**USER DOCUMENTATION**

**GUIDE TO VBA PRODUCTS**

**APPLICATIONS IN ACCOUNTS RECEIVABLE**

*Developed by Cooper D. Cohen*

*Designed for MONRO INC.*

**TABLE OF CONTENTS**

**INTRODUCTION**

I. Preface { 2 }

II. Understanding This document *(recommended for Documentation)* { 4 }

III. Common Control Flow Elements *(optional for Documentation)* { 6 }

**DOCUMENTATION**

1. Reconciliation and Payment Upload v4.2 { 8 }
2. Wholesale COD Customer Aging Template {16}
3. Aging Analysis Template v2 {21}
4. CALTRANS TEMPLATE {33}
5. Wholesale +/- $10 Write Off Template v2 {43}
6. Guest Services CC Chargeback Blackline JE {50}
7. CASHUP AR Subledger Reconciliation Tool {57}
8. Aging Reserve Generator {65}

**QUICK START GUIDE**

1. Reconciliation and Payment Upload v4.2 {75}
2. Wholesale COD Customer Aging Template {77}
3. Aging Analysis Template v2 {78}
4. CALTRANS TEMPLATE {79}
5. Wholesale +/- $10 Write Off Template v2 {80}
6. Guest Services CC Chargeback Blackline JE {81}
7. CASHUP AR Subledger Reconciliation Tool {82}
8. Aging Reserve Generator {83}

**I. Preface**

Between the months of October 2020 and June 2021, 8 discrete products were created by the author to replace existing procedures in Accounts Receivable at *Monro Inc.* These applications were written within Excel Macro-Enabled Workbooks, coded in Microsoft’s Visual Basic for Applications. Each product is stand-alone and self-contained: each one will operate individually to perform their respective tasks and do not require any additional programming/supplemental applications to function as intended.

All projects incorporate to some degree the concepts of *generalization* and *data validation/error handling*. Most projects are generalized to handle varying formats of the data used as the input. As little was hardcoded as possible into the applications. Most instances of hard coding are examples of company standards, such as all Monro finance charges are and always will be coded as “RF” in the Document Type, for example. Any non-standard variables were accommodated such that the user has control over how they should be processed. For example, projects that use the “Customer Type” field are adaptable to additions of new Customer Types in the future. In addition to generalization, all projects utilize custom UserForm Windows to allow the user to make selections before executing the Macros. These windows will validate the types and values of user inputs to ensure that each Macro only receives valid input data, avoiding all but the most obscure errors. This was done to provide a seamless and user-friendly experience that allows for easy cross-training to new employees. Most applications do not require a deep understanding of the process in order to perform and obtain results.

In the event that significant changes occur to the company, the user wishes to extend the functionality of existing programs, or a necessary change in process occurs, it can be speculated that minor programming maintenance may need to occur in order to keep these applications operational without reverting to a manual process. In this rare event, this documentation has been provided to the user and future programmers to offer an exposition to the innerworkings of these applications. This document shall be used as the first resource to consult with when such a situation arises. For each application, the following is provided :

Documentation :

1. A clear and concise tabular abstract of the project, detailing the objective, inputs, outputs, and specificity of the project. Passwords to the VBA project have been included.
2. Project Remarks consisting of all notes regarding the project. All detailed notes on specificity and generalization will be located here.
3. A structure diagram of the Project Architecture. *For navigational use by programmers.*
4. A line by line pseudocode description of what the project is executing internally and when. *This is the most in-depth analysis for each project, used by advanced users and programmers who wish to fully understand how each application works.*

Quick Start Guide :

1. A brief set of explicit instructions to guide the user in performing each application. *No programming background required for use.*

**Runtime Changes of Processes Affected :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Application** | **Applies to Process** | **Previous Runtime of Equivalent Task** | **New Runtime** | **Time Reduction** | **Frequency** |
| Reconciliation and Payment Upload v4.2 | Daily Wholesale Deposit Reconciliation | Varies by location. Minimum of 1.5 hrs | 15 mins for each hour previously required. | 1.25 hrs/each hour previously required. | Daily for each warehouse (7) |
| Wholesale COD Customer Aging Template | Wholesale COD Weekly AR Maintenance | 2.5 hrs | 2 mins | ~2.5 hrs | Once weekly |
| Aging Analysis Template v2 | Various new processes | N/A. Manual functionality was not regularly | 2 mins | N/A. New functionality exists. | Once weekly per subtask |
| CALTRANS TEMPLATE | CALTRANS Detailed Statement Creation – New Process upon acquiring CALTRANS as customer. | If this process was performed manually : 2 hrs | Instantly | Monro would not be able to service CALTRANS if not for this application, as stated by CALTRANS. | Twice monthly |
| Wholesale +/-- $10 Write Off Template v2 | Monthly account maintenance | 30 mins | 1 min | 29 mins | Once monthly |
| Guest Services CC Chargeback Blackline JE | Retail CC chargeback JE | 2 hrs | 1 min | ~2 hrs | Once weekly |
| CASHUP AR Subledger Reconciliation Tool | Account Reconciliations | N/A. Manual functionality did not exist. Operation not required to reconcile but may significantly reduce research depending on account. | 1 min | N/A. New functionality exists. Variable reduction in time. | As needed |
| Aging Reserve Generator | Bad Debt Reserve | 6 hrs (including pre/post prep) | 1 hrs (including pre/post prep) | 5 hrs | Once Quarterly |

**II. Understanding This Document**

It is highly recommended for users unfamiliar with VBA to read with this section before continuing to the main body of the documentation.

Throughout the documentation provided, VBA syntax is used to convey high-level insight into how VBA processes code. This approach was favored over processing this document in plain English to provide enhanced knowledge of how these applications function. Doing so also provides improved visibility into the code provided, without requiring a deep understanding of VBA itself. This section is concerned with informing the user of key background in VBA syntax to properly understand what is being stated in each section. The section titled **Remarks** for each document gives a plain English summary of the intentions and notes of the project, while the section titled **Overview of Procedures** gives a line-by-line description of what the code is performing at a specific location in the application. Code is executed starting with the first line and concludes when the procedure is closed with the keywords **End Sub**.

**Project Architecture**

All code written in VBA must be housed within a Procedure. Every procedure must exist inside a Project Module or Class Module. There are four main types of procedures :

1. Public Sub *(short for Subroutine)*
2. Private Sub
3. Public Function *(also known as UDF for User-Defined Function)*
4. Private Function

The user may only interact with procedures declared as Public. Private procedures may only be used internally within a module/VBA Project. As this is user documentation, any private procedures will be omitted from this document, as they are not necessary to understand how the user will interact with the applications.

**Public Subroutines are what define user-accessible Macros.** The only input the user can give to start a macro is the initiation of the macro, in other words, hitting the RUN button. As such, macros cannot have available parameters, as the user cannot input a value. Any user input must be programmed using UserForms or InputBoxes *(a type of UserForm).*

Subroutines are defined with the following syntax. This syntax will be used to start each section of a Subroutine’s definition :

Public Sub Name()

Body of code; all indented inside Sub

End Sub

Note the following :

1. Keywords are highlighted in Blue that VBA uses to identify code.
2. Parentheses directly after the name of the macro denote the parameter listing, of which a subroutine has none to be available as a macro.
3. End keyword is always required to end any sort of procedure or control flow. Other examples include End Function, End If, End Select.

Public Functions coded in VBA will be available to the user just like any normal Excel formula. Because formulas (functions) are used to return a variable result based on changing inputs, parameter listings must be included in function definitions, unlike subs, which cannot have parameter listings. Functions are defined with the following syntax. This syntax will be used to start each section of a Function’s definition :

Public Function Name(*Parameter1 As Type, Parameter 2 As Type, [Optional Parameter3 As Type = DefaultValue])* As Type

Body of code; all indented inside Function

End Function

Note the following:

1. Parentheses directly after the name of the function denote the parameter listing. Each parameter is listed with the variable name, the keyword As, and the type of the variable. Valid types include but are not limited to String, Boolean, Double, Variant, Long.
2. Optional parameters must be declared last with the keyword Optional. A default value is also required in the event the parameter is not specified, following the syntax above.
3. The data type specified outside the parameter listing represents what type the function can return as its result.

Functions behave differently than subs, as they can only control the VALUE of the cell they are housed in, and nothing else. By contrast, subs are not limited to one cell, but to any cell, in any Worksheet, in any Workbook housed on the PC or network. Subs are a much more powerful tool in VBA than Functions. Subs can control not only the values of cells, but the formatting as well. **After a sub executes, it cannot be undone.**

**Types**

Every variable must have an associated type. Each type is handled differently by the code using it. A basic overview of commonly used types is described below.

Integer : A number with no decimal. Ex: 1, 2, -99, 4005.

Double : A number with a decimal, available at higher number ranges. Ex : -1.01, 4.00009, 234569005.08.

String : Any value represented between quotations. Ex: “foo”, “bar”.

Boolean : True or False. VBA treats True as -1 and False as 0. Must be capitalized.

Variant : A catch-all type that uses a *lot* of memory. Useful for when a function can return multiple data types.

Range : Represents the object Range. A range is any uni/multi-cellular reference. Ranges can be fully qualified with the sheet, ex: Sheets(2).Range(“A1”), or unqualified, ex: Range(“A1”). Ranges represent the entire cell(s), not just the values inside. Ranges can be used to format cells in addition to altering their values.

Sheets : Represents a worksheet. Specify a sheet with the index of the tab in the workbook. Ex Sheets(1).

Columns/Rows : Accesses a column or row. To access a row, use the row number as Rows(1). A column uses integers as well, so column AA would be Columns(27).

**III. Common Control Flow Elements**

Within each application’s documentation, common key concepts are used to describe how the program is making decisions and repeating actions when dealing with a certain block of code. These types of structures are known as Control Flow. The user may choose to read this section if they are not familiar with certain structures.

**If Statement :** Used to execute code only if a condition is True.

If (*Condition* = True) Then

Execute these statements

…

…

ElseIf (*Second Condition* = True) Then

Execute these alternate statements

…

…

Else (*Every other condition was False)*

Execute these catch-all statements

…

…

End If

Note the following :

1. Initial If statement is mandatory.
2. ElseIf and Else keywords are optional.
3. End If closes the block of code.
4. The condition in the IF line is known as the *Guard Statement*.
5. First come first served : If the first IF passes, none of the ElseIf’s/Else will be tested. Control is executed on the first True statement, and nothing more. If you want multiple conditions to be tested, separate the If’s as below :

If (*Condition 1*  = True) Then

Execute this

End If

If (*Condition 2* = True ) Then

Execute that

End If

**For Loop :** Used to repeat lines of code for a specified number of iterations.

For i = 1 To 10

Print( i )

Next i

The output of the above loop will look like :

1

2

3

4

5

6

7

8

9

10

i is increasing by 1 each time, and the statements within the loop are executed a total of 10 times. Use this structure when you have a defined number of iterations to loop through.

**For Each Loop :** Used to iterate through each element in a structure. In the case of Excel, usually looping through each Row or Cell in a certain Selection.

For Each *Element* In *Set*

Execute these statements

Usually do something to *Element*

Next *Element*

Example :

For Each Row In Selection.Rows

Row.RowHeight = 2

Next Row

This will change the height of each Row in the Selection to 2.

For loops can also be nested within each other, like many other structures :

For i = 1 To 5

For ii = 1 To 5

Q = i + ii

Next ii

Next i

This structure will iterate ii 5 times while i = 1, then when i = 2, ii will be ran ANOTHER 5 times, increasing by 1 each time.

The value of Q will follow the sequence : 2 > 3 > 4 > 5 > 6 > 3 > 4 > 5 > 6 > 7 > 4 > 5 > 6 > 7 > 8 > 5 > 6 > 7 > 8 > 9 > 6 > 7 > 8 > 9 > 10.

**Reconciliation and Payment Upload v4.2**

|  |  |
| --- | --- |
| OBJECTIVE | Provides a daily reconciliation template exclusively for Wholesale deposits, of which prepares customer invoice data to be converted into an AR batch. |
| INPUT | Visual analysis of invoice data and deposit totals from PowerFlow or other media (to be keyed), along with a database from VEW or JDE that compiles a list of open invoices and corresponding data. |
| OUTPUT | Compiles the relevant fields for each invoice necessary for upload to JDE. Prepares this data for reference by AR in the future, should a problem arise. The data presented is NOT formatted into an AUTOCASH template with the VBA application alone. |
| SPECIFICITY | This project can be used for other applications that require manual keying of invoice numbers or similar, allowing that the final data will be converted into an AR batch. PrepareRec() is specific to this template. |
| PROCEDURES | Both VBA UDFs and Subroutines are used in this document. |
| PASSWORD | CCProjects5423 |

**Remarks**

Sheets(11) : “README” located in the document provides a comprehensive description of how to start the document and understanding the User-Defined Functions available.

This document was prepared primarily to reduce the keying of each invoice down to only one field (the invoice number), with added functionality to quickly autosum and round cash/check payments without leaving the ActiveCell, dramatically reducing the time required to complete the document.

**Project Architecture**

**The following depicts where each procedure is located within the project modules.**

MODULES

PROCEDURES (*Private procedures omitted*)

FindChecks

Public Sub FindCheck()

Heading

Public Sub Heading()

InvListROA

Public Sub InvListROA()

Public Sub RemoveSuff()

InvoiceClean

Public Function INVOICECLN

InvoiceCustomer

Public Function INVOICECUST

InvoiceRound

Public Function INVOICERND

InvoiceRound2 *(Deprecated)*

InvoiceSum

Public Function INVOICESM

PrepareRec

Public Sub PrepareRec()

**Overview of Procedures**

Public Sub FindCheck()

Available from the button “FIND MISSING CHECKS” on Sheets(1) :   
“Reconciliation v4”

Prompts the user for 2 ranges: the first of two columns representing the invoice totals keyed and the adjacent sums, if any, and another of 1 column representing the totals of the checks being deposited, keyed.

Results in tick marks “x” being placed next to each invoice/check if a match was found between the two ranges, or colored RED if no exact match was found.

Used to resolve variances in keying if too many/too few invoices were included with that day’s deposit batch, as opposed to manually reading through the invoice totals to determine which check is missing.

Error prompts and handling are included in the event of user error.

End Sub

Public Sub Heading()

Available from the button “MAKE HEADING” on Sheets(1).

Reads the value and format of the title of the document and formats the heading on Sheets(1).

Error prompts and handling are included in the event of user error.

Non-essential

End Sub

Public Sub InvListROA()

Available from the button “Insert ROW WW” on Sheets(1).

Moves the ActiveCell to Sheets(10) : “Invoice List” to the last open cell in Columns(2) for ease of access to insert a WorldWriter or Query for additional data. Columns(2) was chosen as to line up additional data with the primary CSV format for the VEW file used for most deposit days.

Non-essential

End Sub

Public Sub RemoveSuff()

Can be accessed by the shortcut CTRL + SHIFT + C

Reads all characters in ActiveCell and truncates value, removing every character after and including the first non-numeric character present. This can be used to correct an entry by removing the suffix if more lines are needed and another line will need the suffix for the same account.

Ex : 1110057-2++ > RemoveSuff()

Results In : 1110057

End Sub

**The following functions are already populated in the template. No change to them is necessary unless using this document for an alternate purpose.**

Public Function INVOICECLN(*Invoice As Range*) As Double

Returns the value of the cell *Invoice* without non-numeric characters. This field is kept after final formatting to allow JDE to read a clean invoice number with no suffixes appended. (A suffix would be added to an invoice number if that invoice total is to be rounded or summed with other invoices above, described in the README on Sheets(11).)

This function can be placed anywhere but is set in the template to Columns(3) and Columns(4), depending on which sheet.

End Function

Public Function INVOICECUST(*Input As Range, Column As Integer, DataS As String*) As Double

Takes the value of the cell *Input* and truncates the suffix to perform an artificial VLOOKUP on the table *DataS* (formatted as a string like “’Invoice List’!$B:$I”), returning the value located at the index *Column* in the data table.

Returns the customer number (specified by *Column*), with removed non-numeric characters and leading zeros, as to format correctly for JDE. Designed for use with VEW downloads where the customer number is in the format TN00######.

Column can be changed to reflect where to look in the data table *DataS*.

End Function

Public Function INVOICERND(*Input As Range, Column As Integer, DataS As String*) As Double

Takes the value of the cell *Input* and truncates the suffix to perform an artificial VLOOKUP on the table *DataS* (formatted as a string like “’Invoice List’!$B:$I”), returning the value located at the index *Column* in the data table.

Returns the amount of the invoice (specified by *Column*), as the exact value, or if a suffix is present, rounded according to user input. More information about how the rounding process works is available in the README. A brief overview is described below :

|  |  |  |
| --- | --- | --- |
| SUFFIX As String | RESULT | EXAMPLE |
| “” | Returns original amount | 1110057 = 212.22 |
| + | Rounds up to nearest 1 | 1110057+ = 213.00 |
| ++ | Rounds up to nearest 5 | 1110057++ = 215.00 |
| / | Rounds up to nearest 10 | 1110057/ = 220.00 |
| \* | Rounds up to nearest 0.10 | 1110057\* = 212.30 |
| \*\* | Rounds up to nearest 0.25 | 1110057\*\* = 212.25 |
| -N ; Where N is an integer | Sums N invoices, including current line. No change to current line amount, only used for INVOICESM function. | 1110055 = 313.66  1110056 = 177.74  1110057-3 = 212.22  No rounding, INVOICESM affected. |
| -N+, -N++, -N/, -N\*, -N\*\* | Sums N invoices, including current line, and rounds SUM according to rules above. Current line is adjusted by amount required to obtain rounded sum value. Affects INVOICESM as well. | 1110055 = 313.66  1110056 = 177.74  1110057-3++ = 213.60  Sum of raw invoices is 703.62. To round to 705.00, last line is adjusted. INVOICESM affected. |
| .. | Autosum. INVOICESM affected. No rounding functionality available. | 1110055 = 313.66  1110056 = 177.74  1110057.. = 212.22  Sum of raw invoices is 703.62, reflected in INVOICESM. |

End Function

Public Function INVOICESM(*Input As Range, Optional AutoColumn As Integer = 1, Optional SumColumn As Integer = 4) As Variant*

Takes the value of cell *Input* and identifies the suffix as described on page {12}. If suffix contains the symbols “-“ or “..”, the sum function is activated. The cell where INVOICESM is located will automatically populate the sum of the given N invoices for the case of “-N” syntax. Because the invoice amounts are NOT located in the same column as INVOICESM, the optional parameter *SumColumn* is available to distinguish where the function should look to perform the sum, given as the column integer. This value defaults to 4 if omitted.

Rounding functionality with “-N” syntax is provided. The given total will reflect the sum rounded to the specified significance as described on page {12}.

When “..” is detected as the suffix, the function will attempt to perform an autosum based on invoices above the current line that have the same account number/subledger identifier. The function will look to Columns(*AutoColumn*)-- which defaults to Columns(1) if omitted-- to identify how many invoices above the current line have the same account number/subledger identifier. When a change is detected in Columns(*AutoColumn)*, the total is determined up to that line in Columns(*SumColumn*) and returned by INVOICESM.

No rounding functionality is supported with the syntax “..”.

If no suffix is detected, INVOICESM returns an empty string, effectively hiding the formula when not needed.

End Function

Public Sub PrepareRec()

This Macro is used to format the result in this Workbook as to be consistent with what JDE requires. Note that this Macro does NOT pre-format data into an AUTOCASH template, as the data provided will be scrutinized further by AR to ensure that any items keyed are still open in the JDE subledger, should this document be uploaded at a later date.

Also provided is functionality to insert a column of Unique Identifiers (UI) next to each invoice on the Cash, MasterCard, Visa, Amex, and Discover tabs. The UIs are created by account and position in the workbook. All *consecutive* invoices with the same account number attached are given the *same* UI at runtime. This will result in every line for a specific account being grouped together on the Receipts Entry screen in JDE, instead of each line item being shown separately. This effectively treats all cash and credit card invoices paid together like a check, so they are shown together on one remittance in JDE.

The UIs are created with the format [M]MDDWUUU where :

[M]M is the month of the date found in Sheets(1).Range(“B5”). JDE does not permit the use of leading zeros for this field, so one-digit months will be show as M.

DD is the day of the date found in Sheets(1).Range(“B5”).

W is a one-digit identifier of the wholesale location identified in Sheets(1).Range(“B4”). This is effectively the last digit of the location given.

UUU is the iterator variable assigned, starting at 000 until 999, wrapping around back to 000 should there be >1000 UIs necessary, which is *highly unlikely*.

Below is an example of this effect. Note that suffixes were removed after the macro has finished executing. Non-consecutive account entries will receive separate UIs:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACCOUNT | UI | INVOICE | PAYMENT | SUM (IF EXISTS) |
| 2557575 | 6034000 | 352989 | 101.22 |  |
| 240510 | 6034001 | 455897 | 856.52 |  |
| 240510 | 6034001 | 455898 | 202.16 | 1,058.68 |
| 3088940 | 6034002 | 351004 | 157.75 |  |
| 3088940 | 6034002 | 351005 | 60.66 | 218.41 |
| 240510 | 6034003 | 455774 | 605.00 |  |
| 244141 | 6034004 | 351646 | 95.11 |  |

Upon executing, the Macro executes tasks in the following order :

1. Workbook is automatically saved at runtime.
2. Sheets 2, 3, 6, 7, 8, 9 are designated to be operated on only.
3. Each sheet is searched in column A for the first available #VALUE error, indicating the data has ended.
4. All data below this error is deleted for Columns(1) To Columns(6).
5. Values are pasted to the remaining cells.
6. The Invoice Raw column (invoice numbers with suffixes) is deleted.
7. For any rows detected that were empty between data, each row is deleted.
8. If no data was entered, sheet is hidden.
9. If UIs were selected before running, they are calculated now and entered in a new column at Columns(2). Columns(1) is used as the account number location.
10. Range(“A1”).Select
11. Steps 3-10 are executed for sheets listed in 2.
12. Sheets(1).Select
13. Range(“B4”).Select

Before executing, the user is given several prompts correcting any errors present in cells B4 and B5, as to create a numeric only UI for each line.

The macro will not run if the date given is not a valid date. A warning is given alerting the user to delete #VALUE errors BETWEEN lines of data, to avoid the macro deleting important data below if the error was erroneously left in.

While the Public Functions above are generalized for other uses, this macro is used only for this specific template: it is NOT GENERALIZED for other uses if the formatting of the document is changed. As such, it has been hardcoded in the following ways :

1. UIs are created by data located specifically in cells B4 and B5 in Sheets(1).
2. Only sheets 2, 3, 6, 7, 8, 9 will be formatted by this macro. If sheets are swapped before running, unintended results may occur.
3. The macro is designed to have headings in each operating tab.
4. The macro is only looking for #VALUE errors in Columns(1) for each tab.
5. The maximum lines per sheet is 2000.
6. Only Columns 1-6 are formatted.
7. Only the column with the title “Invoice Raw” in the first row will be deleted.
8. UIs will only go into a new inserted column at position 2.
9. UIs are only generated using the values in Columns(1). If account numbers are located elsewhere, the UIs will not result as anticipated.

End Sub

**Wholesale COD Customer Aging Template**

|  |  |
| --- | --- |
| OBJECTIVE | Identifies and isolates individual line items representing invoices attached to specific subledger accounts based on available criteria. Used to create data for AR to be formatted into an AUTOCASH template for an AR batch. |
| INPUT | A downloaded aging from JDE with the following fields : {Account number, Invoice Date, Document Type, Open Amount} deposited into a single Excel tab. |
| OUTPUT | 3 Tabs representing the resulting rows of a selected task (3 available). Data is left in the original formatting of the input tab. |
| SPECIFICITY | This project can be used for other applications other than the intended purpose, if the input specifications are satisfied. Certain functions of this application have been superseded by Aging Analysis Template v2, while others are only utilized for this specific purpose. |
| PROCEDURES | Several UserForms and a primary Subroutine are included in this template. |
| PASSWORD | CCProjects5756 |

**Remarks**

Due to the nature of COD customers and their habits, storing their ledger data in a typical Accounts Receivable software (such as JDE) will require routine maintenance that normal House Charge customers will not need. This template is designed to alleviate the intense time commitment require to manually perform these functions. The primary functions of the application are to emulate a “balance forward” method of payment.

The end goal is to isolate the following :

1. Invoices that have matching credits in open amount.
2. Credits that have no matching invoice and oldest invoices that the credit can be applied to.
3. Invoices that satisfy the condition -1.00 < P < 1.00 where P is the current open invoice amount, of which will be written off to sales adjustments.

The isolated lines can then be used in an AUTOCASH template to be directly uploaded to JDE and apply as intended.

Each of these functions can be run together, separately, or not at all. Upon execution, the user is prompted with a UserForm which demands valid entries for the location of the data to operate on.

Added functionality is included to specify specific accounts to ignore in the process. The data stored here is not persistent. That is, each execution of the macro will erase the previous preferences.

The initial data set must be sorted by account number before running the program. This caveat is described in further detail below.

**Project Architecture**

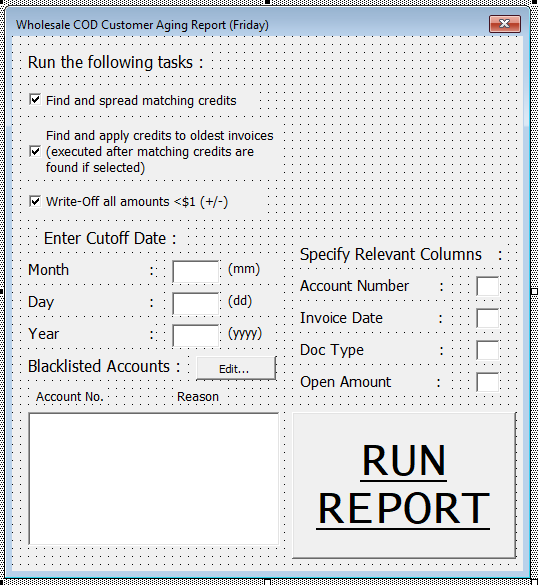
**The following depicts where each procedure is located within the project modules.**

FORMS

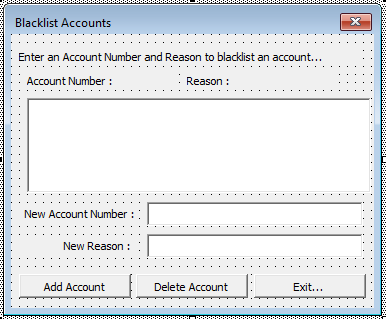
MODULES

PROCEDURES (*Private procedures omitted*)

Cutoff\_Date



BlackList



Friday

Public Sub CODAging()

**Overview of Procedures**

Public Sub CODAging()

Available from the Macros Dialog Box

Triggers the UserForm Cutoff\_Date

Cutoff\_Date.Show

When Cutoff\_Date is shown, the Macro will halt until the user inputs valid values for each of the specified fields. Column locations must be represented as integer X where X > 0. Tab 1 is exclusively used for the data input tab. The fields for the cutoff date itself must result in a valid date.

The cutoff date is used to exclude invoices that are too new to be safely spread onto credits. Any invoice with a date after the cutoff date will be ignored when reading the data, inclusive.

The program will run with a cutoff date of 01/01/0101 if the “X” is clicked, resulting in an empty data set.

When the macro is executed, an error will be thrown if any pre-existing sheets have the following names :

“MASTER DETAIL”

“WORKING SHEET”

“MINOR WRITE OFFS”

“MATCHING CREDITS”

“APPLIED CREDITS”

Ensure that no sheets have these names before running. If you must re-run the macro, either rename the previous resulting sheets or delete them first.

The first tasks executed by the macro are to add 4 sheets and rename them to the names above.

If CheckBox “Write-Off all amounts <$1 (+/-)” was checked Then

Filter all data in the Open Amount Column specified for all values between -1 and 1.

Copy resulting cells to Sheets(3).

End If

Copy all data in Sheets(1) to Sheets(2). Rows will be cut from Sheets(2) onto the appropriate sheet when criteria are matched.

Read through each line on Sheets(2). If the invoice date on each line falls within the cutoff date, assign the row data and address into memory to operate on later. All other invoices will be ignored.

If CheckBox “Find and spread matching credits” was checked Then

The macro will now execute a series of control flow that performs the following :

Loop through each invoice until a credit is discovered. Once a credit is discovered, loop again through the invoices until a match of opposite sign is found.

If a match is found, cut both the credit and matching invoice rows onto Sheets(4).

If no match is found, the macro will then look at all transactions with doctype C[#] to try to see if an unapplied credit matches an NSF check rebill + NSF check fee. If it does, it will cut both rows onto Sheets(4).

End If

If CheckBox “Find and apply credits to oldest invoices” was checked Then

The macro will now execute a series of control flow the performs the following :

Loop through each invoice until a credit is discovered. Once a credit is discovered, loop again through invoices > 0 starting from the oldest.

Deduct as much from the oldest invoice as possible until the total of the credit is reached. If the credit is large enough to cover several invoices, all invoice lines including the credit will be cut onto Sheets(5). The last invoice total will usually be altered, as it will only be paid partially by the available credit. In this case, the macro will wait before cutting this line and finish checking if there are any more credits left. If this is found to be true, the macro will use up the remaining balance of the last previous invoice before continuing and will only copy the “hangover” invoice on one line as to avoid duplicate invoice lines.

Only when all the credits are used up will the last invoice be cut over with an altered total as to only partially apply the remaining credit onto that invoice line.

End If

The above code does not format the results on Sheets(3,4,5) into an AUTOCASH template.

The totals for Sheets(4) and Sheets(5) will be equal to 0.

To avoid using too much computer memory, the macro executes in chunks consisting of 5000 lines. To avoid the bisection of accounts between chunks, before each chunk is calculated, the algorithm “reads” the account column until there is a change in the cell directly underneath the last line of the current chunk, ensuring that all of the last customer’s invoices are included in the chunk.

It can be directly inferred then from above that *the user must sort the initial dataset by account number/subledger identifier in order to obtain results as anticipated.*

End Sub

**Aging Analysis Template v2**

|  |  |
| --- | --- |
| OBJECTIVE | Offers 4 functions to perform on the AR aging report : removes obsolete finance charges, formats an AUTOCASH template for qualifying minor write offs, formats an AUTOCASH for qualifying matching credit spreads, and isolates all accounts that should be taken off credit hold based on fixed criteria. |
| INPUT | Any data set of every open invoice in the AR subledger with the following fields : account number, document type, invoice number, due date, open amount, gross amount (original), business unit (RPMCU) for AUTOCASH only, business unit (RPSFX) for AUTOCASH only, business unit (RPKCO) for AUTOCASH only. To use the fourth function above, another dataset of AR customer master information must be included with the following fields: account number, temporary credit message. |
| OUTPUT | Outputs 1-4 tabs identifying the results to the functions above. Some of these results must be keyed manually into JDE and some can be uploaded. |
| SPECIFICITY | This application is very generalized for multi-purpose use. Any aging format may be used as an input. While the results will always be valid, the final format is hard coded to an AUTOCASH template for compatibility to upload into JDE. |
| PROCEDURES | A comprehensive UserForm and primary Subroutine are used in this template. |
| PASSWORD | CCProjects8811 |

**Remarks**

As with most Accounts Receivable systems, many customer accounts require routine maintenance needs over time. Unless direct human intervention occurs, internal transactions on individual customer accounts that need no input from the customer themselves will not be executed automatically. This template was created to alleviate the labor necessary to perform these transactions manually across the entire subledger.

1. Removal of Obsolete Finance Charges

Accounts that accumulate balances over 60 days past due will qualify for the assessment of finance charges at Monro Inc. While many of these charges will be paid along with the original balance, payment cannot be enforced if the account is otherwise up to date after the finance charges were created. If a customer pays their balance, but not the corresponding finance charges, it is assumed that the customer has disputed these charges and that they must be removed.

To resolve disputes such as these before they are realized, this application is used to evaluate each account and identify any finance charges that no longer correspond to the portion of the account balance that created them in the first place. For example, any finance charges that are older than every other invoice present to date on a given account are clearly dated and should be removed, as the customer has already paid off the balance that once generated that charge in the past.

A fixed criterion of 60 days past the due date of each invoice is used to identify charges that should no longer belong on a given account. This criterion cannot be changed without additional programming, as the software used to generate finance charges in JDE is also fixed to this criterion.

1. Automatic Spreading of Matching Invoices and Credits

In most cases, an invoice with a credit of equal and opposite value present on the same account can be safely applied against each other to keep the account ledger more concise. Like the COD Aging Template, this template also provides a function to isolate all pairs of invoices within each account that fit this criterion to be applied against.

1. Minor Write Off of Immaterial Invoices

When customers make payments that do not equal the original invoice(s) to the penny, line items will still be left open with immaterial amounts. This application can be used to isolate any line item that is between -$1.00 and $1.00, exclusive, for submission for write off.

1. Isolation of Accounts That Do Not Qualify for Credit Hold

Current policy at Monro states that accounts who have a balance over 30 days past the due date of each invoice should be marked for “Credit Hold” and cannot invoice further until their balance is brought to the current period. Unfortunately, the program used within JDE to assign customers on Credit Hold utilizes the invoice date instead of the due date when determining this metric. This application identifies all accounts that are currently on credit hold that should be removed immediately. Accounts that fit this criterion could also have paid their balance very recently, and there has not been a credit refresh since to reflect this.

This portion of the application is primarily used as a work-around to remedy a problem caused by the infrequency of credit refreshment and the improper use of the invoice date instead of the due date by JDE. A more permanent solution is recommended to replace this application in the future.

Each of these functions can be run together, separately, or not at all. Upon execution, the user is prompted with a UserForm which demands valid entries for the location of the data to operate on.

The output results of this file are as follows :

1. Finance Charges : Output as a list of invoices to be keyed manually, as JDE cannot accept “A” adjustment uploads.
2. Spreads : Output as a completed AUTOCASH template that can be directly uploaded to JDE.
3. Minor Write Offs : Output as a completed AUTOCASH template that can be directly uploaded to JDE.
4. Credit Hold Removal : Output as a list of address number, customer name, and new temporary credit message that can be uploaded or manually keyed to the Customer Master by the proper department.

Notes on Specificity :

1. Late charges will only be identified with the code “RF” in the document type column.
2. Late charges will only be assessed as obsolete using the qualifying period off 60 days past and including the due date of each invoice.
3. Matching spreads will use the older of two invoices if both amounts match the credit detected.
4. Spreads will always apply to GL 901.1318 or 00040806 Short ID.
5. Minor write offs will not auto-populate the GL short ID. A reference formula is necessary to determine where to apply the write offs.
6. The credit hold function defines credit hold only as a customer having a positive balance present over 30 days past the due date of each invoice. The number of days late is calculated for each invoice and totaled if the condition holds.
7. Program assumes headers are present in Rows(1); data begins in Rows(2) and is homogenous.
8. Finance charge analysis cannot be retrograde. Only the date of execution can be used to evaluate obsolete finance charges.
9. The doc type “R5” is ignored for minor write offs and spread portions of the program.
10. The application assumes that the dataset is presented with homogenous account detail. That is, every line item for any given account is represented adjacent to other invoices in the account. This is the typical format output of an AR aging report.
11. The following codes are used for identifying Temporary Credit Messages : {“O” = Credit Hold, “\*” = Bypass Credit Hold, “” = Customer not present on DataSet}

**Project Architecture**

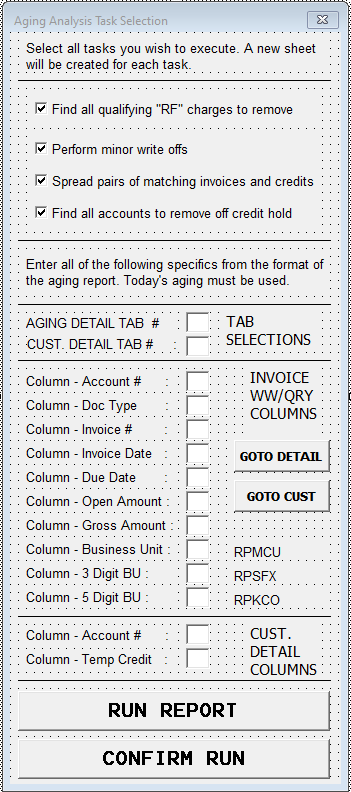
**The following depicts where each procedure is located within the project modules.**

FORMS

MODULES

PROCEDURES (*Private procedures omitted*)

AgingForm



AgingAnalysisAlgorithm

Public Sub AgingAnalysis()

**Overview of Procedures**

Public Sub AgingAnalysis()

AgingForm.Show

The macro halts while the UserForm is shown. The UserForm will only pass control back to the Macro once validated entries are submitted or if the user selects the “X” at the top of the form.

The validation protocol within the UserForm is as follows :

1. Tab selections must be :
   1. Numeric only
   2. <= last tab index
   3. Greater than 0
   4. Representing a tab that is visible
2. Column selections must be :
   1. Numeric only or Alpha only
   2. If numeric, >0 and <16384
   3. If alpha, only 3 characters long and representing a valid column ID
3. Optional Columns (will be colored yellow if no input)
   1. Gross Amount
   2. Business Unit RPMCU
   3. 3 Digit BU RPSFX
   4. 5 Digit BU RPKCO

Without any one of these columns, the Macro will still function. However, not all fields required for AUTOCASH upload will be populated.

Failure to adhere to these guidelines will result in the interface informing the user to correct the selections before continuing.

Once control is passed back to the Macro, the macro executes up to four of the selections chosen from the form. If a box was left unchecked, that certain function is simply stepped over in the code.

All data is copied from the data input tab selected. This tab is renamed to “MASTER DETAIL”

Finance Charges :

Creates 2 sheets, called “RF WORKING SHEET” (M) and “RF CHARGES TO REMOVE” (N). Avoid these names in previously existing worksheets before running the application.

Filters MASTER DETAIL by Document Type for code “RF” –hard coded.

Copies filtered selection to Sheets(M).

As in previous applications, the dataset in Sheets(M) will be operated on in chunks of 2000 rows each, which may slightly vary from chunk to chunk to ensure that each chunk does not partition any customer account when assigning the boundaries. This is done to preserve memory allocated to VBA.

For Each Chunk In DataSet

Record every account that has an RF charge.

Record the date of each RF and attach to account.

Go to MASTER DETAIL.

Evaluate each invoice line for the following :

-Account attached to invoice has RFs

-Current invoice is overdue (Today – Due Date + 58 > 0)

-Invoice is not an RF

-Invoice is a debit, not a credit

If all these criteria are true, record the invoice against the account into memory and record the number of days overdue (Today – Due Date + 58). This invoice qualifies as an invoice that would generate an RF charge and must be compared against the RF charges themselves to determine if the RF still belongs on the account or not.

If no invoices qualified for a given account with RFs Then

Remove all RFs on the account

Else If there exists at least one qualifying invoice Then

Determine how many days overdue the oldest overdue invoice is. This is the earliest date that any RF could be generated with the invoices currently on the account.

Remove any RF that is older than the date described above.

End If

Next Chunk

The RF charges to be removed are pasted onto Sheets(N).

Sheets(N) is formatted as follows :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DT | ACCOUNT | INVOICE | INV DATE | OPEN AMT |
| PL | 2524191 | 103647 | 02/25/21 | -$ 2.70 |
|  | **2524191 Total** |  |  | **- $ 2.70** |
| LC | 2524887 | 100739 | 10/24/20 | $ 2.25 |
| LC | 2524887 | 101457 | 11/28/20 | $ 2.25 |
| LC | 2524887 | 102197 | 12/26/20 | $ 2.25 |
| LC | 2524887 | 102946 | 01/23/21 | $ 2.25 |
| LC | 2524887 | 103648 | 02/25/21 | $ 2.25 |
| LC | 2524887 | 104304 | 03/25/21 | $ 2.25 |
|  | **2524887 Total** |  |  | **$ 13.50** |
| LC | 2551383 | 102952 | 01/23/21 | $ 40.88 |
| LC | 2551383 | 103654 | 02/25/21 | $ 1.02 |
|  | **2551383 Total** |  |  | **$ 41.90** |

Note :

1. RFs to remove are coded as LC when keyed into JDE.
2. RFs that were removed and then paid by the customer will show as a credit. These will also be removed as a PL adjustment (Paid Late). This is equivalent to if the RF charge were never removed in the first place (very rare)
3. Accounts are subtotaled for ease of keying.

Minor Write Offs :

Creates sheet “MINOR WRITE OFFS” (W). Avoid this name in previously existing worksheets before running the application.

Go to MASTER DETAIL.

For Each Invoice Row In Tab

If Open Amount > -1 And Open Amount < 1 Then

Paste Row to Sheets(W)

End If

*Rows with Document Type RF and R5 will be ignored.*

Next Row

Results on Sheets(W) will be formatted like an AUTOCASH template. The heading for such template must be copied over from a blank template.

GL codes are left blank for human determination.

GL date is entered as the current Date.

Matching Spreads :

Creates sheet “SPREAD UPLOAD” (S). Avoid this name in previously existing worksheets before running the application.

Go to MASTER DETAIL.

This section of the application also works on Chunks of 2000 lines to preserve allocated memory. The chunks are slightly adjusted in size to ensure that each account is contained only within one chunk, as to avoid several operations on the same account.

For Each Chunk In DataSet

For Each Row In Chunk

Skip all lines with Document Type of “RF” or “R5” – hard coded.

If current row qualified for a minor write off in previous section (if minor write offs were ran) then skip the current row.

If current Row does not qualify for the invalidating conditions above, record the invoice and attach it to the account into memory.

Next Row

For Each Account that was recorded in the previous loop

Loop through each invoice to find oldest credit, if any

If a Credit is discovered Then

Loop through each invoice starting from oldest.

If any invoice is equal value and opposite sign as Credit Then

Paste both Invoice and Credit onto Sheets(S)

Delete both line items from memory

End If

End If

Repeat for all Credits present on account. Do nothing if a match is not found.

Next Account

Next Chunk

Like the minor write off section, this portion of code will now format the results on Sheets(S) to be in the format of an AUTOCASH template.

Note the following :

1. Date of Execution will be used to populate the GL date.
2. GL account is hardcoded to 901.1318 Short ID 00040806
3. Header is not hardcoded into the template; must be copied from existing blank template.
4. AUTOCASH specifications cannot be changed without additional programming. Customized for use at MONRO Inc.
5. Doctype is not considered when assigning line items other than the fact that Finance Charges (RF) and Rent invoices (R5) will be ignored. This cannot be changed without additional programming.

Credit Hold Removal :

*This section will only operate if an additional “Customer Data” tab is present, and location specified in the UserForm. This tab is renamed to “CUST DETAIL”. The tab should include all customers that would be present on MASTER DETAIL, and a field indicating their current credit status.*

*The criterion “O” is used to identify accounts on Credit Hold in the field described above.*

*Credit Hold is defined as any account where the account maintains a positive balance aged past 30 days of the due date of the balance, AND the total balance of the account is also positive.*

*The due date of the balance is determined by the due date of each line item. The “Past Due Balance” can be calculated by determining if an invoice is 30 days past its scheduled due date. If true, this invoice total is added to the figure “Past Due Balance”. After all invoices have been parsed, this figure can be evaluated to determine the credit status to be updated.*

*Corollary : Accounts that maintain an overall credit balance do not qualify for Credit Hold even if their Past Due Balance is positive.*

*It should be noted that accounts that maintain a positive Past Due Balance but a negative overall balance require immediate maintenance. If no maintenance is performed, it is theoretically possible for the customer to continue invoicing to the point where the overall balance is no longer negative, but the Past Due Balance has not changed, which would trigger the change to Credit Hold. If a spread were to take place before the customer invoiced past the credit balance, this would not occur, as the Past Due Balance would be reduced to 0, eliminating the qualification for Credit Hold. This phenomenon was intentionally excluded in the macro to draw attention to these accounts should they return to Credit Hold before a spread was executed.*

Creates sheets “ACCOUNTS TO TAKE OFF HOLD” (Q) and “ACCOUNTS TO PUT ON HOLD” (R). Avoid this name in previously existing worksheets before running the application.

Go to MASTER DETAIL.

Read through each cell in the Account Number column. Identify all unique entries and record into memory.

Go to CUST DETAIL

Read through each account present in CUST DETAIL DataSet.

For Each Row In DataSet

If Account# Exists in memory Then

Record address of CUST DETAIL row, attach to account.

Record current Temporary Credit Message, attach to account.

End If

Next Row

If Finance Charges were removed earlier Then

Go to Sheets(N)

Remove Subtotals

For Each Row

Record Account and Invoice Number

Next Row

*This set of data will be used to “ignore” invoice lines that are finance charges that will be removed, to accurately reflect the account balances after these adjustments are keyed.*

End If

Go to MASTER DETAIL.

For Each Row in DataSet

If Row represents an invoice that was a minor write off (if previously ran), ignore row.

If Row represents an invoice that was involved in a spread (if previously ran), ignore row.

*The two conditions above will simulate the accounts as if the previous adjustments had already been uploaded, instead of recalculating the aging after the adjustments are keyed, and then running the Credit Hold program.*

If Row represents a Finance Charge Then

If removed in Finance Charge program (if previously ran), ignore row.

Continue if not.

End If

If none of the above three conditions are met, the invoice Row is compared against the execution date with its respective Due Date to determine if the invoice is open past 30 days after the due date. If this is true, it is added to the Past Due Balance as well as the Overall Balance. If not, it is only tabulated in the Overall balance for the account. These values will be used later to determine Credit Status for each account.

Next Row

For Each Account that was previously tabulated

If the Temporary Credit Message (TCM) is “O” Then (*account is already on hold)*

Move to Sheets(Q) if Past Due Balance or Overall Balance is <=0.

ElseIf the TCM is “” Then (*account was missing on CUST DETAIL)*

Mark account into memory as MISSING ACCOUNTS

Else (*account is not on hold)*

If Past Due Balance > 0 AND Overall Balance > 0 AND TCM <> “\*” Then

*Code reserved for accounts to bypass Credit Hold*

Move to Sheets(R) *accounts should be on hold*

End If

End If

Next Account

Add New sheet titled “ACCOUNTS MISSING ON CUST DETAIL”(Y) –as before, avoid this name before running the application.

Move all accounts deemed MISSING ACCOUNTS above to this sheet.

Sheets Q, R, and Y retain the same formatting as the format presented on CUST DETAIL tab.

No formatting changes are made to accommodate the varying method of how this data will be keyed/uploaded into JDE.

End Sub

**CALTRANS TEMPLATE**

|  |  |
| --- | --- |
| OBJECTIVE | Provides a detailed user report displaying detailed invoice information, currently open on a given AR account, formatted for printing and mailing to AR customers. Designed and customized to suit the needs of AR customer #3061378 GY-CALTRANS DIVISION OF EQUIPMENT. |
| INPUT | Two tabs: one specifying invoice header and line item information from F65100 file. Second tab specifying open invoices present on AR account. Invoices on AR account file must be present on F65100 download. |
| OUTPUT | A formatted document displaying detailed invoice information. See remarks for details. |
| SPECIFICITY | Customized for one specific customer but will operate for any customer. Header must be updated after execution if using for another customer/vendor. |
| PROCEDURES | Two UserForms and one primary Subroutine are used in this document. |
| PASSWORD | CCProjects5814 |

**Remarks**

INPUT *(continued)* : The following fields are required in the Data Sets

1. Store Tab (F65100 Download)
   1. Store Location Number
   2. Store Invoice Number
   3. Invoice Gross Amount
   4. GS/AN Req. Code (Or PO or another custom field)
   5. Invoice Total Tax
   6. Invoice Total Taxable Amount
   7. Invoice Vehicle Make
   8. Invoice Vehicle Model
   9. Invoice Vehicle Mileage
   10. Invoice Vehicle License Plate
   11. Invoice VIN Number
   12. Invoice Line Item Number
   13. Invoice Line Item Description
   14. Invoice Line Item Service Writer
   15. Invoice Line Item Price of Parts
   16. Invoice Line Item Price of Labor
   17. Invoice Line Item Quantity
2. AR Ledger Tab
   1. AR Invoice Number
   2. AR Invoice Date

OUTPUT *(continued)* : The produced document will output the following

A Header with the following information :

1. MONRO Corporate Logo
2. DBA Brand Logo (chosen in UserForm)
3. Start date of transactions
4. End date of transactions
5. Account Number (hardcoded to 3061378)
6. “Consolidated Invoice #” = “I” & Start Date & End Date
7. “Invoice Date” = End Date
8. Customer Address (hardcoded as the following)

CALTRANS DIVISION OF EQUIPMENT

691 SOUTH TUSTIN AVENUE

ORANGE, CA 92866-3312

1. Remit To Address (hardcoded as the following)

MNRO HOLDINGS LLC

DBA <BRAND NAME>

PO BOX 845602

BOSTON, MA 02284-5580

1. Total of All Parts
2. Total of All Taxable Parts
3. Total of All Labor
4. Total Tax
5. Consolidated Invoice Total Due

Individual Invoice Detail with the following information :

1. “Reference” = Monro Invoice Number
2. Store Location
3. Date of Service = Invoice Date
4. Vehicle License Plate
5. VIN Number
6. “Unit Number” = GS/AN Req. Code
7. Vehicle Make
8. Vehicle Model
9. Vehicle Mileage
10. Entry Gross Amt = Invoice Gross Amt
11. Entry Tax = Invoice Tax
12. Entry Taxable = Invoice Taxable
13. Entry Total of all Parts
14. Entry Total of all Labor
15. Itemized Line Items with the following information :
    1. Quantity
    2. Item Number
    3. Item Description
    4. Item Service Writer Input (if any, hanging tab below)
    5. Price of Parts
    6. Price of Labor
    7. Extended Parts by Quantity
    8. Extended Parts by Labor

This template was conceived to meet the excessive needs of California Government Agencies in order to receive timely remittance. The template provides a well-formatted “statement” that can be emailed as a PDF or mailed. The produced document can include a logo from the following Monro Brands :

1. Monro Muffler Brake
2. Tire Choice
3. Mr. Tire
4. Tires Now
5. CAR-X
6. Allen Tire
7. Vacant for future use
8. Vacant for future use
9. NONE

Added functionality of a Help Screen was included as a separate UserForm that instructs the user on how to produce the finished document, should they be unaware.

Notes on Specificity :

1. Parts price must be price per item
2. Labor price must be price per item
3. Taxable amount must reflect amount taxable on entire invoice.
4. Tax amount must reflect entire invoice tax.
5. Replace the old tab named “FINISHED DOCUMENT” before re-running the Macro to avoid naming errors.

All invoices detected within the specified date range on the AR tab must also be present on the Store Detail (F65100) download. Failure to do so will result in an error. These Data Sets should intrinsically be a complete intersection, that is, not disjoint or partially intersecting.

*It shall be noted here that the manual production of this document requires several hours of human data manipulation in order to output the fields listed prior in this section without the assistance off a Macro. A request such as this from a customer that demands strict documentation—such as a government agency—is exceptionally rare, and thus required special development of procedure. This document can be used/slightly altered to fit the needs of similar customers that require this information that cannot be obtained from the store locations at an ongoing basis and is otherwise not fulfilled but typical procedures/customer resources used in AR at Monro.*

**Project Architecture**

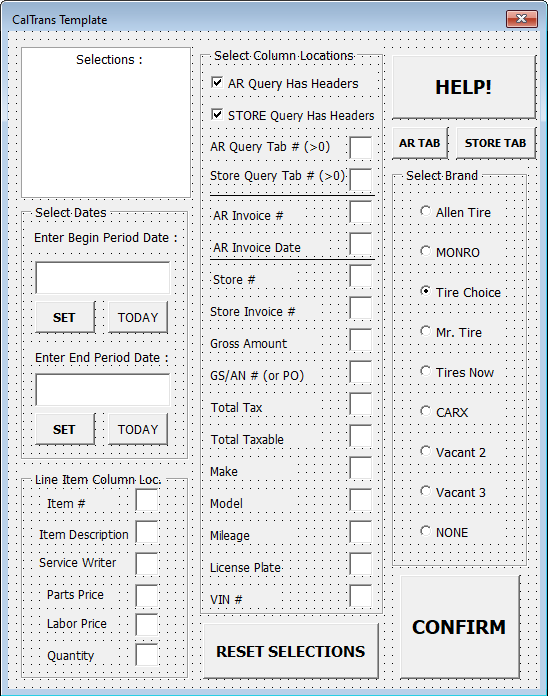
**The following depicts where each procedure is located within the project modules.**

FORMS

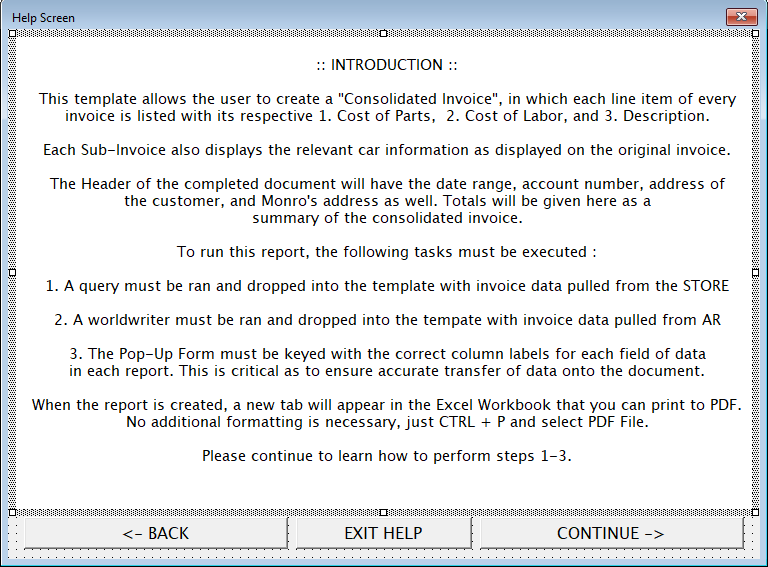
MODULES

PROCEDURES (*Private procedures omitted*)

CalTranForm2



HelpScreen



CalTran

Public Sub CalTran()

**Overview of Procedures**

Public Sub CalTran()

CalTranForm2.Show

The Macro halts while the UserForm is displayed. The UserForm is generalized and secure to the point where invalid values/types will not be accepted and will prompt the user for correction.

The user is prompted for the following inputs :

1. Start date : Used to set the inclusive lower bound of invoice dates. Must be a valid date used with “/” or “-“.
2. End date : Used to set the inclusive upper bound of invoice dates. Must be a valid date used with “/” or “-“. End date cannot be less than the Start date.
3. AR WorldWriter tab location : Must be greater than 0, represent an index of an existing sheet, and represent a sheet that is visible.
4. Store Query tab location : Same constraints as 3.
5. Column locations : Must be an integer between 1 and 16384 or ALPHA character representing column name. Columns required are listed in the **Remarks** section under Input.

Clicking the HELP! Button will trigger the HelpScreen UserForm, of which guides the user how to enter values into the form.

When the UserForm validates the selections, the Macro continues execution.

Adds Sheet named “FINISHED DOCUMENT” at end. Avoid this name in existing sheets before running the Macro.

All font is presented in “Courier New” 9pt.

Before any data is calculated, the Macro executes many lines to format the header as follows :



After the header is complete, the Macro executes the following steps :

Selects all data in AR tab starting at Rows(2) if headers were selected, Rows(1) if not.

Records each invoice that falls on or within the specified date range.

Go to STORE TAB.

Selects all data in Store tab starting at Rows(2) if headers were selected, Rows(1) if not.

For Each Row In Selection

*(each row represents a specific line item for a given invoice. Each line item carries invoice header information as well.)*

Store line item information tagged under invoice number into memory. If this is the first line item of the invoice, store the header information in addition.

Next Row

Go to FINISHED DOCUMENT tab.

Each detected invoice is given a standard entry on the document. An example is shown below.



The cells with the following descriptions are populated from the data directly from the store tab :

1. Reference
2. Store
3. Date of Service
4. License
5. VIN
6. Unit
7. Make
8. Model
9. Mileage
10. Gross AMT
11. Total Tax
12. Parts Taxable

The cells with the following descriptions are populated from the invoice line item data directly from the store tab :

1. Qty
2. Item #
3. Item Description
4. Unit Part
5. Unit Labor

The cells with the following descriptions are calculated from the invoice line item data from the store tab :

1. Extended Part = Unit Part \* QTY
2. Extended Labor = Unit Labor \* QTY
3. Total Parts = Sum of all lines in Extended Part
4. Total Labor = Sum of all lines in Extended Labor

During each entry, the total Parts, total Labor, total Tax, total Taxable, and Gross amount are continuously tabulated to a Grand Total for each category that is populated into the header after all entries are created.

During any given entry, if the Service Writer field is populated, that is, <> “”, the entire phrase will be broken into words and logged into the lines directly below the line item data in groups of 20 characters maximum. And example of this is shown in the photo above under line item “LABOR” where the technician wrote “LEFT FRONT”. The Macro will not separate individual words onto separate lines unless the word is longer than 19 characters.

A new border of “====” will be inserted underneath all line items, signifying that a new entry is starting.

The template will create page breaks after three entries, except on the page containing the header, which will only have two. This will allow enough extra space on each page to ensure that each invoice is not split onto several pages.

After all entries are populated, the totals calculated for each category are placed in the header at this point.

Page margins are adjusted at this stage, along with the page numbers in the footer. Page breaks are adjusted so that the document may be printed immediately to 8.5” x 11” paper without any gaps.

Depending on which logo was selected from the UserForm, a copy has been inserted into the header of the Monro Corporate logo as well as the selection. Two vacant spots were left to accommodate future logo additions into the template.

To add a LOGO as a USER :

1. Unhide the sheet title “LOGOS”
2. Unprotect worksheet
3. Add new logo into the dimensions given at the left of the sheet
4. Protect worksheet
5. Hide worksheet “LOGOS”

The new logo can now be selected from one of the vacant spots on the UserForm. The UserForm logo name must be updated by a programmer.

The Macro has finished formatting the document. The document can be printed to paper or PDF immediately and sent to the customer.

End Sub

**Wholesale +/- $10 Write Off Template v2**

|  |  |
| --- | --- |
| OBJECTIVE | Constructs two AUTOCASH templates that when uploaded results in a balance write-off for all customer accounts present in the Data Set that with a balance between -$10.00 and $10.00. |
| INPUT | A WorldWriter or Aging download of all open invoices on accounts to be operated on. |
| OUTPUT | Two AUTOCASH templates ready for direct upload performing the tasks in the Objective. |
| SPECIFICITY | This application can be run on any set of customers/invoices. Standard AUTOCASH formatting is used as the output. Non-essential features are hard coded into the program, described in the remarks below. |
| PROCEDURES | One UserForm and a primary Subroutine are used in this document. |
| PASSWORD | CCProjects0771 |

**Remarks**

Distinct maintenance needs concerning the Accounts Receivable ledger have been identified exclusively for Wholesale customers at Monro as opposed to Retail or Commercial. As all wholesale business deals with the invoicing of goods only, certain AR functions must be executed periodically to prevent the subledger from becoming unmanageable and unintuitive. Often, cash payments will result in small credits on invoices due to rounding, of which change is not given at the point of sale. Additionally, credit invoices for goods returned offset other debit invoices will accumulate into a small credit or debit balance on any given customer’s account. These scenarios can be remedied by periodically erasing any open entry on every account that falls within a small balance window. Keeping this window between -$10.00 and $10.00 avoids writing off significant amounts each monthly cycle.

This program was designed to isolate these accounts that fit these criteria and format the data in such a way that is interpretable by JDE. This not only concerns the upload format of the AUTOCASH template, but also specific design elements that avoid the need for manual rework after upload.

The prior process required an aging to be scrutinized and manipulated manually to produce the product in about on hour. The new process is designed to execute with a runtime of one minute, with minimal input from the user.

From an accounting perspective, the following tasks must take place to achieve the objective.

1. Account balances need to be identified within the Data Set.
2. Accounts with qualifying balances need to be isolated.
3. All invoices on qualifying accounts need to be paid to a holding/adjustments account (in the case of Monro, this is GL 901.1318) except for one invoice that must retain the total account balance. This will be a $0 transaction.
4. The one isolated invoice must then be written off to the proper GL account depending on store location.

Once this is accomplished, the subledger will be rid of any accounts that are mostly “up-to-date” save for hanging balances within this range, effectively cleaning the subledger. Many times, accounts that have small credit balances offset accounts with small credit balances.

Notes on Specificity :

1. Spread GL code is hard coded to 00040806, GL 901.1318
2. Write off GL codes are hard coded to agree with the table listed in the Procedures section of this document.
3. Both 1 and 2 can be altered post-execution; non-essential.
4. Output is hardcoded to be formatted as an AUTOCASH template.
5. Algorithm only observes accounts with a balance between -$10.00 and $10.00, hard coded.
6. AUTOCASH header must occupy Rows(1) to Rows(5) from a blank template to be included in the output, described below.
7. Data Set must include headers in Rows(1).
8. Previous output sheets must be deleted or renamed before re-executing the Macro. Failure to do so will result in an error.

**Project Architecture**

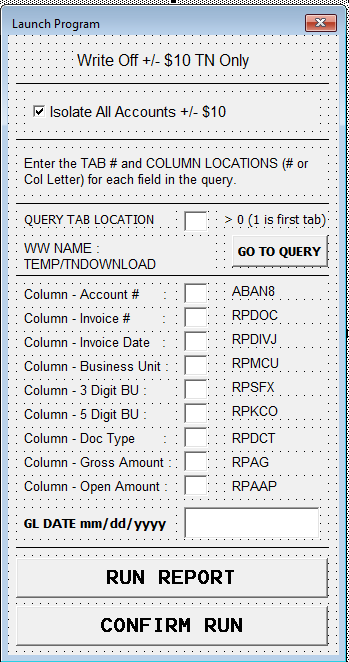
**The following depicts where each procedure is located within the project modules.**

FORMS

MODULES

PROCEDURES (*Private procedures omitted*)

AgingForm



WriteOff

Public Sub WriteOff10()

**Overview of Procedures**

Public Sub WriteOff10()

AgingForm.Show

At this point the macro halts until control is passed back from the UserForm. The UserForm will not release control unless all requested values are validated or until the user clicks the “X” button, triggering the macro to run without processing data, ending immediately.

The required values are as follows :

1. Data Set (Aging or WorldWriter of all open invoice) tab location. Must satisfy the conditions below :
   1. Greater than 0
   2. Represents a tab index that is visible
   3. Represents an index no greater than the last tab.
2. Column locations :
   1. Account number
   2. Invoice number
   3. Invoice date
   4. Business Unit (for AUTOCASH) RPMCU
   5. Business Unit (for AUTOCASH) RPSFX
   6. Business Unit (for AUTOCASH) RPKCO
   7. Document Type RPDCT
   8. Gross Amount
   9. Open Amount

Columns must satisfy the conditions below :

* 1. Must be a number greater than 0 and less than 16384.
  2. Can also be an ALPHA string referencing column location.

1. GL Date
   1. Must be a valid date in format “mm/dd/yyyy”

Once the initial parameters are validated, the Macro continues to execute.

Each invoice that is written off will be read for its respective location code. This code is used to determine which write-off GL will be used.

Sales adjustment/write off GL accounts are hardcoded for wholesale as follows :

|  |  |  |  |
| --- | --- | --- | --- |
| Location ID | Location Name | GL Account | Short ID |
| 1431 | CONOVER | 1431.5016 | 00508366 |
| 1432 | CHARLOTTE | 1432.5016 | 00508412 |
| 1433 | GREENVILLE | 1433.5016 | 00508458 |
| 1434 | KNOXVILLE | 1434.5016 | 00508504 |
| 1537 | JOHNSON CITY | 1537.5016 | 00549803 |
| 1538 | BLACK MOUNTAIN | 1538.5016 | 00550062 |
| 1166 | LOUISVILLE | 1166.5016 | 00366840 |
| 1539 | SALEM *(closed)* | 1539.5016 | 00550321 |

These Short ID codes are used in the RUGLBA column on the AUTOCASH template to assign where to pay invoices to. These GLs are used for write-offs such as this.

Should new locations be added that require this process, programming changes would be required or the user shall manually add the GL accounts in post-execution.

Invoices belonging to alternate locations that were not included in programming will be left with blank GL accounts, which will need to be populated manually by the user.

At this point, the Macro creates two sheets, one for the spread AUTOCASH (S) and one for the write off AUTOCASH (W). These sheets will be populated later after the data is calculated.

Go to Data Set Tab.

Select all data from Rows(2) down to the bottom, gaps will be included but a homogenous data set is preferred.

The next section will read each invoice in the Data Set tabulate the balances for each account.

For Each Row in Selection

Read each invoice within the Row, add invoice value to account balance.

Add +1 to account invoice count.

Next Row

The next step in the process is a large decision tree to determine how the invoices on qualifying accounts must be treated. The following only applies to accounts with a balance greater than -$10.00 and less than $10.00 :

Is the total account balance 0?

=> Add each invoice to Spread

Is the total invoice count = 1?

=> Add only invoice to Write Off

Is every invoice on the account the same sign?

=> No spread required. Add each invoice to Write Off.

Is there an invoice that equals the account balance?

=> Write off qualifying invoice. Spread all other invoices.

Else

For (+) balances

=> Spread all credits. Spread first debit of amount = –(Balance – Debit). Write off debit of amount = account balance. Spread remaining debits.

For (-) balances

=> Spread all debits. Spread first credit of amount = -(Balance – Credit). Write off credit of amount = account balance. Spread remaining credits.

This decision tree is structured so that no credit will ever be applied for more than is available and will only remain as a credit on the spread/write off sheets. This was designed to avoid the JDE discount bug where re-work is necessary if a credit is applied for more than is available, creating a debit.

One the above is completed, the data is completely calculated and must now be formatted. The formatting code is designed to format each line into the format of the AUTOCASH template. The spread worksheet is given a GL code of 00040806, GL 901.1318 for all adjustments. This is hard coded. The GL accounts given to each line item on the Write Off sheet will be looked up in the table presented above. If a match is not found, the cell will be left blank to be added in post-execution. If the location is labeled as 1—the invoice is an unapplied payment from the Store Support Center, the program will cycle through each location in the GL table, rotating each time this event is triggered, effectively balancing the write offs for each location when this occurs. This is done since there is no sales adjustments/write off account for location 1, so unapplied payments with location 1 must be handled as described here.

After formatting is complete, a message prompts the user if they wish to add the AUTOCASH header onto the completed upload sheets. If a blank AUTOCASH template is present in the workbook before the Macro is run, the user should enter the tab number of its location so that Rows(1) to Rows(5) will be copied from this tab to Rows(1) to Rows(5) of the upload tabs. If no AUTOCASH template was available, the user should select CANCEL to skip this step and shall include the header in post-execution.

Once complete, the completed AUTOCASH tabs can be uploaded directly to JDE.

End Sub

**Guest Services CC Chargeback Blackline JE**

|  |  |
| --- | --- |
| OBJECTIVE | To compile credit card data from Guest Services and assemble a Journal Entry detailing weekly Credit Card chargebacks honored to customers. |
| INPUT | Refund data provided by Guest Services. Specific fields detailed in Remarks. |
| OUTPUT | A Blackline Journal Entry in upload format and a separate summary detailing Drive Card and Goodyear refunds. |
| SPECIFICITY | This document produces a very specific formatted output designed for CC refunds only. It is not recommended that this document is used for other applications. |
| PROCEDURES | One UserForm and a primary Subroutine are used in this document. |
| PASSWORD | CCProjects7069 |

**Remarks**

The application provided is used to automatically issue a Journal Entry upload detailing individual customer credit card accepted chargebacks against credit card GL credits originating from the bank statement. This process was formerly complete manually, with each Journal Entry requiring about 30-45 minutes to complete each week. The current process executes with a runtime of about 1 minute, including user input of initial parameters.

The input file required is individual refund detail with the following fields :

1. Medium of refund as String
2. Full name of recipient
3. Recipient US State code
4. Recipient store ID of original transaction
5. Refund gross amount
6. Refund GL code prefix/suffix

The output worksheets are detailed as follows :

General Entry – A Blackline JE that can be directly uploaded.

Summary –

1. A tabulated sheet of all individual Goodyear and Drive chargebacks, with California transactions isolated.
2. A dedicated space to tabulate transactions not processed and tie out with remaining variance.
3. A dedicated space to enter daily totals to tie out to Guest Services data.
4. A calculated space of totals by GL account that must tie out to the fields described in 3.

*It is known that a future process, to be automated from point of sale to the General Ledger is in the process of being created to replace this existing process. It is estimated however, that this process, of which replaces an even more manual process shall be in effect for roughly two more years.*

Notes on Specificity :

1. Journal Entry format is restricted to the format of a BlackLine Journal Entry.
2. Only pay types of MASTERCARD, VISA, AMEX, DISCOVER, Drive CC, and Goodyear will be recognized and added to the JE, identified by String.
3. GL account credits for each CC pay type are fixed according to the table listed in the procedures section.
4. GL account debits are determined only by the rules described in the procedures section.
5. Formatting for the Summary tab cannot deviate from what is produced by the existing Macro. The example given in the procedures section is the hard-coded format.
6. The JE line item detail cannot be changed from what is presented in the procedures section.
7. Only lines with GL 901.3016.3 will be marked for review.
8. Data presented in the Data Set must be homogenous; no gaps can exist.
9. Data must start at cell A1, with or without headers.

**Project Architecture**

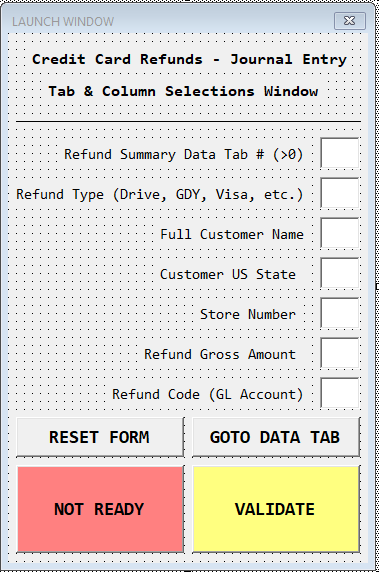
**The following depicts where each procedure is located within the project modules.**

FORMS

MODULES

PROCEDURES (*Private procedures omitted*)

RefundForm



RefundJE

Public Sub RefundJournalEntry()

**Overview of Procedures**

Public Sub RefundJournalEntry()

RefundForm.Show()

At this point the macro halts until control is passed back from the UserForm. The UserForm will not release control unless all requested values are validated or until the user clicks the “X” button, triggering the macro to run without processing data, ending immediately.

The required values must be entered into the UserForm :

1. Refund Summary Data Tab
   1. Must be greater than 0.
   2. Must represent a valid tab index.
   3. Must represent the index of a visible tab.
2. Column Selections
   1. Refund Type
      1. Field representing string containing method of refund.
   2. Customer full name
   3. State of customer address
   4. Store number of original transactions
   5. Refund gross amount
   6. Refund Code (GL) listed on Data Set

Column selections must be a number between 0 and 16384, or ALPHA code representing valid column location.

Once required values are entered, the Macro performs the actions below.

Creates two worksheets, titled “General Entry” and “Summary”. Avoid these names on existing sheets before running the Macro.

The General Entry template is formatted as below :

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Type\*** | | List - Text: | JE |
| **Explanation\*** | | List - Text: |  |
| **GL Date\*** |  | List - Date: |  |
| **Reversing\*** | | List - Text: | N |
| **Memo (BlackLine Only)** | | List - Text: |  |
|  |  |  |  |
| **Upl** | **Account String\*** | **Amount\*** | **Explanation 2** |
|  | List - Text | Value | List - Text |
|  |  |  |  |
| **Totals:** |  | **$ -** |  |

As many lines as necessary are inserted above the total to complete the JE.

This is the basic format for uploading a BlackLine Journal Entry.

The Summary tab is formatted as below :



All the yellow fields that are blank are intended to be manually entered from supporting data from bank statements and Guest Services reports to tie out to the Data Set results.

After basic formatting has completed, the Data Set is read, row by row, to be added to the Journal Entry.

GL codes for debits are processed according to the following rules. The GL code is split into parts divided by the delimiter “.”. These parts will be denoted by A, B, C, etc. The maximum GL format is AAAA.BBBB.CCCC but may be reduced to AAAA.BBBB or similar.

These rules are hardcoded into the Macro as the likeliness of changing these GL codes at Monro is very unlikely.

If only two parts A, B exist, Then

If part A is of the format “90#” then

GL = Existing Value

Else If part A is any other format then

GL = Store ID.BBBB

End If

Else

GL = Existing Value

End If

Each transaction is accounted for in one of the six credit card GL account credits populated in the Journal Entry. The rules for assignment are as follows :

|  |  |  |
| --- | --- | --- |
| Pay Type | GL for 49 US States | GL for California |
| VISA | 901.1150 | 907.1150 |
| MASTERCARD | 901.1150 | 907.1150 |
| AMEX | 901.1150 | 907.1150 |
| DISCOVER | 901.1150 | 907.1150 |
| DRIVE CC | 901.1110.0318 | 907.1110.0318 |
| GOODYEAR CC | 901.1110.0308 | 907.1110.0308 |
| CHECK REFUNDS | OMITTED | OMITTED |

The pay type is determined by reading the String present in the “Refund Type” column. Each String is read for *a partial match* containing the pay type :

{

“Visa” = VISA,

“Master Card” = MASTERCARD,

“American Express” = AMEX,

“Discover” = DISCOVER,

“Drive Card” = DRIVE CC,

“Goodyear” = GOODYEAR CC,

“check” = CHECK REFUNDS

}

Once each transaction is assigned, each row is pasted into the Blackline JE in the following format :

|  |  |  |  |
| --- | --- | --- | --- |
|  | AAAA.BBBB.[CCCC] *(GL code)* | Gross Amount | FIRST LAST STORE |

Each Goodyear and Drive Card transaction is also posted onto the summary sheet, separated by States vs. California. GL credit totals are also populated onto this sheet.

If at any point, a debit transaction is associated with GL account 901.3016.3, the line in the Journal Entry will be colored red, and a count of these lines will increase by one. The row indices of these lines will be listed in Columns(8). At the conclusion of execution, a message box will appear informing the user of how many lines are necessary to review.

End Sub

**CASHUP AR Subledger Reconciliation Tool**

|  |  |
| --- | --- |
| OBJECTIVE | Embedded within a template used for account reconciliations, a VBA Macro and UDF provide a tool to give additional insight on previous account balances when given a specific date. |
| INPUT | An unadulterated version of the PAYMENT WorldWriter with the following fields : Invoice Date (RPDIVJ), Invoice Number (RPDOC), Gross Invoice Amount (RPAG), Doc Type (RPDCT), Pay Type (RPDCTM), GL account (RPGLBA), and GL Date (RPDGJ). |
| OUTPUT | A new worksheet describing either A) Every open line item on account given a specific date or B) All dates in account history where the balance falls within a specified range. |
| SPECIFICITY | For any data set that represents JDE AR subledger data, this tool can be utilized. Data that is not consistent with how JDE represents AR information cannot be used. Available functions only convey data meaningful in the context of the JDE AR subledger. |
| PROCEDURES | One UserForm, one User-Defined Function (UDF) and a primary Subroutine are present in this document. |
| PASSWORD | CCProjects0464 |

**Remarks**

When completing complex account reconciliations for any account within an accounting ledger/subledger, it is sometimes necessary to view the account detail as it was in the past to understand why certain actions occurred. This can be notoriously difficult to do when given data from the JDE subledger, as there is no native feature to “retro” the account back in time. While this can be done with complex SUMIF formulas involving the GL dates, this method can be time consuming and may not be known to the user performing the reconciliation. Given here is a tool that does exactly as described, accessible to any user.

Additionally, the user has access to the function CHECKSEQ, which is used to detect sequences of invoices that add up to a specific total, such as a check total. The function operates by reading a specified integer of invoices directly above/below a given invoice line and sums each combination to compare against a check total given in the function.

For example, given the following invoices :

|  |  |  |
| --- | --- | --- |
| Invoice Number | Gross Amount | CHECKSEQ Result – Start in column B, looking for total $6,275.76, count 4 invoices. |
| 645988 | $ 148.57 | FALSE |
| 645989 | $ 873.89 | TRUE |
| 645990 | $ 5.12 | TRUE |
| 645991 | $ 781.10 | TRUE |
| 645992 | $4,615.65 | TRUE |
| 645993 | $ 56.20 | FALSE |
| 645994 | $ 99.52 | FALSE |

Because lines 2-5 sum to $6,275.76, the function returns TRUE. Notice that the four lines are consecutive, and their count is equal to 4 (the specified parameter). The user can filter by all the TRUE results to find all invoices that qualify for the specified total. In the context of reconciliations, this function can be used to isolate invoices that total exactly to a given check total, allowing the user to apply the check without a direct remittance.

Notes on Specificity :

1. Headers must be present in Rows(1) of Data Set. Data must be homogenous.
2. Documents with the same Invoice Number and Document Type cannot be treated as separate documents.
3. Invoice billing is hardcoded as “” value in Pay Type.
4. Due to the way JDE duplicates RU lines, certain RU lines are ignored depending on the task selected. This cannot be avoided. Only RU document types are treated in this way.
5. The only recognized Pay types belong to the set P = {“”, “RC”, “RS”, “RA”}

**Project Architecture**

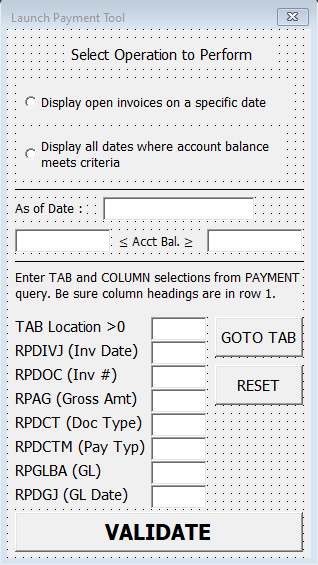
**The following depicts where each procedure is located within the project modules.**

FORMS

MODULES

PROCEDURES (*Private procedures omitted*)

F\_PaymentTool



PaymentTool

Public Function CHECKSEQ

Public Sub PaymentTool()

**Overview of Procedures**

Public Function CHECKSEQ(Check *As Range,* Target *As Range*, Base *As Integer*) As Boolean

Check is the cell which contains the value of the check total to find.

Target is the cell which contains the first invoice line item to start at (more accurately the center line item, because all combinations with line items above this line will be tested as well.)

Base is the number of invoices that will be tested against Check.

With this information, the function first tests the number of invoices above Target, such that Target is the last line item in the Data Set and compares the sum of these values against Check. If this sum does not match, the function continues by shifting this set down one line and re-checking the value.

If the sum matches the value of Check, TRUE is returned, otherwise FALSE is returned. If several solutions are found, TRUE is still returned.

This process continues until Target is the first line in the Data Set, thereby iterating every consecutive combination of the surrounding invoices for Target. Non numeric entries nearby Target will result in a #VALUE error. Check can be positive or negative; only the absolute value of Check is used.

End Function

Public Sub PaymentTool()

F\_PaymentTool.Show

The Macro presents the UserForm and requests the columns in the list below. All entries are validated before accepting.

Option : User must select which operation they would like to perform

1. “Display open invoices on a specific date”
2. “Display all dates where account balance meets criteria”

Option 1 :

As of Date : Must be a valid date, separated by “/” in the format mm/dd/yy[yy]

Tab Location : Must be greater than 0 and represent an existing, visible tab index.

Tab should be a download of every subledger entry for a specific account. This should include all billed invoices as well as payments, spreads, adjustments, and rebills.

Existing WorldWriter PAYMENT in Query Group 3 satisfies this requirement.

Columns : Must be between 1 and 16384 or an ALPHA code representing a valid column heading.

1. Invoice Date (RPDIVJ)
2. Invoice Number (RPDOC)
3. Invoice Gross Amt (RPAG)
4. Document Type (RPDCT)
5. Document Pay Type (RPDCTM)
6. GL Account (RPGLBA)
7. GL Date (RPDGJ)

Option 2 :

Lower Bound : Must be numeric and less than or equal to Upper Bound representing account balance.

Upper Bound : Must be numeric and greater than or equal to Lower Bound representing account balance.

Columns : Same as Option 1.

When valid entries have been submitted, the UserForm hides and the Macro continues execution.

The macro first detects when the first invoice was created. This date is used if Option 2 is selected.

If Option = 1 Then

Select all data starting at Rows(2) *Assuming headers are present*.

For Each Row In Selection

Read Pay Type column in Data Set. If this field = “” and GL date falls under specified date, save into memory and record original amount, doc type, and date.

*A Pay Type of “” indicates the billing of an invoice or creation of an unapplied payment.*

Next Row

For Each Row In Selection

If Pay Type = RC or RA and GL date falls under specified date, deduct, current line item from specified invoice. RU lines will be ignored, unless the value is positive, indicating that an RU is being used and should be included.

*A Pay Type of RC or RA indicates a payment or credit use is being applied against a given invoice or unapplied payment.*

Next Row

A new Worksheet is now created with the As of Date entered as the name.

With this sheet, the entries below are formatted :

Five columns :

1. Invoice #
2. Invoice Date
3. Doc Type
4. Days Old
5. Open Amount

Each invoice that was originally recorded will be pasted onto the next available line if and only if the resulting value of the gross amount is not equal to 0. A value of 0 indicates that the invoice was closed in full, and no credit remains open either.

All entries are then sorted by Invoice Date, starting from the oldest.

The total open amount is calculated and entered on the next available line.

The Macro finishes execution here.

If Option = 2 Then

*This operation will take about a minute to execute, depending on how much account activity has occurred. The evaluating date will be shown in the status bar to inform the user of the progress.*

Starting on the Data Set tab, all data is selected up to Rows(2) *assuming headers are present*.

The next section will iterate once for each date, starting with the oldest invoice date. The account balance will be calculated for each date until the current date of execution.

For Each Row In Selection

If the current Row has a Pay Type of RC, RS, or RA, the amount is added/deducted from the tabulated balance. Pay Types of “” that do not have a document of “RU” are also added to the balance.

*RU documents with Pay Types of “” are redundancies created by JDE and must be ignored.*

Next Row

The calculated balance is compared against the Upper and Lower Bounds given in the UserForm. If the balance falls within the range, the as of date used for this iteration is saved into memory and will be shown in the result.

This section now repeats for every date until the current date. The balance after each date is compared as above.

After all dates have been calculated, a new sheet is created using the upper and lower bounds as the name.

Two columns:

1. Date
2. Account Balance

Each date that qualified during data processing will be pasted onto the next available line, along with the balance that was calculated.

End Sub

**Aging Bad Debt Reserve Generator**

|  |  |
| --- | --- |
| OBJECTIVE | Identifies all unique account categorical types (customer type) present on any Aging download and partitions all line items based on user-defined groups. Isolates any account within a specific group based on certain balance criteria, supplied by the user. Primarily used to build the AR Reserve. |
| INPUT | Any subledger download with invoice detail including specific fields described in the Remarks section. |
| OUTPUT | One new tab for each specified group. Minimum of one group. One additional tab per group if the option was selected to isolate specific account by balance. One additional tab if the user saves their preferences. |
| SPECIFICITY | Designed to be used with any format of the Aging report. Designed to adapt to changes in customer types in the future. Accommodates changes in AR preferences when grouping account types or when deciding criteria for the reserve account. Ability to save and load previous preferences. |
| PROCEDURES | Four UserForms, six Class Modules, and one primary Subroutine are used in this document. |
| PASSWORD | CCProjects9457 |

**Remarks**

From an AR and Credit and Collections perspective, it is often proactive and effective to be aware of certain overdue accounts before the balances become large and problematic. A frequent operation that occurs on the AR aging is partitioning accounts by their Customer Type and isolating which individual accounts maintain a balance that fit certain criteria, determined by the user. Accounts that fall into this category can be contacted/monitored to prevent large changes in the company receivable. Eventually, such accounts will be posted to the “Bad Debt Reserve” GL, a holding account for balances expected to default.

Performing such an operation often requires a lot of keystrokes, filters, and formulas operated on large data sets. Such a task has been referred to as “A day of copy and paste”. The operation in question is also dynamically implemented, depending on the user. Each account representative is responsible for distinct accounts, often demanding varying degrees of analysis and examination.

The program created for this application can not only significantly reduce the time spent simply organizing the data into the desired structure but can also be adjusted to suit the specific needs of whoever may be operating it. Originally conceived to assist with the construction of the quarterly bad debt reserve, this application can be extended to any scenario in which past-due accounts must be identified.

Through a complex system of UserForms, the user can achieve the following :

1. Identify all possible Customer Types present
2. Partition the Aging Data Set into any assortment of groups containing specific Customer Types.
3. Choose to identify any account within a given group that satisfy one of the following :
   1. Maintain a total balance over a specified threshold.
   2. Maintain a balance over a specified number of days (inclusive)
   3. Maintain a balance both over a specified threshold and over a specified number of days.
4. Save the current selections for faster accessibility during next use.

An unlimited number of groups may be assigned, up to the total count of Customer Types.

The following fields in the Data Set are mandatory :

1. Customer Type (Unique Identifier that assigns account. Variable number of Customer Types may exist. Only one Customer Type may be assigned to each account.)
2. Customer Account Number
3. Customer Name (May be blank if not available.)
4. Invoice Total Open Amount
5. Invoice Amount Open 1-29 Days after Invoice Date
6. Invoice Amount Due 30-59 Days after Invoice Date
7. Invoice Amount Due 60-89 Days after Invoice Date
8. Invoice Amount Due 90-119 Days after Invoice Date
9. Invoice Amount Due 1210-149 Days after Invoice Date
10. Invoice Amount Due 150+ Days after Invoice Date

If some of the columns in items 5-10 are missing on the aging, the user can assign blank columns to these fields, as long as the following assumption is satisfied :

Total Invoice Open Amount = (Open 1-29) + (Open 30-59) + (Open 60-89) + (Open 90-119) + (Open 120-149) + (Open 150+)

For example, if only columns aging the invoice up to 60 days are available, the user can use proper columns for items 5 and 6 and assign blank columns to items 7-10.

Notes on Specificity :

1. No customer should have more than one Customer Type associated.
2. Invoice Open amount should be equal to the sum of the Date Bucket columns as described above.
3. Each customer’s account balance should be represented by the sum of the Invoice Open Amount of all line items under that account number.
4. Accounts isolated by Days are limited to day choices in multiples of 30, max 150.
5. Accounts isolated by Balance are limited to amount choices in multiples of 50.

**Project Architecture**

**The following depicts where each procedure is located within the project modules.**

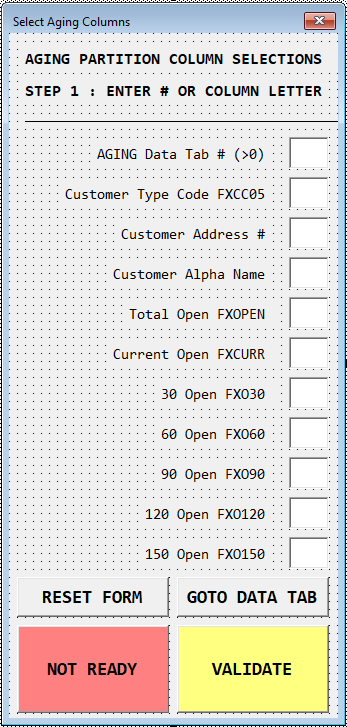
FORMS

CLASS MODULES

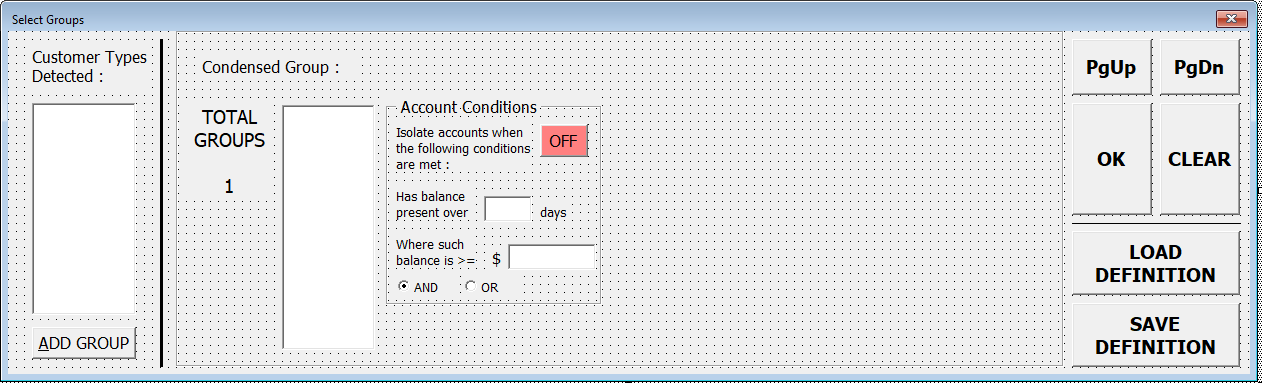
MODULES

PROCEDURES (*Private procedures omitted*)

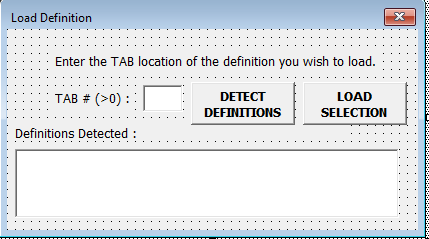
ColumnSelections



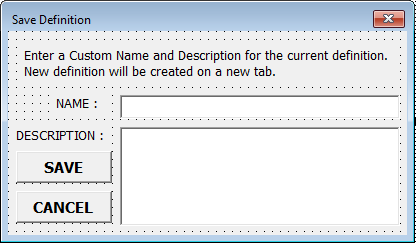
GroupSelections



LoadDef



SaveDef



AmtHandler

DaysHandler

DeleteHandler

InHandler

OutHandler

SwitchHandler

*(Class modules used to handle UserForm button events created at Runtime. The user need not be concerned with their existence.)*

Reserve

Public Sub Reserve()

**Overview of Procedures**

Public Sub Reserve()

ColumnSelections.Show

UserForm is shown prompting user to input locations of required fields. Data validation is used to ensure only valid inputs are received by the Macro.

All column selections must be either numeric between 1 and 16384, or ALPHA characters representing a valid column.

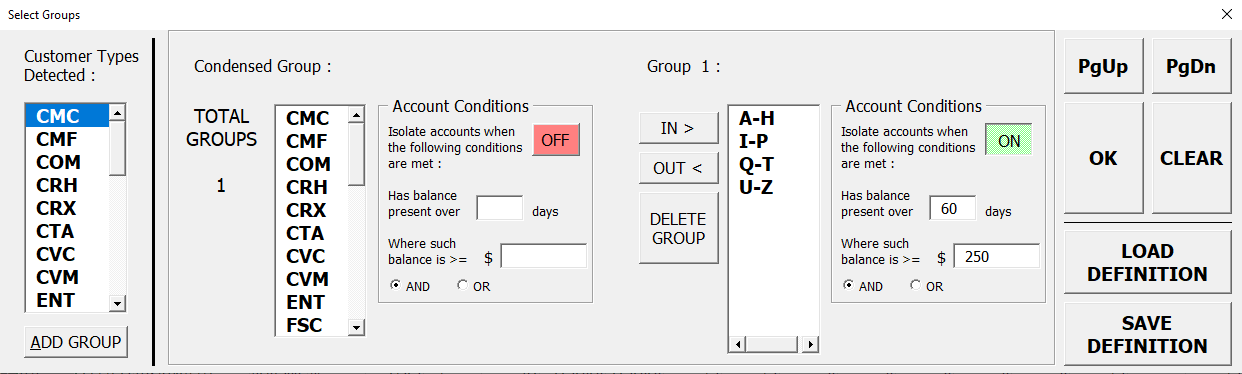
Tab location must be greater than 0 and represent an existing, visible tab.

Once valid entries have been submitted, the Macro continues executing the steps below.

All data in the Customer Type column is selected. The Macro identifies all of the unique codes present in this column. These unique codes are inserted into the next UserForm.

GroupSelections.Show

Here the user must now determine which customer types will belong in which groups. All of the available types detected will be present in the List “Customer Types Detected”. As the user adds groups, the ability to add any type will become available :



As types are added, they will be removed from the detected types box. Any types left unselected will be placed in the “Condensed Group” (Group 0), so that no type is omitted in the result.

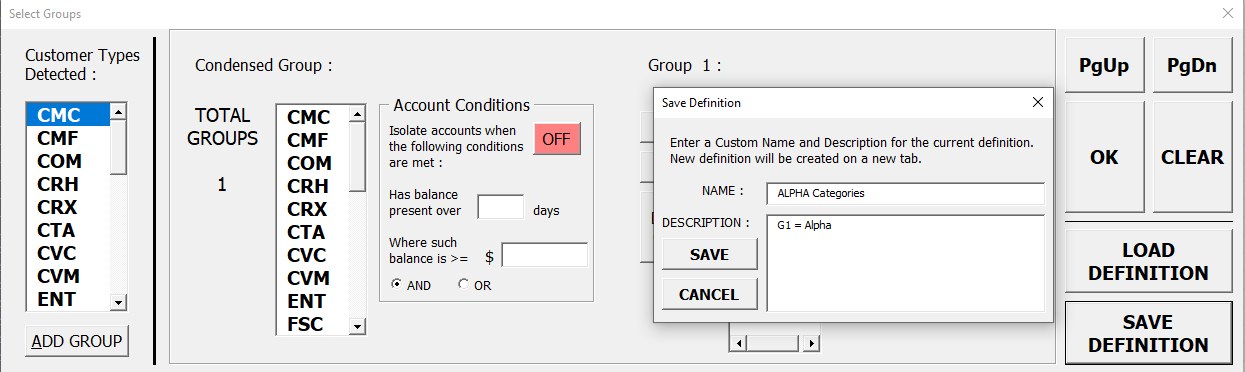
Groups left blank when the form is submitted will be ignored.

Day entries will automatically round to the lowest 30 days, maximum 150.

Amount entries will automatically round to the lowest $50.

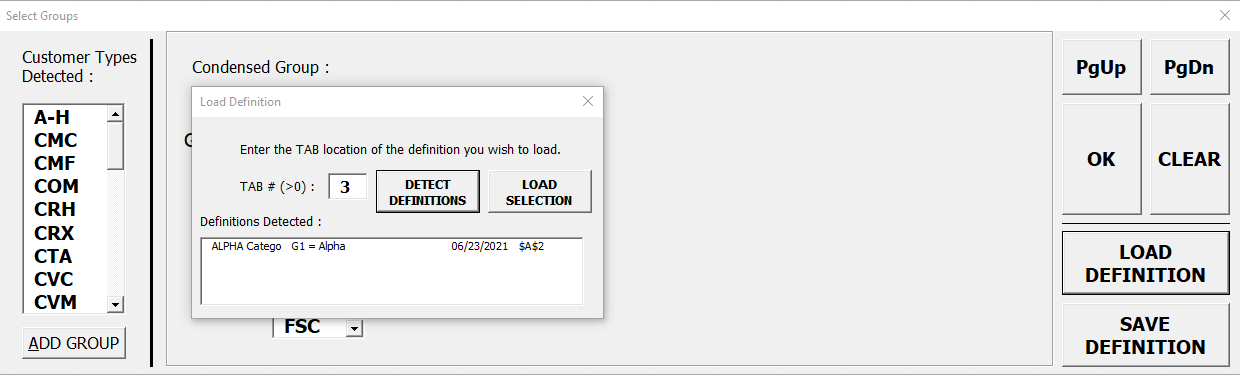
Default Option is AND, indicating accounts with a balance present past X days AND over amount Y will be isolated. Option OR will isolate accounts with a total balance over amount Y OR any balance present over X days.

Clicking the “SAVE DEFINITION” button will result in the SaveDef form to show :



The user may enter a custom name and description if desired. Upon clicking “SAVE”, a new tab will be created with the unique string detailing the current preferences. This tab can be loaded later to return to the preferences currently entered. Avoid tampering with the definition on the tab to avoid results unanticipated.

Clicking the “LOAD DEFINITON” button will result in the LoadDef form to show :



Entering the known TAB number of the definition will prompt the Macro to search for an existing definition. A definition will only be detected if the cell directly above = “<<<” and the cell directly below = “>>>”. The definition uses unique notation that can only be interpreted by this application.

Multiple definitions can be moved to a single tab, as long as the above criterion is satisfied. If multiple definitions are discovered, the user may select from a list.

Upon loading, the GroupSelections form is returned to by the Macro with the desired definition re-entered, dramatically reducing the number of keystrokes necessary to design the result.

Clicking “OK” twice will submit the current selections and prompt the Macro to execute.

A sheet is added for each group. Additional sheets are added for the isolated accounts. If the option was set to “OFF”, this second sheet is deleted as it will not be used. Each sheet is titled “GROUP N” where N is the group number. The second sheet for each group is titled “GN ACCOUNTS” where N is again the group number.

Headers are copied from the Aging sheet to each group sheet.

All data is selected by Row on the Aging sheet.

For Each Group

For Each Row In Selection

Each Row must be referenced to the user’s selections to determine which sheet to move the line item to.

If Customer Type present in Group

Move to Group Sheet

End If

Next Row

Next Group

The next section iterates through each group again and calculates which accounts should be isolated. Each group is only analyzed if the Conditions button for each group was set to “ON”.

For Each Group

Go to Group Sheet and select all data.

Calculate total balance by reading Invoice Open Column for each line, attach total to account.

Depending on the group Days selection, one or more columns for the Aging totals is/are read and calculated into the “conditional balance”. This total is also attached to the account separately.

For example, if 60 days was entered, the Macro will sum the value in column 60, 90, 120, and 150+ to get the total balance due after 60 days.

Once every line is read, the Macro goes to the second group sheet for isolated accounts. The Macro will paste each account recorded on the next available line if it meets the criteria selected :

If AND was selected :

1. Account balance after specified days must be over specified amount to be pasted.

If OR was selected :

1. Account balance after specified days must be greater than 0.
2. OR Account total balance must be greater than specified amount.

After each account is pasted, the current sheet is then formatted to be human readable.

Next Group

End Sub

**QUICK START GUIDE**

*Operation instructions for*

*applications outlined*

*in Documentation.*

**Reconciliation and Payment Upload v4.2**

*Objective : Prepare a daily reconciliation document used to key Wholesale invoices and deposits.*

1. Open template containing the title above.
2. Save As
   1. Save document in the format – “WWWW Name mm.dd.yyyy” where WWWW is the Warehouse or location ID and Name is the name of the location in text.
3. Click “MAKE HEADING”.
4. Enter name in cell B6.
5. Go to Sheets(10) “Invoice List”
6. Insert chosen Data Set containing open invoices. Column B should contain the invoice number, column C should contain the customer number, and column I should contain the invoice total. If the user wishes to alter the location of required fields, refer to the Documentation to alter the function arguments. Adherence to these guidelines warrants no change in the existing function arguments.
7. Document is now prepared for the keying process.

How to Key :

1. Enter relevant deposit totals in the GREEN fields on Sheets(1) “Reconciliation v4”.
2. For Each pay type tab :
   1. Enter the invoice number you wish to key. If the invoice total must be rounded or summed, follow the instructions from the table below :

|  |  |  |
| --- | --- | --- |
| SUFFIX As String | RESULT | EXAMPLE |
| “” | Returns original amount | 1110057 = 212.22 |
| + | Rounds up to nearest 1 | 1110057+ = 213.00 |
| ++ | Rounds up to nearest 5 | 1110057++ = 215.00 |
| / | Rounds up to nearest 10 | 1110057/ = 220.00 |
| \* | Rounds up to nearest 0.10 | 1110057\* = 212.30 |
| \*\* | Rounds up to nearest 0.25 | 1110057\*\* = 212.25 |
| -N ; Where N is an integer | Sums N invoices, including current line. No change to current line amount, only used for INVOICESM function. | 1110055 = 313.66  1110056 = 177.74  1110057-3 = 212.22  No rounding, INVOICESM affected. |
| -N+, -N++, -N/, -N\*, -N\*\* | Sums N invoices, including current line, and rounds SUM according to rules above. Current line is adjusted by amount required to obtain rounded sum value. Affects INVOICESM as well. | 1110055 = 313.66  1110056 = 177.74  1110057-3++ = 213.60  Sum of raw invoices is 703.62. To round to 705.00, last line is adjusted. INVOICESM affected. |
| .. | Autosum. INVOICESM affected. No rounding functionality available. | 1110055 = 313.66  1110056 = 177.74  1110057.. = 212.22  Sum of raw invoices is 703.62, reflected in INVOICESM. |

* 1. More information is available on Sheets(11) “README”

1. Extra Deposit information as well as Non-AR deposit information must be keyed manually.
2. When finished keying, work out any variances.
3. If the Check tab does not balance, the user may decide to use the “FIND MISSING CHECKS” macro to help deduce what has been missed.
   1. Key all check totals in a separate column on the check tab.
   2. Go to Sheets(1) “Reconciliation v4”
   3. Click “FIND MISSING CHECKS”
   4. Highlight all invoice totals and sum totals (two columns) on the check tab. Do not include errors below data.
   5. Highlight all check totals keyed off to the right side.
   6. Enter.
   7. All checks not found will be colored RED, as well as any invoice/sum of invoices that did not match any check total.
4. When all variances have been cleared :
5. Save document
6. Click “COMPLETE REC & ADD UNIQUE RECEIPTS” on Sheets(1).
7. Follow the prompts.
8. When the Macro finishes execution, save the document once more.
9. Reconciliation is completed.

**Wholesale COD Customer Aging Template**

*Objective : Perform minor write offs of qualifying invoices, perform matching spreads of invoices and credits on each account, apply remaining credits to oldest invoices on each account. Details described in Documentation.*

1. Open template containing title above.
2. Delete/rename previous result tabs that contain names in step 11.
3. Deposit any format of the AR aging Detail into **the first tab**. Aging must include all line items for each customer account. Aging should contain the following fields :
   1. Customer account number
   2. Invoice Date
   3. Invoice Document Type
   4. Invoice Open Amount
4. Organize aging so that data is sorted by account number (usually the default).
5. Go to VIEW > Macros > CODAging > RUN
6. Check/Uncheck all of the operations the user wishes to perform.
7. Enter Cutoff Date, separating mm, dd, yyyy into proper TextBoxes.
8. Enter column specifications
   1. Must be numeric or ALPHA representing valid column location
   2. If known accounts are present in the dataset that must be ignored, click “EDIT”.
   3. Enter the desired account number and an optional reason. Click “Add Account”. This account will now be ignored.
   4. Click “Exit…”
   5. Click “RUN REPORT”
9. Click “OK”
10. Wait until the Macro finishes execution. The Status Bar will return to blank when the Macro is finished.
11. Tabs titled “MINOR WRITE OFFS”, “MATCHING CREDITS”, “APPLIED CREDITS” are the results of the Macro.

**Aging Analysis Template v2**

*Objective : Remove obsolete Late Charges, perform minor write offs, spread matching credits, identify all accounts to be removed off of Credit Hold. Details described in Documentation.*

1. Open template containing the title above.
2. Delete/rename previous result tabs with names present in step 14.
3. Keep a blank AUTOCASH template in the document, if user so wishes to include the header.
4. Deposit any format of the AR aging Detail into any blank tab. Aging must include all line items for each customer account. Aging should contain the following fields :
   1. Customer account number
   2. Invoice Document Type
   3. Invoice Number
   4. Invoice Date
   5. Invoice Due Date
   6. Invoice Open Amount
   7. Invoice Gross Amount (optional)
   8. Invoice Business Unit RPMCU (optional)
   9. Invoice 3 Digit BU RPSFX (optional)
   10. Invoice 5 Digit BU RPKCO (optional)
5. If the user wishes to perform the function “Find all accounts to remove off credit hold”, then another data tab must be deposited on any blank tab. This second data set must contain a list of every AR customer including the following fields :
   1. Customer account number
   2. Customer Temporary Credit Message
6. Go to View > Macros > AgingAnalysis > RUN
7. Check/Uncheck all of the functions the user wishes to perform.
8. Enter in the relevant tab indices for each required data set.
   1. Indices should be greater than 0 and represent a valid, visible tab.
9. Enter in the relevant column locations for each required field.
   1. Must be numeric from 1 to 16384 or ALPHA representing column code.
   2. GOTO buttons are included to bring the user to the data tab, should they need to view the columns.
10. Click “RUN REPORT”
11. Fix any errors in the entries submitted.
12. Click “CONFIRM RUN”
13. Wait until the Status Bar returns to blank. Operations will be described during execution.
14. Result tabs : “RF CHARGES TO REMOVE”, “MINOR WRITE OFFS”, “SPREAD UPLOAD”, “ACCOUNTS TO TAKE OFF HOLD”, “ACCOUNTS TO PUT ON HOLD”, “ACCOUNTS MISSING ON CUST DETAIL”.
15. For tab “MINOR WRITE OFFS”, GL accounts must be manually entered or referenced from another data table to finish the AUTOCASH. The Macro does not support functionality for this.
16. The header for the first five lines of the Spread and Minor Write Off AUTOCASH must be manually copied from an existing template. The Macro does not support functionality to automatically populate this.

**CALTRANS TEMPLATE**

*Objective : Construct a “Consolidated Invoice” compiling all Monro/dba Brand invoices for a specific customer account, including comprehensive invoice information.*

1. Open template containing the title above.
2. Delete the previous sheet titled “FINISHED DOCUMENT”.
3. Go to MONRO Queries. Edit Query titled CALTRAN > Select Records.
4. Change the field IKDATE to lie between the dates you wish to create the Consolidated Invoice for.
5. Save and run query definition. (temp/caltran) *no “s”*
6. Go to AR WorldWriter (Q03).
7. Do not edit, only run the WW named CALTRANS. (temp/Caltrans) *with “s”*
8. Download both the query and WW into separate tabs in the template.
9. The query is simply a download of the F65100 and F65101 files to list out every line item for each invoice ever billed within the specified period. The required fields are :
   1. Store Number
   2. Store Invoice Number
   3. Invoice Gross Amount
   4. GS/AN Requirement code (PO can be used instead if not available)
   5. Total Invoice Tax
   6. Total Invoice Taxable Amount
   7. Invoice Vehicle Make
   8. Invoice Vehicle Model
   9. Invoice Vehicle Mileage
   10. Invoice Vehicle License Plate
   11. Invoice Vehicle VIN
   12. Invoice Line Item, Item Number
   13. Invoice Line Item, Item Description
   14. Invoice Line Item Service Writer
   15. Invoice Line Item Parts Price
   16. Invoice Line Item Labor Price
   17. Invoice Line Item Quantity

The WorldWriter is simply a download of the AR open invoice file. This file is used to determine which of the invoices on the query are actually open. The required fields are :

* 1. AR Invoice Number
  2. AR Invoice Date

1. Go to View > Macros > CalTrans > RUN
2. Check/Uncheck the boxes at the top to confirm if the queries used have headers in Rows(1) or not. If the data starts at Rows(2), leave the box checked.
3. Input proper tab indices for both the Store Query and AR WorldWriter data. Tab indices must be greater than 0 and represent a valid, visible tab.
4. Input all the required column locations for the fields described above. Column codes must be numeric between 1 and 16384 or ALPHA code representing a valid column location.
5. Enter the begin and end date of the period you wish to include invoices from.
6. Select the DBA Brand logo you wish to include in the header beneath the MONRO corporate logo.
7. If further assistance is required, click the “HELP!” button.
8. Click “CONFIRM” and then “RUN” to execute the Macro with your selections.
9. Finished Product will be listed on tab “FINISHED DOCUMENT”. You may print immediately to PDF or paper copy.

**Wholesale +/- $10 Write Off Template v2**

*Objective : Identify all accounts with a balance between -$10 and $10 (exclusive) and write off existing balance. A spread is required first, and then a write off of a single invoice.*

1. Open the template containing the title above.
2. Delete/rename the previous result sheets titled “$10 SPREAD” and “$10 MINOR WRITE OFFS”.
3. Keep a blank AUTOCASH template in an existing tab before running the Macro.
4. Download any format of the AR Aging Detail and deposit it into a new tab or replace the old data. The required fields in the download are :
   1. Customer Account Number
   2. Invoice Number
   3. Invoice Date
   4. Business Unit RPMCU
   5. 3 Digit BU RPSFX
   6. 5 Digit BU RPKCO
   7. Document Type
   8. Gross Amount
   9. Open Amount
5. Go to View > Macros > WriteOff10 > RUN
6. Keep the checkbox at the top checked.
7. Enter the index of the aging detail tab. The index must be greater than 0 and represent a valid, visible tab.
8. Enter in the column locations for all of the required fields described above. Column selections must be numeric from 1 to 16384 or ALPHA code representing a valid column location.
9. Enter the desired GL date to record on the AUTOCASH.
10. Click “RUN REPORT” then “CONFIRM RUN”
11. Wait until the Macro finishes execution. The user will know that the Macro has finished when the Status Bar at the bottom returns to blank.
12. The Macro will prompt the user to add the header from the AUTOCASH template to the finished sheets. Enter the valid tab index of the AUTOCASH template or click “Cancel” if the user need not add the header at this time. The header is imperative for upload into JDE.
13. The finished sheets will be titled “$10 SPREAD” and “$10 WRITE OFFS”. These sheets can now be uploaded to JDE.

**Guest Services CC Chargeback Blackline JE**

*Objective : Create a summary and Blackline Journal Entry detailing individual credit card chargebacks matched to GL credit card bank credits. Data is provided from Guest Services.*

1. Open the template containing the title above.
2. Delete/rename the previous result sheets titled “General Entry” and “Summary”.
3. Deposit on a new tab (or overwrite an existing tab) data received from Guest Services detailing CC chargebacks recently processed. The required fields are :
   1. Refund Type (MC, VISA, AMEX, Discover, Check, Drive, GDYR). These types can be embedded in a string such as “Credit to your Visa account”. The Macro will still detect that this is a VISA transaction, so no manipulation is necessary.
   2. Customer Full Name
   3. Customer US State
   4. Store Number
   5. Total Chargeback Amount
   6. Refund Code (GL code or partial GL code).
4. Go to View > Macros > RefundJournalEntry > RUN.
5. Enter the index of the Guest Services data tab. The index must be greater than 0 and represent a valid, visible tab.
6. Enter the column locations of the fields described above. Column selections must be numeric between 1 and 16384 or ALPHA code representing a valid column location.
7. Click “VALIDATE”
8. Click “LAUNCH”
9. Wait for the Macro to finish execution. A message will appear describing how many JE lines need to be reviewed due an unacceptable GL code. These lines will be listed on the “General Entry” tab.
10. Ensure that the JE total is 0.
11. The sheet “General Entry” can be directly uploaded to Blackline.
12. Use the “Summary” sheet to tie out Drive card and Goodyear card totals to GL totals before submitting the JE. Yellow fields on the “Summary” tab are intended for manual input of data.

**CASHUP AR Subledger Reconciliation Tool**

*Objective : A tool embedded in a reconciliation template used to retro customer account data “back in time” to view open invoices on a specific date. Also includes a function used for tying out check totals.*

1. Open the template containing the title above.
2. Prep the document as you would to complete an account reconciliation. This is not needed to operate the Macro properly but is usually present for the context of this application.
3. Download the “PAYMENT” WorldWriter into the tab labeled “PAYMENTS”. Ensure headers are present in Rows (1). The specifications of the WW should be :

Required Output Fields :

Invoice Date RPDIVJ

Invoice Number RPDOC

Gross Amt RPAG

Document Type RPDCT

Pay Type RPDCTM

GL Acct RPGLBA

GL Date RPDGJ

Appropriate Data Selection :

AND Address Number VALUES [Account # to isolate]

Sequence 10 ABAN8

1. Go to View > Macros > PaymentTool > RUN
2. Select which operation to run.
3. Enter the As of Date for Option 1 to show all invoices open on that date in time. For option 2, enter upper and lower bounds for the account balance to isolate every day that had a balance in that range.
4. Enter the tab location of the Payments WW. Tab index should be greater than 0 and represent a valid, visible tab.
5. Enter all the column locations for each required field. Column indices should be either numeric from 1 to 16384 or ALPHA representing a valid column code.
6. Validate > Confirm and Run.
7. A new tab will be created and titled with the given As of Date/LU Bounds. This tab contains the results.

Function CHECKSEQ(Check As Range, Target As Range, Base As Integer) As Boolean

This function can be used to determine if a consecutive run of invoices below or above a designated center invoice sums to a specified total, such as a check total.

1. “Check” is the cell reference of the check total you wish to sum to. This can be positive or negative: the absolute value will be used.
2. “Target” is the cell reference of the center of the invoices to evaluate around.
3. “Base” is an integer describing how many invoices should total to “Check”.

Function will return TRUE or FALSE depending on if a match was found. Fill down this function on all possible candidates to isolate all of the invoices that sum to the Check total.

**Aging Reserve Generator**

*Objective : With an Aging Detail report as input, identify all possible Customer Types and allow the user to partition each type into a user-designated “Group” containing the detail for each account that qualifies for that Group. Allow the user to specify for each group, account conditions that will isolate specific accounts in that group that meet balance criteria based on amount and/or days overdue. Used to build the Bad Debt Reserve GL.*

1. Open the template containing the title above.
2. Run the AR Aging Detail Report via another program or WorldWriter.
3. Drop the Aging Report into any tab. Ensure that headers are in Rows(1).
4. Delete/rename any tab with a name in the format “GROUP X” or “GX ACCOUNTS”. These are old result tabs that must be deleted/renamed before running the Macro.
5. Ensure that the Aging detail has the following fields :
   1. Customer Type (Critical)
   2. Customer Number (Address Number)
   3. Customer Name
   4. Total Open Amount
   5. Current Open Amount
   6. >=30 Days Open Amount
   7. >=60 Days Open Amount
   8. >=90 Days Open Amount
   9. >=120 Days Open Amount
   10. >=150 Days Open Amount

Note : If the downloaded Aging Detail does not contain aging buckets all the way up to 150 days, columns can be omitted as long as the following statement holds for each line item. Use blank columns to represent missing buckets :

Total Open = Current + >=30 + >=60 + >= 90 + >= 120 + >=150

Corollary : When specifying account conditions (described below), isolating accounts by Days Overdue can only be as accurate as the maximum Days column specified for the aging buckets described above. For example, if only columns 30-90 are present, accounts can only be isolated by having a balance up to 90 days overdue, no greater. Attempting to isolate accounts that are 150 days overdue will be meaningless, as this information cannot be conveyed when only columns 30-90 are present.

1. Enter the tab index of the Aging Detail. The index should be greater than 0 and represent a valid, visible tab.
2. Enter all of the required column locations for the Aging. Column locations should be either numeric from 1 to 16384 of ALPHA representing a valid column code.
3. Validate > Launch
4. User will now be shown the Group Selections Window. Use this window to create the custom Groups for the detected Customer Types.
5. If the user wishes to load a previous definition, select “LOAD DEFINITION”.
   1. Enter the tab index of the saved definition. Click “DETECT DEFINITIONS”
   2. If a definition exists, it will appear below. Select which definition line to load.
   3. Once highlighted, click “LOAD SELECTION”.
   4. The definition will now be loaded. Further changes may still be made.
6. To Create a Group :
   1. Click Add Group
   2. Select desired Customer Type to include from window under “Customer Types Detected”.
   3. Click IN > next to the Group to add.
   4. Repeat b-c for each type desired for the current group.
   5. If the user wishes to isolate accounts in the current group, leave the switch in the “ON” position (Green).
   6. Enter the number of days an account must be aged in order to be isolated. For example, an entry of 60 will isolate all accounts with a balance over 60 days. Intervals of 30 days only.
   7. Enter the balance amount required for an account to be isolated. For example, an entry of 5000 will isolate all accounts with a balance of $5000 or greater. Intervals of $50 only.
   8. Select AND to ensure that both conditions must be met for an account to be isolated. Select OR if only one condition must be met.

Possible Templates :

Isolate all accounts that satisfy or exceed a balance :

Days : 0

Amt : Desired Balance

Opt : AND

Isolate all accounts with a balance over X days :

Days : X

Amt : 0

Opt : AND

Isolate all accounts with a balance over X days where such balance is greater than Y.

Days : X

Amt : Y

Opt : AND

Isolate all accounts with either any balance over X days or a *total* balance greater than Y.

Days : X

Amt : Y

Opt : OR

* 1. Current Group is now finished. Repeat steps a-h to create another Group. Page down as necessary to view all groups.
  2. Group 0 will be the Condensed Group of all of the Customer Types not placed in any other group. Default conditions are off but can be altered.

1. Once all groups are created, the user can decide whether or not to save these group selections for the next run or not. If the user wishes to save their preferences, click “SAVE DEFINITION”.
   1. Enter an optional NAME and DESCRIPTION for the definition and click “SAVE”.
   2. The resulting definition will have its own tab created with the necessary data to load for a later execution. Do not alter the data string.
2. After saving, the application can be run. Click OK > RUN to execute the program with current selections.
3. The Macro will describe in the Status Bar of its progress. A message will prompt the user when the program has finished execution.
4. Result tabs will be named “GROUP X” and “GROUP X ACCOUNTS” for each index X entered in the Group Selections Window.

