS".

Individual transactions can be deleted by using the delete icon button in the transaction row.

Maintenance Area

Now we have the Maintenance menu. This menu option is only available to super users. It provides access to create the accounting periods, adjustments and view a partial history of what happened on the account.

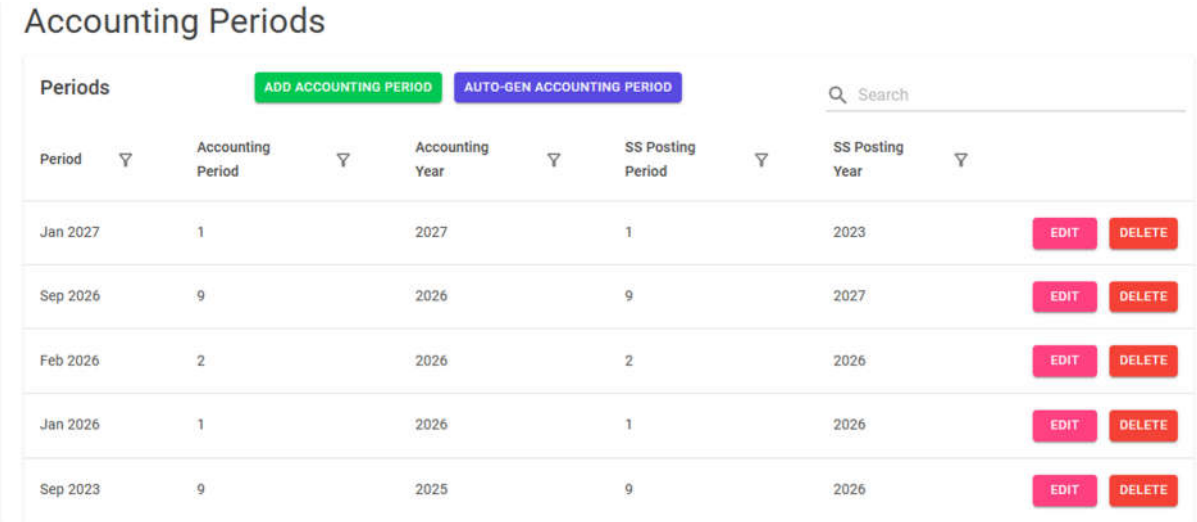


The following menu options are present:

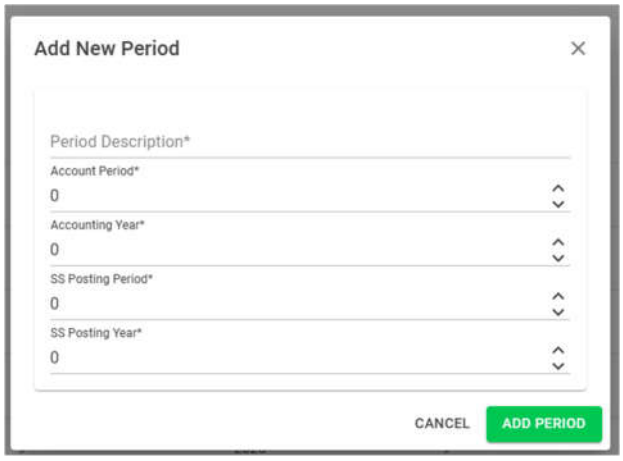
- Accounting Period
- SS Clearance File
- Adjustments
- View Process History

Accounting Period

The Accounting Period screen allows us to create new accounting periods for use in the system.



To create a new period, press the ADD ACCOUNTING PERIOD button. The system will then open the Add New Period dialog window.

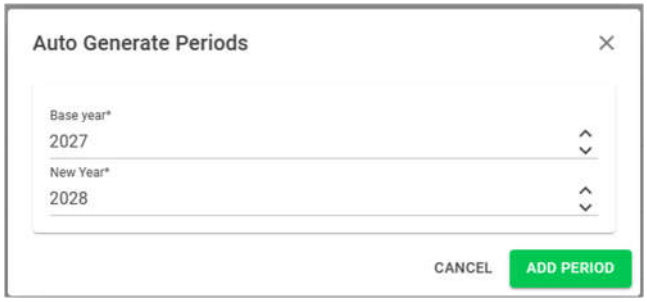


Enter a description of the period, eg. Jan 2023. Enter the account period which could represent the month and also the accounting period year. The fields SS Posting Period and SS Posting Year represent the Smartstream posting period and year respectively. (Due to different fiscal periods these do not correspond to calendar months) Once all the information is entered press the ADD PERIOD button to add the period.

To edit or delete a period, press the corresponding buttons on the grid next to the period you would like to edit/delete.

Once accounting periods for a year is entered, the system can generate new periods using a year as a basis.

Press the AUTO-GEN ACCOUNTING PERIOD button. The following dialog will open.



Specify the year to be used as a template and the year for which the periods should be generated. Once done click the ADD PERIOD button and the accounting periods will be generated for the year specified.

SS Clearance File

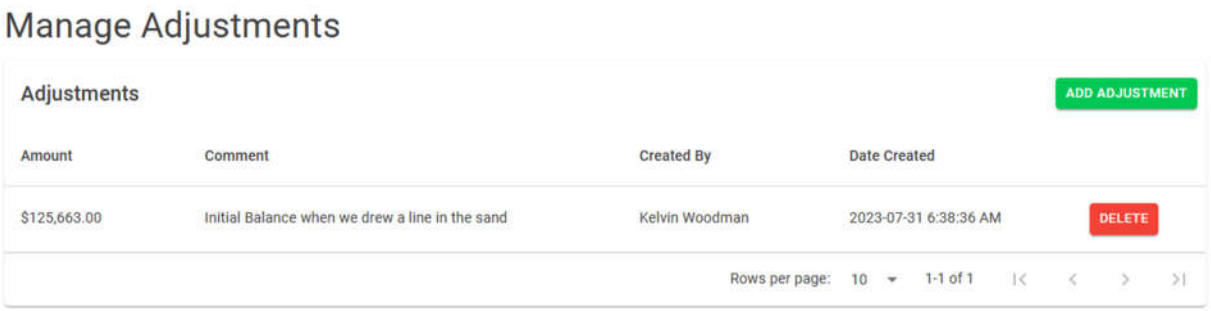
This screen is used to generate a file that clears the payments on the Smartstream system. This may or may not be necessary depending on your use of Smartstream.



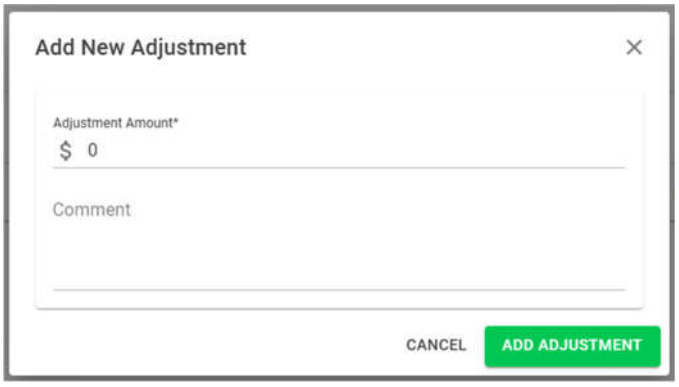
Press the GENERATE CLEARANCE FILE button and the system will produce a file containing the transactions which were matched for clearance in the Smartstream system.

Adjustments

This screen allows you to add any adjustments that are necessary for the reconciliation.



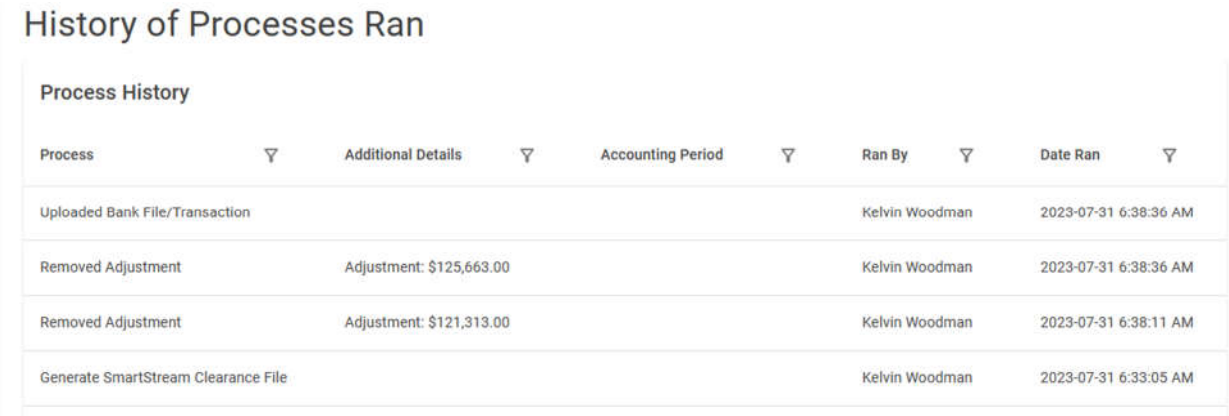
To add an adjustment, press the ADD ADJUSTMENT button. The following dialog will appear.



Enter the amount and a comment. Then press the ADD ADJUSTMENT button. Once pressed the adjustment will be registered on the system. Adjustments cannot be edited, only deleted. To delete an adjustment, press the DELETE button next to the adjustment you want to delete.

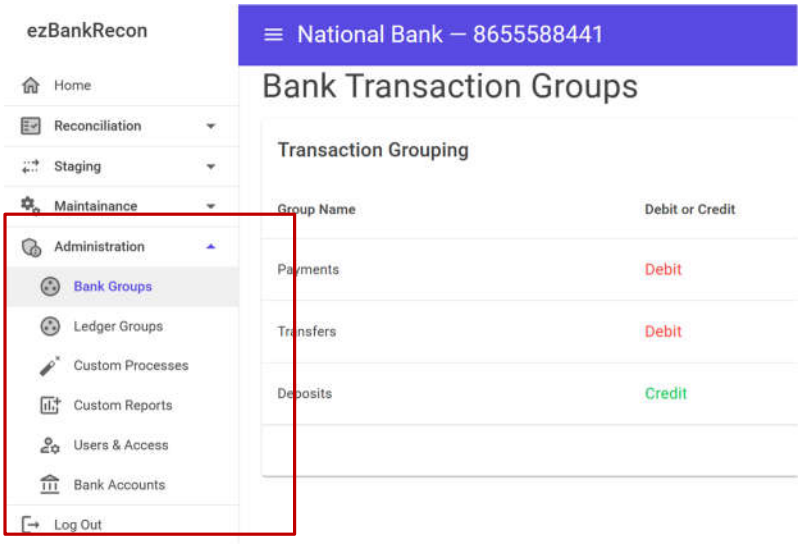
View Process History

This screen shows you the date and user who ran what the system considers significant processes, for example adding or removing adjustments.



Administrator Area

This section covers the menu items in the administrative area. There are various settings which will seldom change during regular operations and should be set up once the account is setup or requirements change

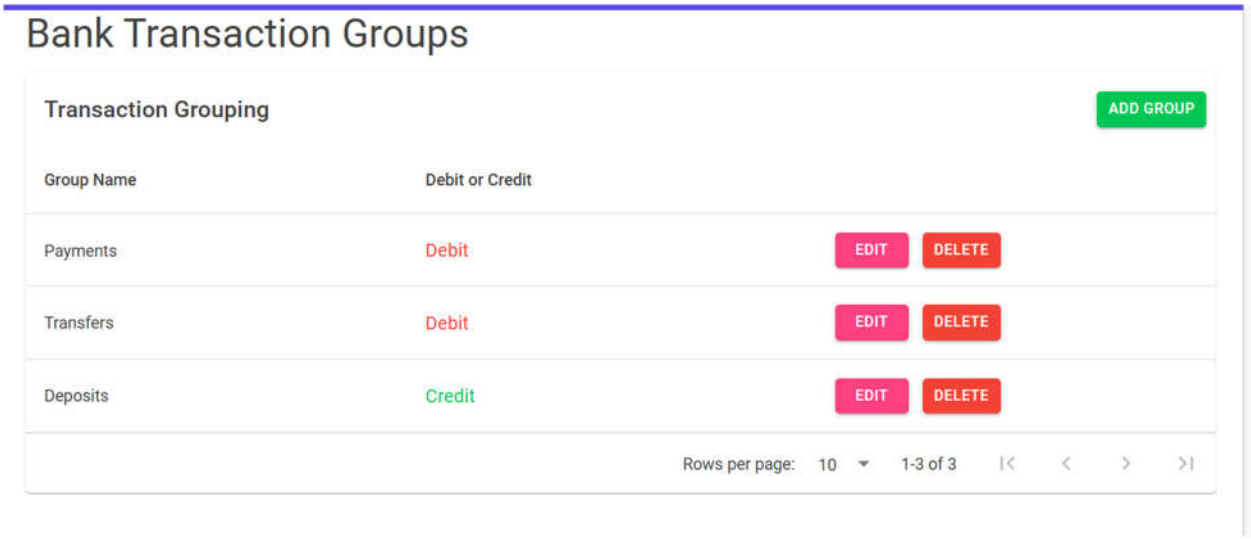


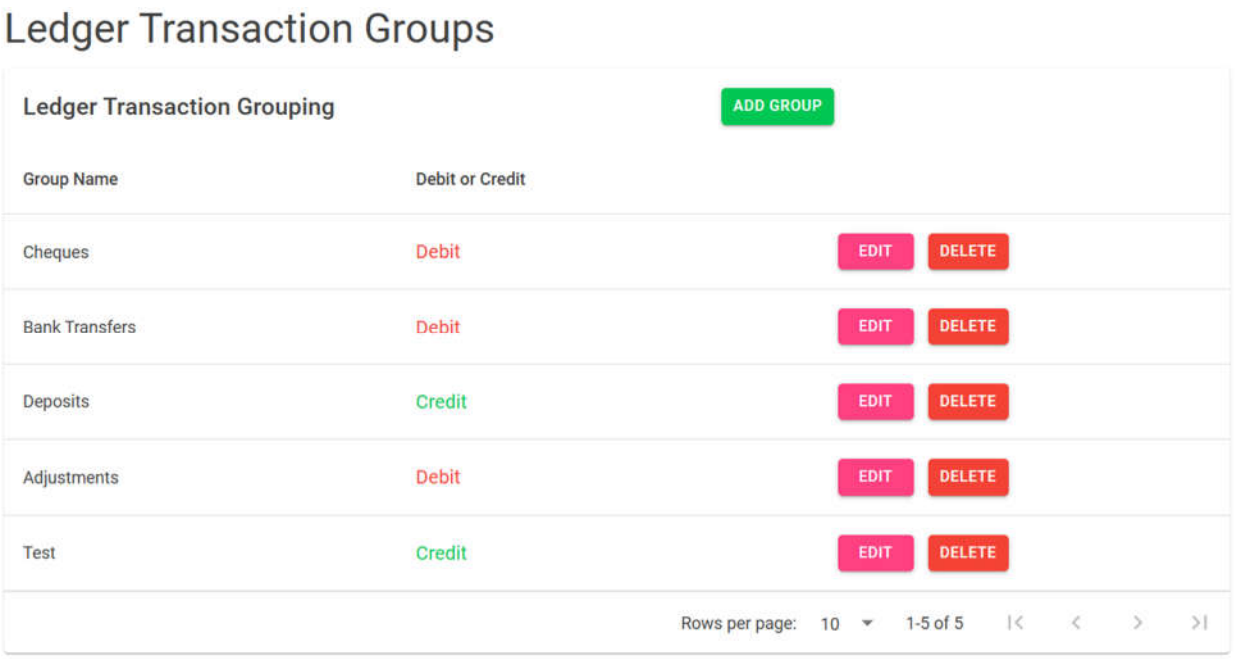
The menu items are as follows:

- Bank Groups
- Ledger Groups
- Custom Processes
- Custom Reports
- Users & Access
- Bank Accounts

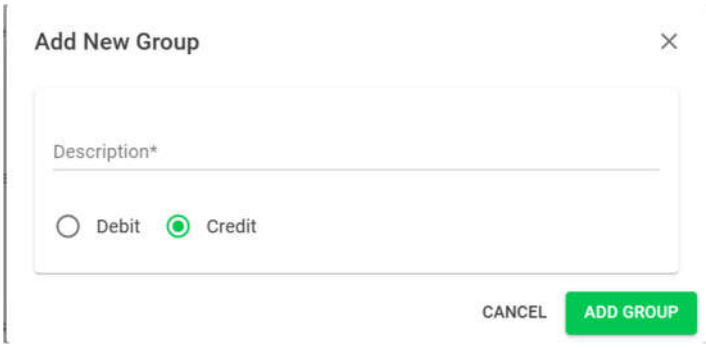
Bank and Ledger Groups

The Bank and Ledger Group screens allows you to create the classification that can be used for the Bank and ledger transactions respectfully. These classifications will be required for reporting and other processes. It is best to decide on the classification before you begin using the system. The system allows you to go as detailed or as generic as you would like. For example, you may just have debits and credits, or break them further into checks, bank transfers, adjustments etc.





To add a new group, press the ADD GROUP button. The following dialogue will open.

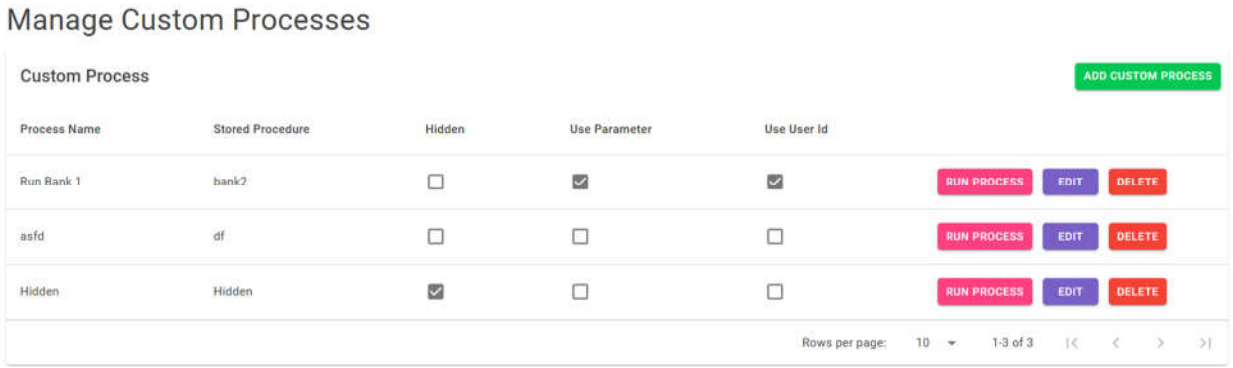


Enter the description of the group that you would like to create and select whether it goes in the debit or credit category. Click the ADD GROUP Button to save the new group.

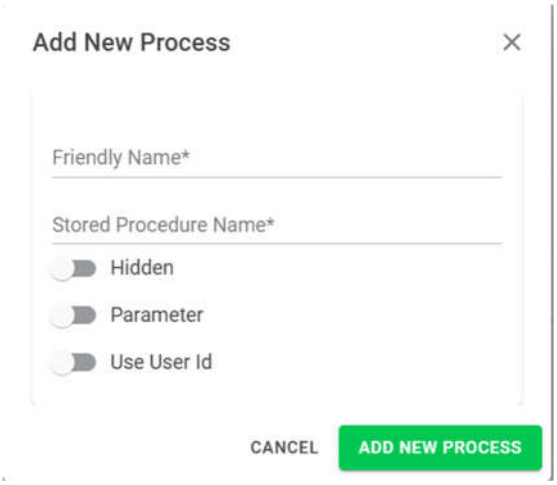
Note. Your administrator may have to configure the stored procedures

Custom Processes

The system allows you to create stored packages to run on your data. For example, you may need a function to split the description given by the bank. This screen allows you to add the function/procedure so that the end user can run the function after it is created.



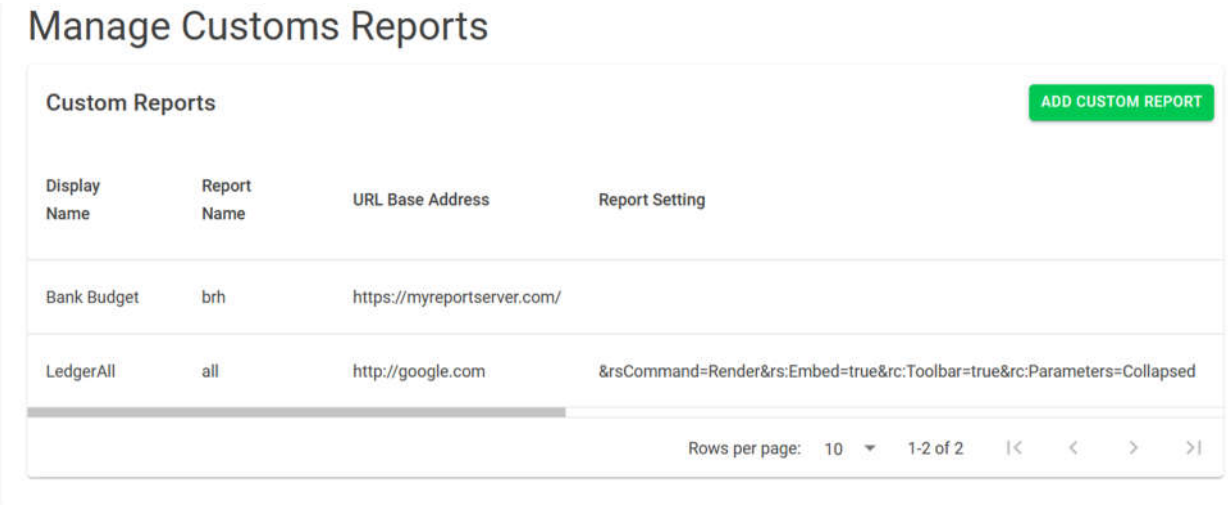
To add a custom process, first make sure that it is added to the database under the correct account schema. Next use this screen to create a link to run the process. Click the ADD CUSTOM PROCESS button and the following dialog will open.



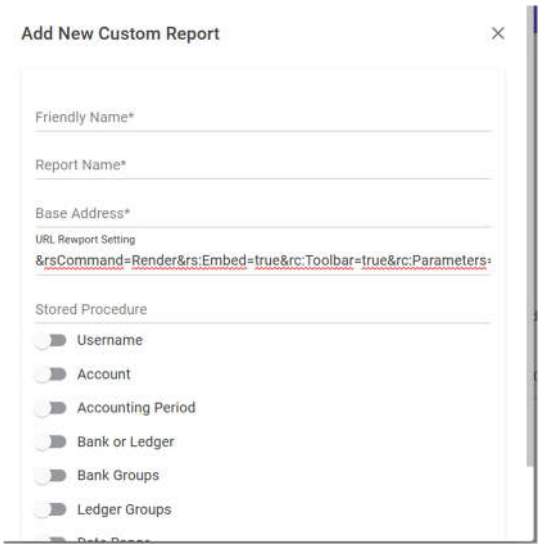
Enter a friendly name for the end user, and enter the actual name of the stored procedure. Hit the ADD CUSTOM PROCESS button to save. The custom process will now show up as an option on the Match Transactions screen.

Custom Reports

The system has the ability to link to a Microsoft SQL Server Reporting Service. Therefore, any reports that are needed can be added to the system. For example, there may be a need for custom branding or report that pulls data from outside the system.



To enter a custom report, access the custom reports screen. Press the ADD CUSTOM REPORT button whereby the following dialog will open.

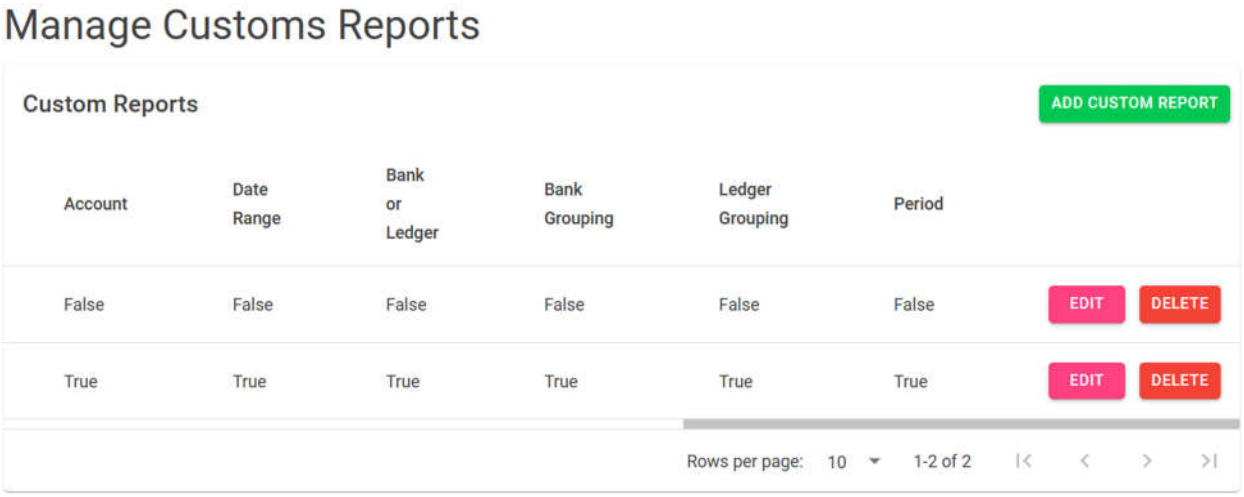


Enter the information and then press the ADD REPORT button. The report will now be available for selection in the Reporting section.

Field	Description
Friendly Name	The name displayed for the user
Report Name	The actual name of the report
Base Address	The url of the report server

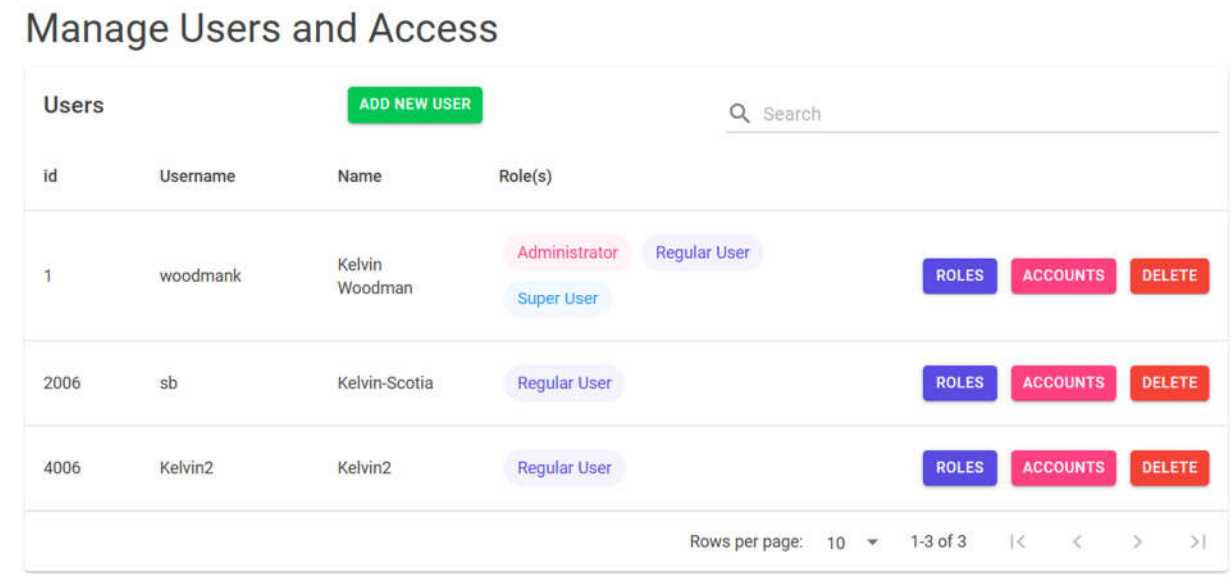
Url Report Settings	The tail portion of the url. Usually standard but can be customized
Stored Procedure	The name of the stored procedure to run before running the report
User Name	Pass the username to the report or not
Account	Pass the account to the report or not
Accounting Period	Display the accounting period selector
Bank or Ledger	Display the bank or ledger selector
Bank Group	Show the bank group selector
Ledger Group	Show the ledger group selector
Date Range	Show the date range selector

To Edit or Delete a report, press the corresponding button to the far right of the grid next to the report you would like to edit or delete.



Users and Access

This screen is used to manage access and security of the system. Users of the system need to have an active directory account before they are able to use the software. Once an active directory account is created, you can proceed to add the user to the system.



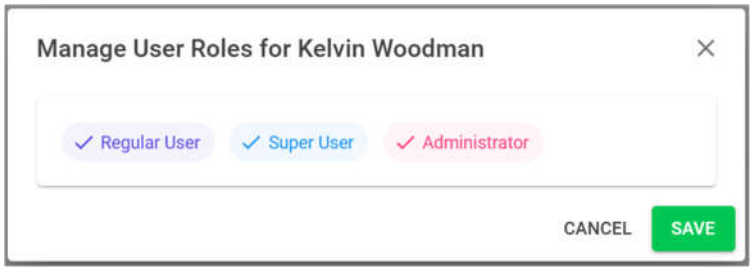
To add a new user, press the ADD NEW USER button. The following dialog will open



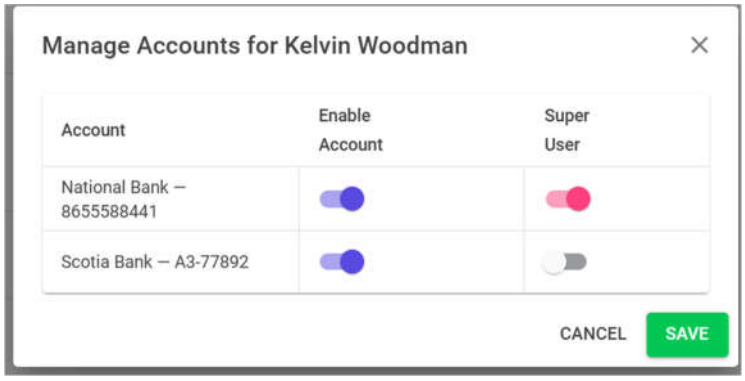
Enter the name of the user and the active directory user name. Click the ADD USER Button to add the user. Once the user is added, you now need to grant the user a role. The roles on the system are:

Role	Description
Regular user	Has access to Reconciliation Area
Super User	Has access to Staging Area
Administrator	Has access to Administrative Area

Press the ROLE button next to the user to grant the user a role. Select the role or roles that you would like to grant the user. Then click the SAVE button.



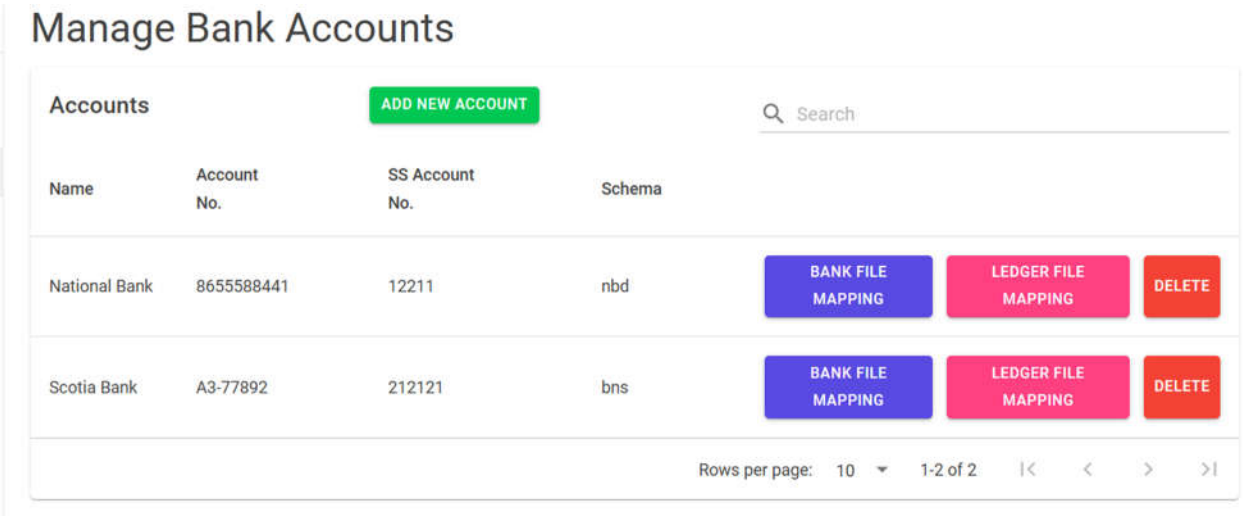
After roles are selected, the user will need to be granted access to an account. To add a user to an account select the ACCOUNTS button next to the user. The following dialog will open.



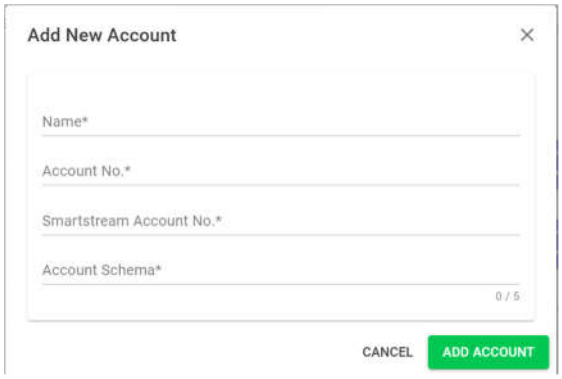
The dialog will list all available accounts. Enable the account that you would like to give the user access to. If the user is a super user for the account, enable the super user field. Click the SAVE button when done.

Bank Accounts

The Bank Account screen maintains the different accounts that the system will reconcile.



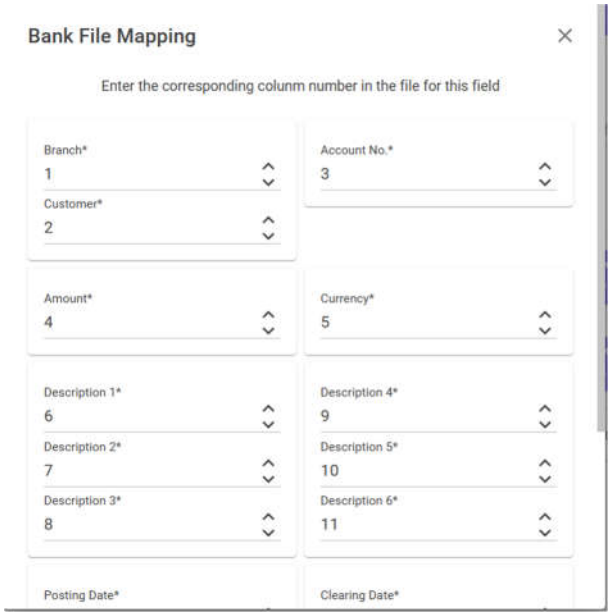
Click the ADD NEW ACCOUNT button to add a new account to reconcile to the system. Once, he button is pressed the following dialog box will open.



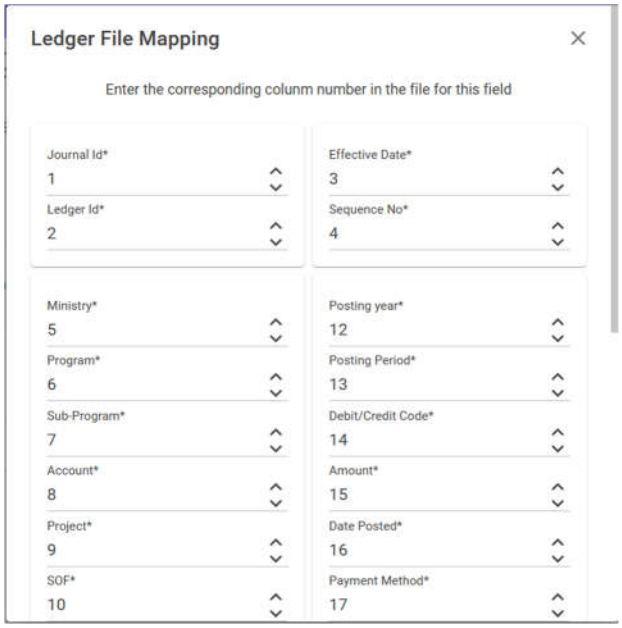
Enter the account Name, number, Smartstream account number (this allows the system to pull information from the Smartstream system) and a unique combination for the schema. The schema is the database schema on which all the tables and procedures will be created under.

Once done press the ADD ACCOUNT button. The new account will be created along with the database tables.

The next matter to configure with the account is the Bank File Mapping and Ledger File Mapping. This maps the corresponding columns on the to the fields when doing file uploads into the system.



You will not need to change anything here unless the file format from the bank has changed. If the file format from the Bank has changed, simply enter the corresponding column number with the corresponding field. Then press the save button.



The same goes for the Ledger file mapping. You would rarely, if ever need to use ledger file upload as the system should pull the information straight from the Smartstream system. Make sure that the column number matches the corresponding column.

If an account is no longer needed it may be deleted by pressing the DELETE button next to the account. Once the delete button is pressed a confirmation dialogue is presented to the user. If you choose to continue, you will also have to decide if you would like to keep the database objects that were created or not.

Advanced Configuration

Database Overview

The database for this program changes with every account added or removed to the system. Tables with the "dbo" prefix are used by the main program. Any account specific tables are prefix by the "schema" value entered when the account was created.

The main tables are:

- [dbo].[account_bank_file_mapping]
- [dbo].[account_ledger_file_mapping]
- [dbo].[accounts]
- [dbo].[processes]
- [dbo].[roles]
- [dbo].[user_accounts]
- [dbo].[user_preferences]
- [dbo].[user_roles]
- [dbo].[users]

Account Related Tables

Each account has tables, views and procedures associated with it. The account tables is as follows:

- {schema}.[accounting_periods]
- {schema}.[adjustments]
- {schema}.[bank_transaction_groups]
- {schema}.[bank_transactions]
- {schema}.[bank_transactions_staging]
- {schema}.[closing_balances]
- {schema}.[custom_processes]
- {schema}.[custom_reports]
- {schema}.[ledger_transaction_groups]
- {schema}.[ledger_transactions]
- {schema}.[ledger_transactions_staging]
- {schema}.[process_logs]
- {schema}.[reports_history]
- {schema}.[user_bank_columns]
- {schema}.[user_ledger_columns]

Stored Procedures

Each account schema also contains the following stored procedures:

- {schema}.[bank_staging_process]
- {schema}.[generate_ss_fILe]
- {schema}.[import_ledger_transactions]
- {schema}.[ledger_staging_process]
- {schema}.[matching_process]

Also 3 database view:

- {schema}.[ledger_closing_balance_vw]
- {schema}.[ledger_trans_sum_vw]
- {schema}.[ledger_trans_sum_vw]

Application Settings

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Default": "Data Source=WOODZONE;Initial Catalog=ezBankRecon;Integrated
Security=True;TrustServerCertificate=true",
    "Domain": "gocd.mofpdomgov.gov"
  },
  "SQLLinkServer": "gocdssp",
  "BaseUrl": "/",
  "TimeOut": "1200"
}
```

Setting	Description
Default	The connection string to the database
Domain	The domain that hat the users authenticate with
SQLLinkServer	The name of the link server created on the database that links to the smartstream server.
BaseUrl	Specifies the base url. Useful if running the app from a subdomain. Eg. “http://gov.dm/”

TECHNICAL INFORMATION

Technologies & Resources

.Net Blazor

The web program was built using the latest C# and Blazor web technologies. C# is a **modern, innovative, open-source, cross-platform** object-oriented programming language. And is used from enterprise development.

https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor

MudBlazor

Mudblazor is an opensource UI framework. It provides the various components which provide a consistent experience for the end user. It follows the Google Material Design system.

https://mudblazor.com/

Closedxml

ClosedXML is a .NET library for reading, manipulating and writing Excel 2007+ (.xlsx, .xlsm) files. It is used in the program to generate excel files.

https://docs.closedxml.io/en/latest/

Closedxml.Reports

ClosedXML.Report is a tool for report generation and data analysis in .NET applications through the use of Microsoft Excel. It is used in the program in the generation of standard reports.

https://closedxml.io/ClosedXML.Report/

Dapper

Dapper is a simple object mapper for .NET. It is used in the program to connect to the SQL Server database.

https://dapperlib.github.io/Dapper/

Signal R

SignalR allows server side code to push updates to connected clients (primarily over WebSockets) as soon as new content/information is available, without any need for clients to constantly poll the server for updates. It is used in the program to update and sync the transactions across different sessions for the users.

https://dotnet.microsoft.com/en-us/apps/aspnet/signalr

Excel Data Reader

This is a lightweight and fast library written in C# for reading Microsoft Excel files. It is used in the program for processing the uploaded files.

https://github.com/ExcelDataReader/ExcelDataReader

Microsoft SQL Server

Microsoft SQL Server is used to power the back end database in the application.

https://www.microsoft.com/en-us/sql-server

Installation

Requirements

The program runs on Microsoft Technologies. Due to .Net Core, it can run standalone if so desired. However, the supplied compile version was compiled to run on Microsoft ISS on windows sever.

Before you install on the windows server, please ensure that the .Net SDK is installed on the server.

You can download the SDK at

https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/sdk-7.0.305-windows-x64-installer

For more information please check out the following website.

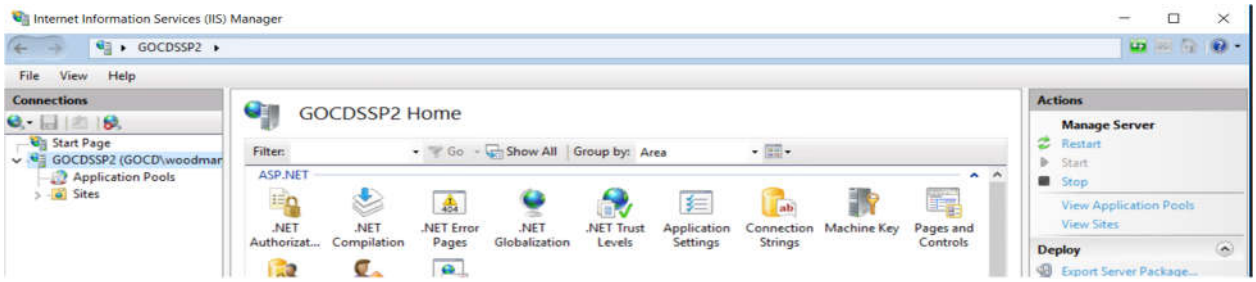
https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/?view=aspnetcore-7.0#iis-configuration

Once the SDK is installed, you may use the WEB Deploy functionality to install the software. The software may be installed on it's own domain or part of a domain.

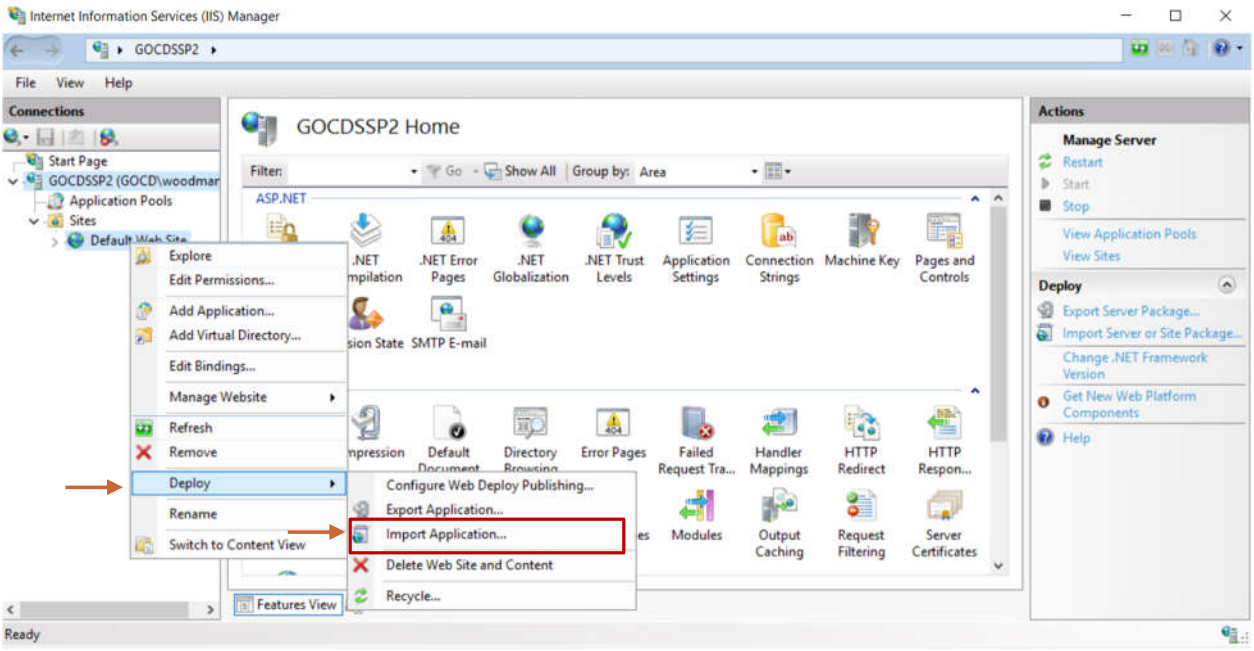
The application uses https. This provides secure communication. Therefore, a SSL certificate needs to be installed. The program will work without an SSL certificate. You may create a self signed certificate or get a one from a certificate authority.

Sample Installation

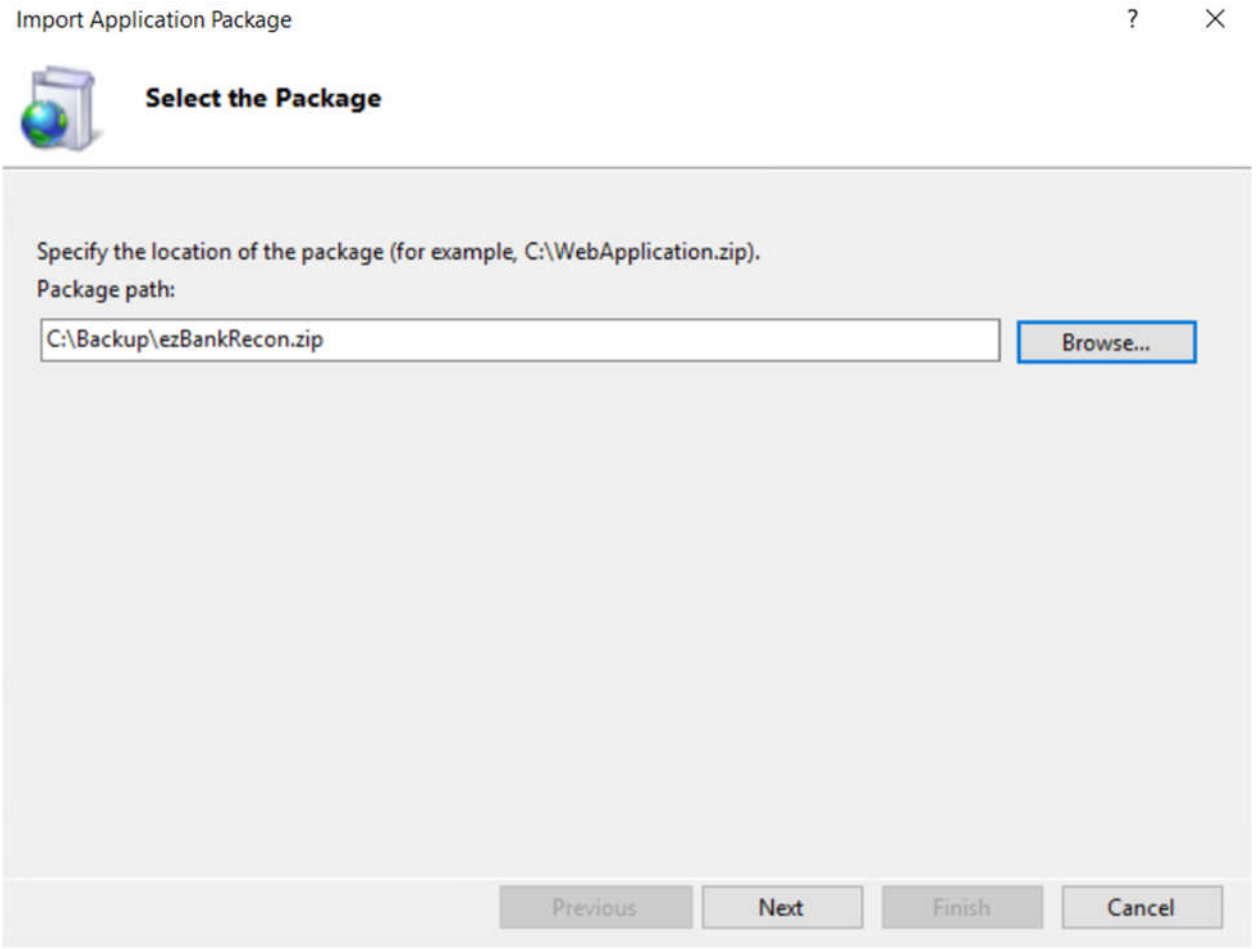
1. Open IIS



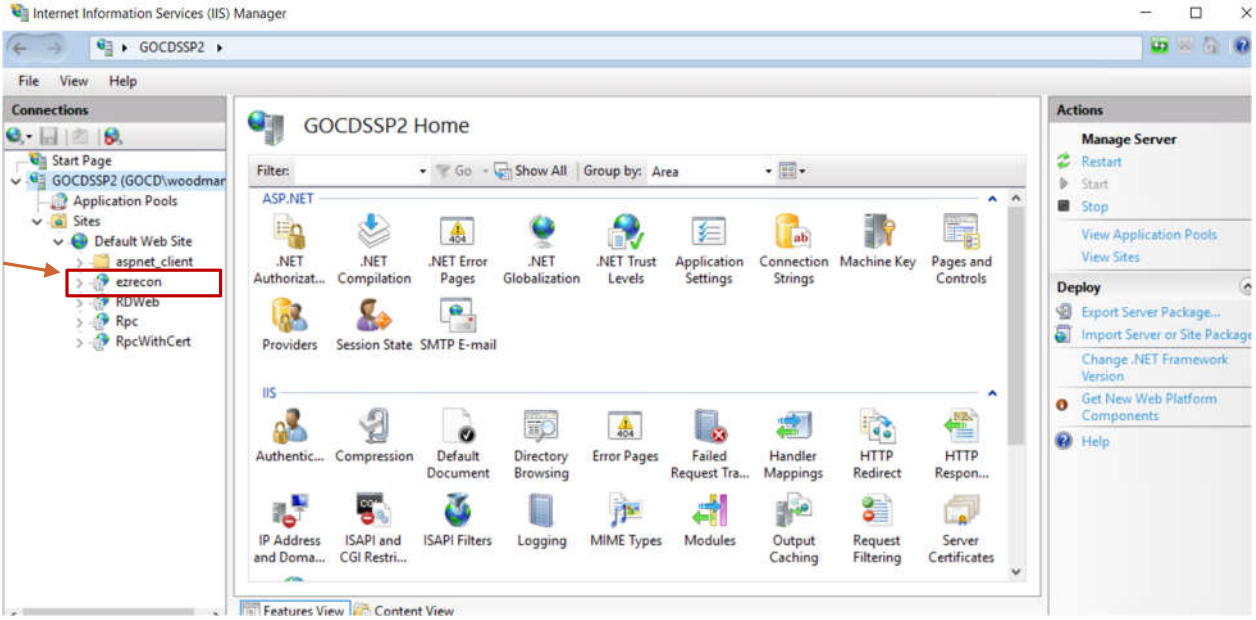
2. Select the site you would like to deploy the application to. Right Click. Select Deploy and then Import Application



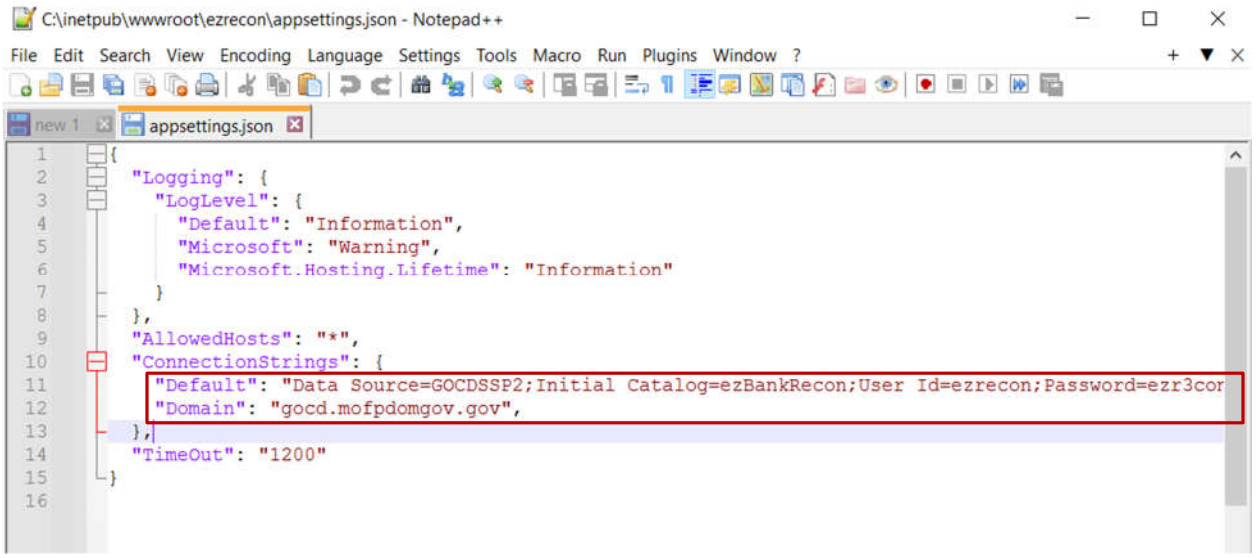
3. Follow the instructions on the wizard



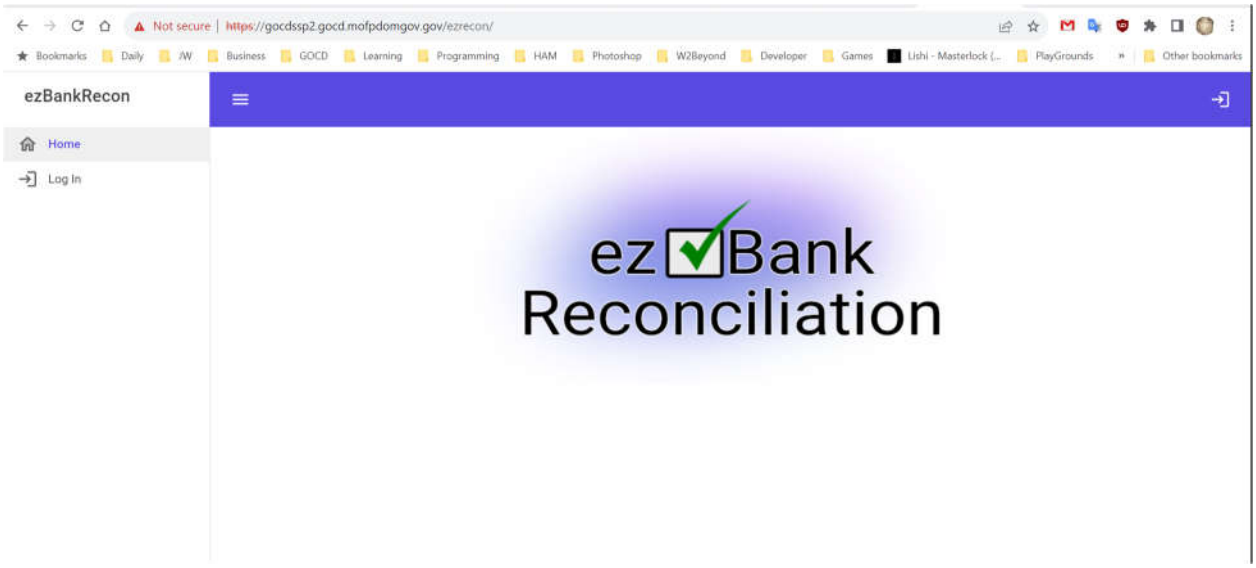
4. Once complete the application should be installed



5. Next, locate the config file in the inetpub, wwwroot directory. Open the folder, ezrecon and edit the appsettings.json file.



6. Make sure that the settings under the "ConnectionStrings" are pointing to the correct domain and correct database. And that the SQLLinkServer is configured if you have the databased linked. Save the settings and run the open the website.



7. Navigate to the {url}/admin/setup to configure the database. The setup installs the database tables and creates a new admin user.



Enter the admin information and click SETUP DATABASE.

Once done you will get a confirmation. The program is now ready to be used.

Database Schema

Table Name	Purpose
[dbo].[account_bank_file_mapping]	Used to store the bank file mappings for the columns for when the application uploads a file. This is stored per account
[dbo].[account_ledger_file_mapping]	Used to store the ledger file mappings for the columns for when the application uploads a file. This is stored per account
[dbo].[accounts]	This stores the various accounts that are entered in the application that are available for reconciliation
[dbo].[processes]	A list of standard processes that the system tracks
[dbo].[roles]	Stores a description of the roles in the database
[dbo].[user_accounts]	Stores the accounts that a user is able to access on the system
[dbo].[user_preferences]	Stores the user preferences
[dbo].[user_roles]	Stores the roles assigned to the users
[dbo].[users]	Stores the users who are able to access the system
The following tables are generated per each account	
{schema}.[accounting_periods]	Stores the accounting periods
{schema}.[adjustments]	Stores the adjustments that are entered
{schema}.[bank_transaction_groups]	Stores the groupings decided on for the bank transactions
{schema}.[bank_transactions]	Stores the bank transactions that are used in the reconciliation process
{schema}.[bank_transactions_staging]	Houses transactions that are uploaded until they are ready for processing. This is where the transactions are uploaded to
{schema}.[closing_balances]	Stores the closing balances for the various periods
{schema}.[custom_processes]	Store the list of custom processes that are available for the end user to run
{schema}.[custom_reports]	Stores a list of reports for the end user to run
{schema}.[ledger_transaction_groups]	Stores the groupings decided on for the ledger transactions
{schema}.[ledger_transactions]	Stores the ledger transactions that are used in the reconciliation process
{schema}.[ledger_transactions_staging]	Houses transactions that are uploaded until they are ready for processing. This is where the transactions are uploaded to
{schema}.[process_logs]	Stores partial audits of action taken in the system
{schema}.[reports_history]	Stores the saved reconciliation reports
{schema}.[user_bank_columns]	Stroes the user preference for the columns displayed in the bank table
{schema}.[user_ledger_columns]	Stroes the user preference for the columns displayed in the ledger table

APPENDIX 1 – Scripts for Main Tables

account_bank_file_mapping

```
IF OBJECT_ID(N'[dbo].[account_bank_file_mapping]', N'U') IS NULL BEGIN
	CREATE TABLE [dbo].[account_bank_file_mapping](
	[account_id] [int] NOT NULL,
	[branch] [int] NOT NULL,
	[customer] [int] NOT NULL,
	[account_no] [int] NOT NULL,
	[amount] [int] NOT NULL,
	[currency] [int] NOT NULL,
	[descp1] [int] NOT NULL,
	[descp2] [int] NOT NULL,
	[descp3] [int] NOT NULL,
	[descp4] [int] NOT NULL,
	[descp5] [int] NOT NULL,
	[descp6] [int] NOT NULL,
	[posting_date] [int] NOT NULL,
	[value_date] [int] NOT NULL,
	[clearing_date] [int] NOT NULL,
	[trans_no] [int] NOT NULL,
	[posting_no] [int] NOT NULL,
 CONSTRAINT [PK_bank_file_mapping] PRIMARY KEY CLUSTERED
(
	[account_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];
END
```

account_ledger_file_mapping

```
IF OBJECT_ID(N'[dbo].[account_ledger_file_mapping]', N'U') IS NULL BEGIN
	CREATE TABLE [dbo].[account_ledger_file_mapping](
	[account_id] [int] NOT NULL,
	[jrnl_id] [int] NOT NULL,
	[ldr_entity_id] [int] NOT NULL,
	[eff_date] [int] NOT NULL,
	[jrnl_seq_nbr] [int] NOT NULL,
	[ministry] [int] NOT NULL,
	[program] [int] NOT NULL,
	[subprog] [int] NOT NULL,
	[account] [int] NOT NULL,
	[project] [int] NOT NULL,
	[sof] [int] NOT NULL,
	[sector] [int] NOT NULL,
	[posting_yr] [int] NOT NULL,
	[posting_pd] [int] NOT NULL,
	[dr_cr_code_1] [int] NOT NULL,
	[descp] [int] NOT NULL,
	[trans_amt] [int] NOT NULL,
	[date_posted] [int] NOT NULL,
	[pmt_meth_id] [int] NOT NULL,
	[pmt_ref_nbr] [int] NOT NULL,
```

```
      [pmt_ref_date] [int] NOT NULL,
      [bank_id] [int] NOT NULL,
      [bank_acct_nbr] [int] NOT NULL,
      [vendor_id] [int] NOT NULL,
      [vendor_loc_code] [int] NOT NULL,
      [vendor_name] [int] NOT NULL,
      [payable_entity_id] [int] NOT NULL,
      [pmt_rqst_gross_amt] [int] NOT NULL,
      [pmt_amt] [int] NOT NULL,
      [type_of_pmt] [int] NOT NULL,
 CONSTRAINT [PK_account_ledger_file_mapping] PRIMARY KEY CLUSTERED
(
      [account_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];
END;" ,

@"IF OBJECT_ID(N'[dbo].[accounts]', N'U') IS NULL BEGIN
      CREATE TABLE [dbo].[accounts](
      [id] [int] IDENTITY(1,1) NOT NULL,
      [account_name] [varchar](100) NOT NULL,
      [account_schema] [varchar](5) NOT NULL,
      [account_no] [varchar](50) NOT NULL,
      [ss_account_no] [varchar](50) NOT NULL,
 CONSTRAINT [PK_ez.accounts] PRIMARY KEY CLUSTERED
(
      [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
 CONSTRAINT [UK_account_schema] UNIQUE NONCLUSTERED
(
      [account_schema] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];
END;
```

processes

```
IF OBJECT_ID(N'[dbo].[processes]', N'U') IS NULL BEGIN
      CREATE TABLE [dbo].[processes](
      [id] [int] NOT NULL,
      [process] [varchar](50) NOT NULL,
 CONSTRAINT [PK_ez.process] PRIMARY KEY CLUSTERED
(
      [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];

  INSERT INTO [dbo].[processes]([id],[process])
     VALUES (1,'Added Closing Balance'),(2,'Removed Closing Balance'),(3,'Imported
Bank Transaction'),(4,'Imported Ledger Transactions'),(5,'Uploaded Bank
File/Transaction'),(6,'Uploaded Ledger File/Transactions'),(7,'Ledger Staging
Process'),(8,'Bank Staging Process'),(9,'Added Adjustment'),(10,'Removed
```

```
Adjustment'),(11,'Ran Auto Matching Process'),(12,'Ran Custom
Process'),(13,'Generate SmartStream Clearance File');

END;
```

roles

```
IF OBJECT_ID(N'[dbo].[roles]', N'U') IS NULL BEGIN
      CREATE TABLE [dbo].[roles](
      [id] [int] IDENTITY(1,1) NOT NULL,
      [role] [varchar](50) NOT NULL,
      [role_descp] [varchar](100) NOT NULL,
 CONSTRAINT [PK_ez.roles] PRIMARY KEY CLUSTERED
(
      [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];

INSERT INTO [dbo].[roles]([role],[role_descp])
     VALUES('admin','Administrator'),('super','Super User'),('regular','Regular
User');

END;
```

user_accounts

```
IF OBJECT_ID(N'[dbo].[user_accounts]', N'U') IS NULL BEGIN
      CREATE TABLE [dbo].[user_accounts](
      [user_id] [int] NOT NULL,
      [account_id] [int] NOT NULL,
      [enable_super] [bit] NOT NULL
) ON [PRIMARY];
END;" ,

@"IF OBJECT_ID(N'[dbo].[user_preferences]', N'U') IS NULL BEGIN
      CREATE TABLE [dbo].[user_preferences](
      [user_id] [int] NOT NULL,
      [grid_size] [int] NOT NULL,
PRIMARY KEY CLUSTERED
(
      [user_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];
END;
```

users

```
IF OBJECT_ID(N'[dbo].[users]', N'U') IS NULL BEGIN
      CREATE TABLE [dbo].[users](
      [id] [int] IDENTITY(1,1) NOT NULL,
      [name] [varchar](50) NOT NULL,
      [username] [varchar](50) NOT NULL,
```

```
 CONSTRAINT [PK_ez.users] PRIMARY KEY CLUSTERED
(
      [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
 CONSTRAINT [UK_users] UNIQUE NONCLUSTERED
(
      [username] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];
END;" ,
@"IF OBJECT_ID(N'[dbo].[user_roles]', N'U') IS NULL BEGIN
      CREATE TABLE [dbo].[user_roles](
      [user_id] [int] NOT NULL,
      [role_id] [int] NOT NULL,
 CONSTRAINT [PK_ez.user_roles] PRIMARY KEY CLUSTERED
(
      [user_id] ASC,
      [role_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];

END;" ,
@"INSERT INTO [dbo].[users]([name],[username])
     VALUES('{0}','{1}');",
@"INSERT INTO [dbo].[user_roles]([user_id],[role_id])
     VALUES(1,1);
```

APPENDIX 2 – Scripts for Account Tables

accounting_periods

```
  IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = '{0}')
                   BEGIN
                    EXEC( 'CREATE SCHEMA {0}' );
                   END;"
             ,@"IF OBJECT_ID(N'{0}.[accounting_periods]', N'U') IS NULL BEGIN
       CREATE TABLE {0}.[accounting_periods](
              [id] [int] IDENTITY(1,1) NOT NULL,
              [descp] [varchar](50) NOT NULL,
              [posting_pd] [int] NOT NULL,
              [posting_year] [int] NOT NULL,
              [accounting_pd] [int] NOT NULL,
              [accounting_year] [int] NOT NULL,
        CONSTRAINT [PK_{0}_accounting_period] PRIMARY KEY CLUSTERED
       (
              [id] ASC
       )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
       ) ON [PRIMARY];
 END; ",

@"IF OBJECT_ID(N'{0}.[adjustments]', N'U') IS NULL BEGIN
       CREATE TABLE {0}.[adjustments](
              [id] [int] IDENTITY(1,1) NOT NULL,
              [amount] [money] NOT NULL,
              [comment] [varchar](500) NULL,
              [entered_by] [int] NULL,
              [date_imported] [datetime] NULL
       ) ON [PRIMARY];
END
```

bank_transaction_groups

```
IF OBJECT_ID(N'{0}.[bank_transaction_groups]', N'U') IS NULL BEGIN
       CREATE TABLE {0}.[bank_transaction_groups](
              [id] [int] IDENTITY(1,1) NOT NULL,
              [descp] [varchar](50) NOT NULL,
              [is_debit] [bit] NOT NULL,
        CONSTRAINT [PK_{0}_bank_code_groups] PRIMARY KEY CLUSTERED
       (
              [id] ASC
       )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
       ) ON [PRIMARY];

INSERT INTO {0}.[bank_transaction_groups](
              [descp],
              [is_debit])
VALUES('Credit',0),('Debit',1);

END;
```

bank_transactions

```
IF OBJECT_ID(N'{0}.[bank_transactions]', N'U') IS NULL BEGIN
	CREATE TABLE {0}.[bank_transactions](
		[id] [int] IDENTITY(1,1) NOT NULL,
		[branch] [varchar](50) NULL,
		[customer] [varchar](50) NULL,
		[account_no] [varchar](50) NULL,
		[amount] [money] NULL,
		[currency] [varchar](10) NULL,
		[descp1] [varchar](500) NULL,
		[descp2] [varchar](500) NULL,
		[descp3] [varchar](500) NULL,
		[descp4] [varchar](500) NULL,
		[descp5] [varchar](500) NULL,
		[descp6] [varchar](500) NULL,
		[posting_date] [date] NULL,
		[value_date] [date] NULL,
		[clearing_date] [date] NULL,
		[trans_no] [int] NULL,
		[posting_no] [int] NULL,
		[accounting_period_id] [int] NULL,
		[matched] [bit] NOT NULL,
		[matched_bank_transactions] [varchar](max) NULL,
		[matched_ledger_transactions] [varchar](max) NULL,
		[date_matched] [datetime] NULL,
		[matched_by] [int] NULL,
		[user_comments] [varchar](max) NULL,
		[row_color] [varchar](15) NULL,
		[is_locked] [bit] NOT NULL,
		[lock_user] [int] NULL,
		[date_imported] [datetime] NULL,
		[import_user] [int] NULL,
		[bank_transaction_group_id] [int] NOT NULL,
	 CONSTRAINT [PK_{0}_bank_transactions] PRIMARY KEY CLUSTERED
	(
		[id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY];


	ALTER TABLE {0}.[bank_transactions]  WITH CHECK ADD  CONSTRAINT
[FK_{0}_bank_transactions_accounting_periods] FOREIGN KEY([accounting_period_id])
	REFERENCES {0}.[accounting_periods] ([id]);

	ALTER TABLE {0}.[bank_transactions] CHECK CONSTRAINT
[FK_{0}_bank_transactions_accounting_periods];

END;
```

bank_transactions_staging

```
IF OBJECT_ID(N'{0}.[bank_transactions_staging]', N'U') IS NULL BEGIN
	CREATE TABLE {0}.[bank_transactions_staging](
		[id] [int] IDENTITY(1,1) NOT NULL,
		[branch] [varchar](50) NOT NULL,
```

```
		[customer] [varchar](50) NOT NULL,
		[account_no] [varchar](50) NOT NULL,
		[amount] [money] NOT NULL,
		[currency] [varchar](10) NOT NULL,
		[descp1] [varchar](500) NULL,
		[descp2] [varchar](500) NULL,
		[descp3] [varchar](500) NULL,
		[descp4] [varchar](500) NULL,
		[descp5] [varchar](500) NULL,
		[descp6] [varchar](500) NULL,
		[posting_date] [date] NULL,
		[value_date] [date] NULL,
		[clearing_date] [date] NULL,
		[trans_no] [int] NULL,
		[posting_no] [int] NULL,
		[accounting_period_id] [int] NULL,
		[matched] [bit] NOT NULL,
		[matched_bank_transactions] [varchar](max) NULL,
		[matched_ledger_transactions] [varchar](max) NULL,
		[date_matched] [datetime] NULL,
		[matched_by] [int] NULL,
		[user_comments] [varchar](max) NULL,
		[row_color] [varchar](15) NULL,
		[is_locked] [bit] NOT NULL,
		[lock_user] [int] NULL,
		[date_imported] [datetime] NULL,
		[import_user] [int] NULL,
		[bank_transaction_group_id] [int] NULL,
	 CONSTRAINT [PK_{0}_bank_transactions_staging] PRIMARY KEY CLUSTERED
	(
		[id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY];


	ALTER TABLE {0}.[bank_transactions_staging]  WITH CHECK ADD  CONSTRAINT
[FK_{0}_bank_transactions_staging_accounting_periods] FOREIGN
KEY([accounting_period_id])
	REFERENCES {0}.[accounting_periods] ([id]);


	ALTER TABLE {0}.[bank_transactions_staging] CHECK CONSTRAINT
[FK_{0}_bank_transactions_staging_accounting_periods];

END;
```

closing_balances

```
IF OBJECT_ID(N'{0}.[closing_balances]', N'U') IS NULL BEGIN
	CREATE TABLE {0}.[closing_balances](
		[id] [int] IDENTITY(1,1) NOT NULL,
		[bank_balance] [money] NOT NULL,
		[accounting_period_id] [int] NOT NULL,
		[ledger_balance] [money] NOT NULL,
		[entered_by] [int] NOT NULL,
```

```
            [date_entered] [datetime] NOT NULL,
       CONSTRAINT [PK_{0}_closing_balances] PRIMARY KEY CLUSTERED
      (
            [id] ASC
      )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
      ) ON [PRIMARY];
END;
```

ledger_transaction_groups

```
IF OBJECT_ID(N'{0}.[ledger_transaction_groups]', N'U') IS NULL BEGIN
      CREATE TABLE {0}.[ledger_transaction_groups](
            [id] [int] IDENTITY(1,1) NOT NULL,
            [descp] [varchar](50) NOT NULL,
            [is_debit] [bit] NOT NULL,
       CONSTRAINT [PK_{0}_ss_code_groups] PRIMARY KEY CLUSTERED
      (
            [id] ASC
      )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
      ) ON [PRIMARY];

INSERT INTO {0}.[ledger_transaction_groups](
            [descp],
            [is_debit])
VALUES('Credit',0),('Debit',1);

END;
```

ledger_transactions

```
IF OBJECT_ID(N'{0}.[ledger_transactions]', N'U') IS NULL BEGIN
      CREATE TABLE {0}.[ledger_transactions](
            [id] [int] IDENTITY(1,1) NOT NULL,
            [jrnl_id] [varchar](50) NOT NULL,
            [ldr_entity_id] [nchar](10) NOT NULL,
            [eff_date] [date] NOT NULL,
            [jrnl_seq_nbr] [int] NOT NULL,
            [ministry] [nchar](10) NOT NULL,
            [program] [nchar](10) NOT NULL,
            [subprog] [nchar](10) NOT NULL,
            [account] [nchar](10) NOT NULL,
            [project] [nchar](10) NOT NULL,
            [sof] [nchar](10) NOT NULL,
            [sector] [nchar](10) NOT NULL,
            [posting_yr] [int] NOT NULL,
            [posting_pd] [int] NOT NULL,
            [dr_cr_code_1] [nchar](10) NULL,
            [descp] [varchar](100) NULL,
            [trans_amt] [money] NOT NULL,
            [date_posted] [date] NOT NULL,
            [pmt_meth_id] [nchar](10) NULL,
            [pmt_ref_nbr] [int] NULL,
            [pmt_ref_date] [date] NULL,
```

```
		[bank_id] [nchar](10) NULL,
		[bank_acct_nbr] [varchar](50) NULL,
		[vendor_id] [varchar](50) NULL,
		[vendor_loc_code] [nchar](10) NULL,
		[vendor_name] [varchar](50) NULL,
		[payable_entity_id] [nchar](10) NULL,
		[pmt_rqst_gross_amt] [money] NULL,
		[pmt_amt] [money] NULL,
		[type_of_pmt] [nchar](10) NULL,
		[accounting_period_id] [int] NULL,
		[matched] [bit] NOT NULL,
		[matched_bank_transactions] [varchar](max) NULL,
		[matched_ledger_transactions] [varchar](max) NULL,
		[date_matched] [datetime] NULL,
		[matched_by] [int] NULL,
		[user_comments] [varchar](max) NULL,
		[row_color] [varchar](15) NULL,
		[is_locked] [bit] NOT NULL,
		[lock_user] [int] NULL,
		[date_imported] [datetime] NULL,
		[import_user] [int] NULL,
		[ledger_transaction_group_id] [int] NULL,
		[generated] [bit] NULL,
		[date_generated] [datetime] NULL,
	 CONSTRAINT [PK_{0}_ledger_transactions] PRIMARY KEY CLUSTERED
	(
		[id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY];

ALTER TABLE {0}.[ledger_transactions]  WITH CHECK ADD  CONSTRAINT
[FK_{0}_ledger_transactions_accounting_periods] FOREIGN KEY([accounting_period_id])
	REFERENCES {0}.[accounting_periods] ([id]);

	ALTER TABLE {0}.[ledger_transactions] CHECK CONSTRAINT
[FK_{0}_ledger_transactions_accounting_periods];

END;
```

ledger_transactions_staging

```
IF OBJECT_ID(N'{0}.[ledger_transactions_staging]', N'U') IS NULL BEGIN
	CREATE TABLE {0}.[ledger_transactions_staging](
		[id] [int] IDENTITY(1,1) NOT NULL,
		[jrnl_id] [varchar](50) NOT NULL,
		[ldr_entity_id] [nchar](10) NOT NULL,
		[eff_date] [date] NOT NULL,
		[jrnl_seq_nbr] [int] NOT NULL,
		[ministry] [nchar](10) NOT NULL,
		[program] [nchar](10) NOT NULL,
		[subprog] [nchar](10) NOT NULL,
		[account] [nchar](10) NOT NULL,
		[project] [nchar](10) NOT NULL,
		[sof] [nchar](10) NOT NULL,
		[sector] [nchar](10) NOT NULL,
		[posting_yr] [int] NOT NULL,
```

```
		[posting_pd] [int] NOT NULL,
		[dr_cr_code_1] [nchar](10) NULL,
		[descp] [varchar](100) NULL,
		[trans_amt] [money] NOT NULL,
		[date_posted] [date] NOT NULL,
		[pmt_meth_id] [nchar](10) NULL,
		[pmt_ref_nbr] [int] NULL,
		[pmt_ref_date] [date] NULL,
		[bank_id] [nchar](10) NULL,
		[bank_acct_nbr] [varchar](50) NULL,
		[vendor_id] [varchar](50) NULL,
		[vendor_loc_code] [nchar](10) NULL,
		[vendor_name] [varchar](50) NULL,
		[payable_entity_id] [nchar](10) NULL,
		[pmt_rqst_gross_amt] [money] NULL,
		[pmt_amt] [money] NULL,
		[type_of_pmt] [nchar](10) NULL,
		[accounting_period_id] [int] NULL,
		[matched] [bit] NOT NULL,
		[matched_bank_transactions] [varchar](max) NULL,
		[matched_ledger_transactions] [varchar](max) NULL,
		[date_matched] [datetime] NULL,
		[matched_by] [int] NULL,
		[user_comments] [varchar](max) NULL,
		[row_color] [varchar](15) NULL,
		[is_locked] [bit] NOT NULL,
		[lock_user] [int] NULL,
		[date_imported] [datetime] NULL,
		[import_user] [int] NULL,
		[ledger_transaction_group_id] [int] NULL,
		[generated] [bit] NULL,
		[date_generated] [datetime] NULL,
	 CONSTRAINT [PK_{0}_ledger_transactions_staging] PRIMARY KEY CLUSTERED
	(
		[id] ASC
	)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
	) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY];

ALTER TABLE {0}.[ledger_transactions_staging]  WITH CHECK ADD  CONSTRAINT
[FK_{0}_ledger_transactions_staging_accounting_periods] FOREIGN
KEY([accounting_period_id])
	REFERENCES {0}.[accounting_periods] ([id]);


	ALTER TABLE {0}.[ledger_transactions_staging] CHECK CONSTRAINT
[FK_{0}_ledger_transactions_staging_accounting_periods];



END;
```

process_logs

```
IF OBJECT_ID(N'{0}.[process_logs]', N'U') IS NULL BEGIN
	CREATE TABLE {0}.[process_logs](
		[id] [int] IDENTITY(1,1) NOT NULL,
```

```
            [process_id] [int] NOT NULL,
            [accounting_period_id] [int] NULL,
            [user_id]  [int] NOT NULL,
            [date_executed] [datetime] NOT NULL,
            [descp] [varchar](100) NOT NULL,
       CONSTRAINT [PK_{0}_process_logs] PRIMARY KEY CLUSTERED
      (
            [id] ASC
      )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
      ) ON [PRIMARY];

END;
```

user_bank_columns

```
IF OBJECT_ID(N'{0}.[user_bank_columns]', N'U') IS NULL BEGIN
      CREATE TABLE {0}.[user_bank_columns](
            [user_id] [int] NOT NULL,
            [branch] [bit] NOT NULL,
            [customer] [bit] NOT NULL,
            [account_no] [bit] NOT NULL,
            [amount] [bit] NOT NULL,
            [currency] [bit] NOT NULL,
            [descp1] [bit] NOT NULL,
            [descp2] [bit] NOT NULL,
            [descp3] [bit] NOT NULL,
            [descp4] [bit] NOT NULL,
            [descp5] [bit] NOT NULL,
            [descp6] [bit] NOT NULL,
            [posting_date] [bit] NOT NULL,
            [value_date] [bit] NOT NULL,
            [clearing_date] [bit] NOT NULL,
            [trans_no] [bit] NOT NULL,
            [posting_no] [bit] NOT NULL,
            [grouping] [bit] NOT NULL,
            [period] [bit] NOT NULL,
            [user_comments] [bit] NOT NULL,
       CONSTRAINT [PK_{0}_user_bank_colunms] PRIMARY KEY CLUSTERED
      (
            [user_id] ASC
      )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
      ) ON [PRIMARY];

END;
```

user_ledger_columns

```
IF OBJECT_ID(N'{0}.[user_ledger_columns]', N'U') IS NULL BEGIN
      CREATE TABLE {0}.[user_ledger_columns](
            [user_id] [int] NOT NULL,
            [jrnl_id] [bit] NOT NULL,
            [ldr_entity_id] [bit] NOT NULL,
            [eff_date] [bit] NOT NULL,
```

```
            [jrnl_seq_nbr] [bit] NOT NULL,
            [ministry] [bit] NOT NULL,
            [program] [bit] NOT NULL,
            [subprog] [bit] NOT NULL,
            [account] [bit] NOT NULL,
            [project] [bit] NOT NULL,
            [sof] [bit] NOT NULL,
            [sector] [bit] NOT NULL,
            [posting_yr] [bit] NOT NULL,
            [posting_pd] [bit] NOT NULL,
            [dr_cr_code_1] [bit] NOT NULL,
            [descp] [bit] NOT NULL,
            [trans_amt] [bit] NOT NULL,
            [date_posted] [bit] NOT NULL,
            [pmt_meth_id] [bit] NOT NULL,
            [pmt_ref_nbr] [bit] NOT NULL,
            [pmt_ref_date] [bit] NOT NULL,
            [bank_id] [bit] NOT NULL,
            [bank_acct_nbr] [bit] NOT NULL,
            [vendor_id] [bit] NOT NULL,
            [vendor_loc_code] [bit] NOT NULL,
            [vendor_name] [bit] NOT NULL,
            [payable_entity_id] [bit] NOT NULL,
            [pmt_rqst_gross_amt] [bit] NOT NULL,
            [pmt_amt] [bit] NOT NULL,
            [type_of_pmt] [bit] NOT NULL,
            [grouping] [bit] NOT NULL,
            [period] [bit] NOT NULL,
            [user_comments] [bit] NOT NULL,
       CONSTRAINT [PK_{0}_user_ledger_colunms] PRIMARY KEY CLUSTERED
      (
            [user_id] ASC
      )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
      ) ON [PRIMARY];

END;
```

custom_processes

```
        IF OBJECT_ID(N'{0}.[custom_processes]', N'U') IS NULL BEGIN  CREATE TABLE
{0}.[custom_processes](
      [id] [int] IDENTITY(1,1) NOT NULL,
      [process] [varchar](50) NOT NULL,
[process_name] [varchar](50) NOT NULL,
[hidden] bit NOT NULL,
[parameter] bit NOT NULL,
[userid] bit NOT NULL,
 CONSTRAINT [PK_{0}.custom_processes] PRIMARY KEY CLUSTERED
(
      [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];
END;",
            @"IF OBJECT_ID(N'{0}.[custom_reports]', N'U') IS NULL BEGIN   CREATE
TABLE {0}.[custom_reports](
```

```
	[id] [int] IDENTITY(1,1) NOT NULL,
	[report_name] [varchar](50) NOT NULL,
	[display_name] [varchar](50) NOT NULL,
	[base_address] [varchar](100) NOT NULL,
	[url_setting] [varchar](100)  NULL,
	[stored_procedure] [varchar](50) NULL,
	[parUser] [bit] NULL,
	[parDateRange] [bit] NULL,
	[parBankorLedger] [bit] NULL,
	[parBankGrouping] [bit] NULL,
	[parLedgerGrouping] [bit] NULL,
	[parAccountingPeriod] [bit] NULL,
	[parAccount] [bit] NULL,
 CONSTRAINT [PK_{0}_custom_reports] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY];
End;
```

reports_history

```
IF OBJECT_ID(N'{0}.[reports_history]', N'U') IS NULL BEGIN
CREATE TABLE {0}.[reports_history](
	[id] [int] IDENTITY(1,1) NOT NULL,
	[file_name] [varchar](50) NOT NULL,
	[accounting_period_id] [int] NOT NULL,
	[date_generated] [DateTime] NOT NULL,
	[generated_by] [int] NOT NULL,
 CONSTRAINT [PK_reports_history] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
END;
```

APPENDIX 3 – Scripts for Views

ledger_closing_balance_vw

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'V' AND OBJECT_ID =
OBJECT_ID('{0}.ledger_closing_balance_vw'))
  BEGIN
  EXEC('Create view {0}.ledger_closing_balance_vw as
SELECT        ldr_entity_id,  processing_yr, amt_class_type,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5+ldr_amt_6+ldr_amt_7 as
ldr_amt_1,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5+ldr_amt_6+ldr_amt_7+ldr_
amt_8 as ldr_amt_2,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5+ldr_amt_6+ldr_amt_7+ldr_
amt_8+ldr_amt_9 as ldr_amt_3,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5+ldr_amt_6+ldr_amt_7+ldr_
amt_8+ldr_amt_9+ldr_amt_10 as ldr_amt_4,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5+ldr_amt_6+ldr_amt_7+ldr_
amt_8+ldr_amt_9+ldr_amt_10+ldr_amt_11 as ldr_amt_5,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5+ldr_amt_6+ldr_amt_7+ldr_
amt_8+ldr_amt_9+ldr_amt_10+ldr_amt_11+ldr_amt_12 as ldr_amt_6,
ldr_amt_0+ldr_amt_1 as ldr_amt_7,
ldr_amt_0+ldr_amt_1+ldr_amt_2 as ldr_amt_8,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3 as ldr_amt_9,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4 as ldr_amt_10,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5 as ldr_amt_11,
ldr_amt_0+ldr_amt_1+ldr_amt_2+ldr_amt_3+ldr_amt_4+ldr_amt_5+ldr_amt_6 as ldr_amt_12
FROM            {2}DBSglep.dbo.ldr_acct_bal AS lb
                          WHERE        amt_class_type = ''ACTUAL'' AND ldr_entity_id
= ''D'' AND account = ''{1}'' and ministry=''31''
                                and program=''F500''
                                and subprog=''F51'';') END;
```

ledger_trans_sum_vw

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'V' AND OBJECT_ID =
OBJECT_ID('{0}.ledger_trans_sum_vw'))
  BEGIN
  EXEC('Create {0}.ledger_trans_sum_vw as
select accounting_period_id, sum(trans_amt) as amount
from {0}.ledger_transactions
group by  accounting_period_id') END;
```

bank_trans_sum_vw

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'V' AND OBJECT_ID =
OBJECT_ID('{0}.ledger_bank_sum_vw'))
  BEGIN
  EXEC('Create {0}.bank_trans_sum_vw as
```

```
select accounting_period_id, sum(amount) as amount
from {0}.bank_transactions
group by  accounting_period_id') END;
```

APPENDIX 4 – Scripts for Stored Procedures

bank_staging_process

```
           IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID =
OBJECT_ID('{0}.[bank_staging_process]'))
BEGIN
EXEC('
CREATE procedure {0}.[bank_staging_process]
                    (
                          @p_period int
                          ,@p_user int
                    )
                    as
                    declare @p_year int
                          ,@p_month int

                    begin
                          /* disable messages*/
                          set nocount on;
                          update {0}.[bank_transactions_staging]
                          set [bank_transaction_group_id]=1
                          where amount>0;
                          update {0}.[bank_transactions_staging]
                          set [bank_transaction_group_id]=2
                          where amount<0;
                    end;
')
End;
```

import_ledger_transactions

```
  IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID =
OBJECT_ID('{0}.[import_ledger_transactions]'))
BEGIN
EXEC('CREATE procedure {0}.[import_ledger_transactions]
(
      @p_period int
      ,@p_user int
)
as
declare @p_year int
    ,@p_month int

begin
      /* disable messages*/
    set nocount on;
      /*Get the year and month value*/
      select @p_month=posting_pd, @p_year=posting_year
      from {0}.accounting_periods
      where id=@p_period;
```

```
      /* Import the transactions*/
 select
jl.jrnl_id,ldr_entity_id,eff_date,jl.jrnl_seq_nbr,ministry,program,subprog,account,p
roject,sof,sector,
                    posting_yr,posting_pd,dr_cr_code_1,descp,trans_amt,date_posted,


      pmt.pmt_meth_id,pmt.pmt_ref_nbr,pmt.pmt_ref_date,pmt.bank_id,pmt.bank_acct_nb
r
                                  ,pmt.vendor_id
,pmt.vendor_loc_code,pmt.vendor_name,pmt.payable_entity_id
,pmt.pmt_rqst_gross_amt,pmt.pmt_amt,pmt.type_of_pmt,

                                  @p_period as accounting_period_id,0 as matched,0 as
is_locked,GETDATE() as date_imported,@p_user as import_user
                                  into #tmp_ledger_transaction_prestaging
                                  from {2}DBSjepc.dbo.posted_jrnl_line jl
                                  Left Join {2}DBSpymt.dbo.pmt pmt on pmt.jrnl_id =
jl.jrnl_id
                                  where posting_yr=@p_year
                                  and posting_pd=@p_month
                                  and ldr_entity_id =''D''
                                  and ministry=''31''
                                  and program=''F500''
                                  and subprog=''F51''
                                  and account=''{1}'';


            /*Remove duplicates and insert into staging*/
      insert into {0}.ledger_transactions_staging([jrnl_id]
           ,[ldr_entity_id]
           ,[eff_date]
           ,[jrnl_seq_nbr]
           ,[ministry]
           ,[program]
           ,[subprog]
           ,[account]
           ,[project]
           ,[sof]
           ,[sector]
           ,[posting_yr]
           ,[posting_pd]
           ,[dr_cr_code_1]
           ,[descp]
           ,[trans_amt]
           ,[date_posted]

           ,[pmt_meth_id]
           ,[pmt_ref_nbr]
           ,[pmt_ref_date]
           ,[bank_id]
           ,[bank_acct_nbr]
           ,[vendor_id]
           ,[vendor_loc_code]
           ,[vendor_name]
           ,[payable_entity_id]
           ,[pmt_rqst_gross_amt]
           ,[pmt_amt]
```

```
           ,[type_of_pmt]

           ,[accounting_period_id]
           ,[matched]
           ,[is_locked]
           ,[date_imported]
           ,[import_user])
 select [jrnl_id]
           ,[ldr_entity_id]
           ,[eff_date]
           ,[jrnl_seq_nbr]
           ,[ministry]
           ,[program]
           ,[subprog]
           ,[account]
           ,[project]
           ,[sof]
           ,[sector]
           ,[posting_yr]
           ,[posting_pd]
           ,[dr_cr_code_1]
           ,[descp]
           ,[trans_amt]
           ,[date_posted]

           ,[pmt_meth_id]
           ,[pmt_ref_nbr]
           ,[pmt_ref_date]
           ,[bank_id]
           ,[bank_acct_nbr]
           ,[vendor_id]
           ,[vendor_loc_code]
           ,[vendor_name]
           ,[payable_entity_id]
           ,[pmt_rqst_gross_amt]
           ,[pmt_amt]
           ,[type_of_pmt]

           ,[accounting_period_id]
           ,[matched]
           ,[is_locked]
           ,[date_imported]
           ,[import_user]
                from #tmp_ledger_transaction_prestaging
                where jrnl_id COLLATE SQL_Latin1_General_CP1_CI_AS not in (select
jrnl_id from {0}.ledger_transactions_staging)
                and jrnl_id COLLATE SQL_Latin1_General_CP1_CI_AS not in (select
jrnl_id from {0}.ledger_transactions);

      /* drop the temp table*/
      drop table #tmp_ledger_transaction_prestaging;
end;')
End;
```

ledger_staging_process

```
            IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID
= OBJECT_ID('{0}.[ledger_staging_process]'))
```

```
BEGIN
EXEC('CREATE procedure {0}.[ledger_staging_process]
(
      @p_period int
      ,@p_user int
)
as
begin
      /* disable messages*/
    set nocount on;
      /*Set the groups for the transactions*/

      update {0}.ledger_transactions_staging
      set ledger_transaction_group_id=1
      where trans_amt>0;
      update {0}.ledger_transactions_staging
      set ledger_transaction_group_id=2
      where trans_amt<0;
end;')
End;
```

generate_ss_fILe

```
            IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID
= OBJECT_ID('{0}.[generate_ss_fILe]'))
BEGIN
EXEC('Create procedure {0}.[generate_ss_fILe]  as
begin
select  [type_of_pmt] as pmt_type
       ,convert(varchar, bt.posting_date, 110) tdate,
         CAST(lp.pmt_amt as varchar) amt,
         CAST(lp.pmt_ref_nbr as varchar) ref_nbr
         ,lp.id as id
         into #temp_clearance
        from {0}.ledger_transactions lp, {0}.bank_transactions bt
        where  lp.matched=1
        and bt.descp1=cast(lp.pmt_ref_nbr as varchar)
        and bt.bank_transaction_group_id in (0)--edit the group
        and  lp.pmt_amt=(bt.amount)
        and (lp.generated=0 or lp.generated is null);

update {0}.ledger_transactions
set generated=1,date_generated=GETDATE()
where id in (select id from #temp_clearance);

--edit the bank info
select ''/*bank,bank_no,{1},PC,''+
ISNULL(pmt_type,'''')+'',''+ISNULL(tdate,'''')+'',''+ISNULL(amt,'''')+'',''+ISNULL(r
ef_nbr,'''')
from #temp_clearance;
End;')
End;
```

matching_process

```
     IF NOT EXISTS (SELECT * FROM sys.objects WHERE type = 'P' AND OBJECT_ID =
OBJECT_ID('{0}.[matching_process]'))
BEGIN
EXEC('CREATE procedure {0}.[matching_process]
as

begin
      /* disable messages*/
    set nocount on;
      /*Start with bank*/
DECLARE @BankId int
DECLARE @Amount money
DECLARE @ValToCompare varchar(50)
DECLARE @Rows int
DECLARE @LedgerId int
DECLARE @BankGroup varchar(50)


DECLARE bank_cursor CURSOR FOR
SELECT bt.id, bt.amount, bt.descp1, LOWER(g.descp)
FROM  {0}.[bank_transactions] bt
LEFT JOIN {0}.[bank_transaction_groups] g on bt.bank_transaction_group_id=g.id
WHERE matched is null or matched=0

OPEN bank_cursor
FETCH NEXT FROM bank_cursor INTO @BankId,@Amount,@ValToCompare,@BankGroup

WHILE @@FETCH_STATUS = 0
BEGIN
      /*Make sure we only match to 1 record, let us do a check*/
      DECLARE ledger_cursor CURSOR STATIC FOR
      SELECT lt.id
      FROM {0}.[ledger_transactions] lt
      LEFT JOIN {0}.[ledger_transaction_groups] g on
lt.ledger_transaction_group_id=g.id
      WHERE trans_amt=@Amount
      and LOWER(g.descp)=@BankGroup
      /*<------IF WE GOT RULES FOR COMPARISM HERE*/

      OPEN ledger_cursor
   /*if we got 1 row then we match, if not then move to the next transaction*/
      IF @@Cursor_Rows=1
      BEGIN
        FETCH NEXT FROM ledger_cursor INTO @LedgerId
        /*Maks surs we update both*/
        BEGIN TRANSACTION
        /*Update bank*/
        UPDATE  {0}.[bank_transactions]
        SET matched=1
        ,is_locked=0
        ,lock_user=0
        ,date_matched=GETDATE()
        ,matched_by=-1
        ,[matched_bank_transactions]=0
        ,[matched_ledger_transactions]=@LedgerId
        WHERE id =@BankId
        /*Update the ledger*/
        UPDATE  {0}.[ledger_transactions]
```

```
          SET matched=1
          ,is_locked=0
          ,lock_user=0
          ,date_matched=GETDATE()
          ,matched_by=-1
          ,[matched_bank_transactions]=@BankId
          ,[matched_ledger_transactions]=0
          WHERE id =@LedgerId

          COMMIT;

        END
        CLOSE ledger_cursor
        DEALLOCATE ledger_cursor


        FETCH NEXT FROM bank_cursor INTO @BankId,@Amount,@ValToCompare,@BankGroup

END

CLOSE bank_cursor
DEALLOCATE bank_cursor
/*FIN*/
end;')
End;
```