EB Scheme UW Application – Documentation and user guide

Contents

[Workbooks - Overview 4](#_Toc478387521)

[Opportunity 4](#_Toc478387522)

[Rating 4](#_Toc478387523)

[Utility 4](#_Toc478387524)

[Reporting 4](#_Toc478387525)

[General AddIn 5](#_Toc478387526)

[Modules 5](#_Toc478387527)

[Class Modules 7](#_Toc478387528)

[Workbooks - Detailed 10](#_Toc478387529)

[Opportunity 10](#_Toc478387530)

[Worksheets 10](#_Toc478387531)

[Forms 11](#_Toc478387532)

[Modules 11](#_Toc478387533)

[Utility 14](#_Toc478387534)

[Worksheets 14](#_Toc478387535)

[Forms 14](#_Toc478387536)

[Modules 14](#_Toc478387537)

[Reporting 17](#_Toc478387538)

[Worksheets 17](#_Toc478387539)

[Modules 17](#_Toc478387540)

[Rating 18](#_Toc478387541)

[Worksheets 18](#_Toc478387542)

[Forms 19](#_Toc478387543)

[Modules 19](#_Toc478387544)

[Transactions 25](#_Toc478387545)

[Opportunity Transaction/Status 25](#_Toc478387546)

[Quote Transaction/Status 26](#_Toc478387547)

[DB Diagram 27](#_Toc478387548)

[Release Instructions 28](#_Toc478387549)

[Add-In Installation Instructions 28](#_Toc478387550)

[Postlist Upload Instructions (Utility Workbook – Opportunity/Quote) 28](#_Toc478387551)

[Generali Excel VBA Standards 30](#_Toc478387552)

[Naming Standards 30](#_Toc478387553)

[Classes 30](#_Toc478387554)

[pOpportunityID 30](#_Toc478387555)

[Language 30](#_Toc478387556)

[Worksheets 30](#_Toc478387557)

[Code Modules 31](#_Toc478387558)

[Standard Modules 32](#_Toc478387559)

[VBA Cheat Sheet 33](#_Toc478387560)

[Developer Window 33](#_Toc478387561)

[Recording Macros 33](#_Toc478387562)

[Controls 33](#_Toc478387563)

[Dot Notation 34](#_Toc478387564)

[Subroutines and Functions 34](#_Toc478387565)

[Subs 34](#_Toc478387566)

[Functions 34](#_Toc478387567)

[Application.WorksheetFunction.function vs Application. function 35](#_Toc478387568)

[Variables 35](#_Toc478387569)

[Conditional Logic 36](#_Toc478387570)

[Case statements 36](#_Toc478387571)

[Operators 36](#_Toc478387572)

[Loops and Iteration 37](#_Toc478387573)

[For Loops 37](#_Toc478387574)

[While Loops 37](#_Toc478387575)

[Do While 37](#_Toc478387576)

[Do Until 37](#_Toc478387577)

[Basic Mathematics 37](#_Toc478387578)

[Modules and code structure 38](#_Toc478387579)

[Comments 38](#_Toc478387580)

[Excel interaction 38](#_Toc478387581)

[Excel syntax within VBA 38](#_Toc478387582)

[Debugging VBA 38](#_Toc478387583)

[Useful Excel skills 38](#_Toc478387584)

[Advanced VBA 39](#_Toc478387585)

[Interacting with SQL 39](#_Toc478387586)

[Arrays 41](#_Toc478387587)

[Working with Files 41](#_Toc478387588)

[Dates in VBA 42](#_Toc478387589)

[Using Buttons 43](#_Toc478387590)

# Workbooks - Overview

## Opportunity

* Used for creating Opportunities, Quotes (Quote Lite) and Chasers
* Can create Underwriting Notes for Opportunities
* Can be used to progress and manage an Opportunity from Initial Checks stage through to chasing and eventual outcome
* Stores all data on a SQL database – no data stored on spreadsheet

## Rating

* Used for creating detailed quotes with full rating info
* Able to automate calculations and validations relevant to Quotes
* One workbook per Quote (per Opportunity)

## Utility

* Used to upload (insert or update) associated data to the database, such as Rate and Broker lists
* Can add new Users to the system to enable them to access the collection of workbooks
* Can import Opportunities and Quotes from the existing Postlist workbook

## Reporting

* Provides a number of reports to aid business development

# General AddIn

## Modules

|  |  |
| --- | --- |
| M01 General | This module contains lots of general subs and functions which cam be used throughout the Workbooks and modules |
| PerformanceStart() | Call this sub at the beginning of a module to improve its performance |
| PerformanceEnd() | Put this function at the end of the end of a module to improve performance |
| AddBlankTableRow() | This subroutine creates a blank row on an Excel table of specified height |
| AddTableRow() | This function adds a row to an Excel table. If the table has only one row data will be added to that row |
| DeleteTableRows() | Deletes all rows from an Excel table |
| ClearTableFormatting() | clear cell fill, font bold and italic from a table |
| ClearUpData() | On exit of the spreadsheet, reset Hidden Data sheet |
| AutofitCellHeight() | Changes the height of a cell to allow word wrapped text to be displayed in full in a cell. |
| AuditError() | Manages populating and clearing of AuditError table on Hidden Data sheet |
| GetWorksheetName() | get the name of the sheet a table appears on |
| IsAlphabetic() | This function tests whether the string passed in is alphabetic |
| TOSplit() | This function returns a particular word in a sentence, first word is 0 |
| TOCountString() | This function counts the number of words in a string |
| StringToInt() | Turns a numeric string into an integer, or 0 if not possible |
| GetNextWorkDay() | Calculates next working day, ignoring weekends |
| VBAtoSQLFormatDate | Convert date between VBA and SQL |
| VBAtoSQLFormatDateTime | Convert date between VBA and SQLwhen sticking in a basket |
| VBAtoSQLUpdateFormatDate | Convert date between VBA and SQL for an UPDATE query |
| VBAtoSQLUpdateFormatDateTime | Convert date between VBA and SQL for an UPDATE query |
| SQLtoVBAFormatDate | when getting from SQL to VBA, works with Dates and datetimes |
| SQLtoXLSFormatDateTime() | when getting from SQL to Excel, works with Dates and datetimes |
| VBAtoXLSFormatDate | when getting from VBA to VBA, works with Dates |
| VBAtoXLSFormatDateTime | when getting from VBA to Excel, works with DateTimes |
| SQLFriendlyString | cleanse input string ready for entering into DB |

|  |  |
| --- | --- |
| M02\_GeneralClass | This module contains details needed for Class which can't go in the class modules |
| InstantiateApplicationControl |  |
| InstantiateChaser |  |
| InstantiateBroker |  |
| InstantiateOpportunity |  |
| InstantiateRateBlock |  |
| InstantiateUser |  |
| InstantiateUWNote |  |
| InstantiateQuote |  |

|  |  |
| --- | --- |
| M03\_ApplicationControl | This module contains all set up for Application Control |
| SetupApplicationControl() | Manages initial connection to SQL DB |

|  |  |
| --- | --- |
| M04\_Audit | This module contains details needed for Audit Table functionality |
| InsertAuditRecord () | This inserts a record to the audit table, called when an Opportunity or Quote status is updated |
| GetAuditUserID | This function finds the user who performed a particular transaction |
| GetLastTransDate | This function tests if a particular transaction has occurred and is used to validate if a subsequent transaction is valid. It returns the date when the transaction occurred or 0 |
| CancelTransactions() | This subroutine cancels transactions which are no longer valid, due to further activity eg. a declined transaction is no longer valid if it has been undone. |
| OAClearAuditRecords() | removes all Transactions from the Audit Excel table |
| OAPutRecordsToAuditTable() | Download all the Transactions from the DB and place them inside the Excel table |

|  |  |
| --- | --- |
| M05\_Broker | This module contains code about Brokers which isn't part of Class |
| PopulateBrokerComboBoxFromSQL() | populates combo box with a list of broker names from DB |
| GetBrokerName | returns the broker name from the ID |
| GetBrokerID | returns the brokerID from the name |

|  |  |
| --- | --- |
| M06\_Chaser | This module contains details needed for Chaser Table functionality. Although Chaser is a class, some modules are outside the class as they act on more than one opportunity |
| OCClearChaserRecords() | removes all entries from the chaser table in Excel |
| OCPutRecordsToChaserTable() | Download all the Transactions from the DB and place them inside the Excel table |

|  |  |
| --- | --- |
| M07\_Opportunity | This module contains details needed for Opportunity List (OL) Table functionality. Although Opportunity is a class, some modules are outside the class as they act on more than one opportunity |
| OLClearOppRecords() | removes all entries from the Opportunity\_Table in excel |
| OLSelectAndLockOpportunity() | allows selecting of an Opportunity and controls locking/unlocking of Opportunities |
| OLUnselectOpportunity() | unlocks the currently selected Opportunity |
| OLPutRecordsToOppTable() | Download all the Transactions from the DB and place them inside the Excel table called Opportunity\_Table. |
| OpportunityIssuedStatusUpdate() | Update Issued Status |

|  |  |
| --- | --- |
| M08\_Quote | This module contains details needed for Quote List Table functionality. |
| QLPutRecordsToQuoteTable | Function to download all the Quotes from the DB and place them inside the Quote List Excel table |
| QLClearQuoteRecords() | Clear the quote list Excel table |
| QLUnselectQuote() | De-select the record in the hidden data |
| AreQuotesReadyToBeIssued | function to test whether there are some quotes ready to be sent |
| UpdateQuotesStatuses | Function to update the status of each quote under an Opportunity |
| DetermineUpdateQuoteStatus | Function to update the status of an individual quote |
| GetMaxQuotes | Gets the number of quotes under a particular opportunity |
| GetMaxInitialQuotes | Gets the number of initial quotes under a particular opportunity |
| OpenQuoteCount | Get the number of open Quotes under a particular opportunity |
| GetMaxQuoteVersion | Gets the number of entries for a quote under a particular opportunity |
| WithdrawOtherQuotes() | Withdraw any other quotes which have been found with the same original quote number |
| CreateNewQuote | Creates a new Initial Quote from the Opportunity Quote list, Quotelite or rating spreadsheet |
| CopyQuote | Creates a revised quote based on initial quote number |
| GetLastUpdated | Function which checks DB for LastUpdated value for Quote |

|  |  |
| --- | --- |
| M09\_User | This module contains details needed for User functionality |
| IsUserAuthorised | looks if a user is valid to carry out a certain transaction |
| PutRecordsToUserHD | Populates the list of users on the Hidden data sheet |
| GetUserID | function to return UserID from Username |
| GetUsername | function to return Username from ID |

|  |  |
| --- | --- |
| M10\_UWNote |  |
| UWNotePopulateTable | populate table of UW notes |
| IsUWNoteOutstanding | checks for incomplete UW note |

## Class Modules

|  |  |
| --- | --- |
| CAapplicationControl | Class module for ApplicationControl |
| IsConnectionValid() | Function to check whether server connection is valid |

|  |  |
| --- | --- |
| CBroker | Class module for Brokers |
| GetBroker() | get a record from the utility screen |
| PutBroker() | put a record to the utility screen |
| UploadBroker() | upload a record to the DB, insert or update as appropriate |
| InsertBroker() | insert a record to the DB |
| SelectBroker() | select a record from the DB |

|  |  |
| --- | --- |
| CChaser | Class module for Chaser |
| GetInputFromOC() | Get the information from the top half of the Opportunity Chasing screen |
| Validate | validate record (function, returns error) |
| Insert () | save to the database |
| PutToOC() | Put details to the Chaser\_Table after a new 'chase' has been made |

|  |  |
| --- | --- |
| CMember |  |

|  |  |
| --- | --- |
| COpportunity | Class module for Opportunity |
| GetInputFromIO() | get details from the Opportunity Initial Checks screen |
| PutToIO() | Put details to the Opportunity Initial Checks screen |
| PutToIC() | Put details to the Rating Initial Checks screen |
| PutToOC() | Put details to the Opportunity Chasing screen |
| Insert () | Insert details to SQL DB |
| Update() | Update details to SQL DB |
| UpdateFromOC() | updates opportunity details which can be updated on the Chase details screen |
| UpdateStatus | This module updates the status of the Opportunity, then writes to the audit table, then ripples down status to quotes |
| UpdateUnderwriter() | This subroutine updates the allocated Underwriter |
| SelectDetails | Select the details from the SQL DB |
| OpportunityLocker | This sub checks the Opportunity is not locked and then locks it, if its already locked it will return a message saying so |
| OpportunityUnlocker() | This subroutine unlocks an Opportunity |
| UpdateIssuedStatus | Update issued status of an Opportunity |
| PLGetOpportunity() | Get the Opportunities from the Opportunity View Table |
| PLUploadOpportunity() | upload a record to the DB, insert or update as appropriate |
| PLInsertOpportunities() | Insert details to SQL DB |
| GetInputFromNT() | Gets info from the Notes screen |
| PutToNT() | Puts opportunity data to the Notes screen |

|  |  |
| --- | --- |
| CQuote | Class module for Quote |
| GetNext | Create next initial quote, increments and sets quote number |
| GetInputFromGQ() | Sub to Get the details from the GIP quote screen |
| GetInputFromRating() | Sub to Get the details from the GIP quote screen |
| Insert() | Save the new quote to the database |
| Update() | Update an existing quote to the database |
| UpdateUnderwriter() | Update an existing quote to the database |
| SelectDetails() | get data from database |
| PutToGQ() | puts the instance of the Quote to the GIP Quote Screen |
| PutToRating() | puts the instance of the Quote to the GIP Quote Screen |
| Copy() | Create a new quote copying an existing |
| Validate | Function to validate details entered in a quote, it returns a string containing error message |
| UpdateStatus() | This module updates the status of the Quote and then writes to the audit table |
| RevertStatus() | rollback the status of the quote to the previous one (used when taking a quote off hold) |
| PrivateUpdateStatus() | update quote status (when no audit record required) |
| GetPreviousTransaction() | Gets the Quote status currently stored in DB so we know what to revert back to if Quote is put on hold |
| PLGetQuotes() | Get the Quotes from the Postlist Quote View Table |
| PLInsertQuotes() | Insert details to SQL DB |

|  |  |
| --- | --- |
| CRateBlock | Class module for RateBlock |
| GetRateBlock() | Get the Rate Block from the Utility screen |
| PutRateBlock() | Put the Rate Block back to the Excel sheet Utility Screen |
| InsertRateBlock() | Insert a block of rates to SQL server |
| DownloadRateBlock() | Download the chosen rate block from the DB |
| DeleteRateBlock() | Delete a block of rates from the SQL Table |

|  |  |
| --- | --- |
| CUser | Class module for User |
| GetFromUU() | Get the Username from the Utility Screen Table |
| GetFromRS() | Get an instance of a username from a Recordset |
| PutToUU() | Put the Username back to the Utility Screen Table |
| PutToHD() | Puts the data back to the hidden data screen |
| InsertOrUpdate() | upload a record to the DB, insert or update as appropriate |
| Insert() | Insert a user record to the DB |
| SelectDetails() | Download a current User from the DB and place it in an instance of the class |

|  |  |
| --- | --- |
| CUWNote | Class module for UWNote |
| GetPreviousFromUN() | Get details from the Underwriting Notes Screen |
| GetNewFromUN() | updates the record |
| PutToUN() | put details to the UW Notes screen |
| InsertOrUpdate() | save the instance to the DB |

# Workbooks - Detailed

## Opportunity

### Worksheets

|  |  |
| --- | --- |
| Sheet1 - Reference data (RD) |  |

|  |  |
| --- | --- |
| Sheet2 - Hidden data (HD) |  |

|  |  |
| --- | --- |
| Sheet7 - Opportunity List (OL) |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_BeforeDoubleClick() | manages user selection of Opportunity |

|  |  |
| --- | --- |
| Sheet3 - Initial Checks (IO) | This module contains all the sheet code for the Opportunity Initial Checks sheet |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Deactivate() | Checks whether Opportunity should be saved, calls routines to save |
| Worksheet\_Change() | Manages cell height for text cells |
| ComboBox1\_MouseDown() | Controls use of combobox |
| ComboBox1\_DropButtonClick() | Controls use of combobox |

|  |  |
| --- | --- |
| Sheet4 - UWNotes (UN) | This module contains details needed for sheets UW Notes Excel Screen |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_BeforeDoubleClick() | Manages user selection of UWNote |
| Worksheet\_Change() | Manages cell height for text cells |

|  |  |
| --- | --- |
| Sheet9 - Quote List (QL) | This contains all the worksheet code for the Quote Lite sheet |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_BeforeDoubleClick() | Manages user selection of a Quote |

|  |  |
| --- | --- |
| Sheet10 – GIP Quote Lite | This contains all the worksheet code for the GIP Quote Lite tab in the QuoteLite workbook |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Deactivate() | Checks whether Quote should be saved, calls routines to save |
| Worksheet\_Change() | Manages cell height for text cells |

|  |  |
| --- | --- |
| Sheet6 – Issued and beyond (OC) | This contains all the worksheet code for the Opportunity Chasing sheet |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Change() | Manages cell height for text cells |

|  |  |
| --- | --- |
| Sheet11 – Notes(NT) |  |
| Worksheet\_Deactivate() | Calls sub that saves a note |

|  |  |
| --- | --- |
| Sheet5 – Opportunity Audit (OA) | This contains all the worksheet code for the Opportunity Audit sheet |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |

|  |  |
| --- | --- |
| ThisWorkbook | This contains all the worksheet code for the workbook |
| Workbook\_Open() | Calls routines required upon opening this workbook |
| Workbook\_BeforeClose() | Manages what routines are called on workbook close action |

### Forms

1. BrokerView\_Form

### Modules

|  |  |
| --- | --- |
| M11\_OpportunityGeneral | This module contains all the General code which is specific for the Opportunity WB |
| SetupSpreadsheet() | Manages application control, users and protection on initial workbook open |
| ProtectSheets() | Protects editing of worksheets |
| UnprotectSheets() | enables editing of worksheets |

|  |  |
| --- | --- |
| M12\_OpportunityInitialChecks | This module contains all the code for the Opportunity Initial Checks sheet |
| IODisplayButtons() | This module controls button display on this sheet |
| IOPopulateScreen() | This module populates the Opportunity Initial Checks screen with available values |
| IONewOpportunity() | Create a new opportunity if validated |
| IOSave() | Save opportunity if validated |
| IOICComplete() | Save opportunity and mark initial checks as completed if validated |
| IOICReviewed() | Save opportunity and mark initial checks as reviewed if validated |
| IOCancelOpportunity() | Cancel opportunity and set status to cancelled |
| IOAllocateUW() | Assign underwriter to Opportunity |
| IODeclineOpp() | Decline Opportunity |
| IODeclineAuthOpp() | Authorise decline of Opportunity |
| IOUndoDecline() | Undo decline of Opportunity |
| IOSendSpecToGEB() | Mark Opportunity as sent to GEB |
| IORequestedGEBInfo() | Mark that we have requested pooling info from GEB |
| IOSummaryGEBInfoSent() | Mark that we have requested pooling info from GEB |
| PopulateBrokerComboBox() | Populate broker combo box with list of broker names |
| IOShowBrokerInfo() | Display information for selected broker |

|  |  |
| --- | --- |
| M13\_OpportunityAudit | This module contains details needed for Audit Table Excel Screen |
| OADisplayButtons() | This module controls button display on this sheet |
| OAUpdateAuditTable() | Manages populating of audit table |
| OAShowAllOppTransactions() | allows you to see transactions for Opportunity and all quotes |

|  |  |
| --- | --- |
| M14\_OpportunityList | This module contains details needed for Audit Table Excel Screen |
| OLDisplayButtons() | This module controls button display on this sheet |
| OLUpdateOppList() | update the list of Opportunities shown |
| OLUnselectRecord() | Unselect any Opportunity which has previously been selected |
| OLAddQuery() | update the list of Opportunities shown |

|  |  |
| --- | --- |
| M15\_UWNotes | This module contains details needed for UW Notes Excel Screen |
| UNClearPreviousEntry() | Clears data from previous UW Note entry |
| UNSave() | Save UW Note if validated |
| UNSelectAndCopy() | Copies new entry to previous entry holding cells |
| UNClearTable() | Removes data from UWNote\_Table |
| UNPopulateTable() | Populates UWNote\_Table from SQL |

|  |  |
| --- | --- |
| M16\_OpportunityChasing | This module contains All the stuff for the Opportunity Chasing screen |
| OCDisplayButtons() | This module controls button display on this sheet |
| OCPutOpportunity() | puts existing Opportunity details to OC screen |
| OCUpdateOpportunity() | Manages updating of Opportunity details and displaying on screen |
| OCQuoteProductionComplete() | update Opportunity status to Quote Production Complete |
| OCQuotePackSent() | Update Opportunity status to Quote Pack Sent |
| OCEnteredOnPhoenix() | Update Opportunity status to Entered on Phoenix |
| OCChased() | Sets chaser status to Chased |
| OCWon() | Sets chaser status to Won |
| OCRetained() | Sets chaser status to Retained |
| OCWonOrRetained | Processes a win or retained (updates Opportunity and related quotes) |
| OCLost() | Sets chaser status to Lost |
| OCWithdrawn() | Sets chaser status to Withdrawn |
| OCNotTakenUp() | Sets chaser status to Not Taken Up |
| OCCreateAndUpdate() | creates new chaser entry on Chaser\_Table (and Audit\_Table) based on chaser status/transaction |
| OCRevisedQteUpdate() | updates IssuedStatus to 'Revised Quote' |
| OCBestRateUpdate() | updates IssuedStatus to 'Best Rates' |
| OCNearWinUpdate() | updates IssuedStatus to 'Await Confirmation of Win' |

|  |  |
| --- | --- |
| M17\_QuoteList | This module contains details needed for Quote Table Excel Screen |
| QLDisplayButtons() | This module controls button display on this sheet |
| QLUpdateQuoteList() | update the list of Quotes shown |
| QLUnselectRecord() | Unselect any Quote which has previously been selected |
| QLCreateInitialQuote() | Enables creation of a new initial quote |
| QLCopyQuote() | Creates a revised quote based on initial quote number |

|  |  |
| --- | --- |
| M18\_QuoteLite | This module contains details needed for GIP Quote Entry Screen |
| GQDisplayButtons() | This module controls button display on this sheet |
| GQPutQuote() | Puts existing quote details to GQ screen |
| GQResetScreen() | Clears existing data and resets Quote Lite screen to default values |
| GQNewQuote() | Enables creation of a new initial quote |
| GQSave() | Saves Quote details |
| GQPutOnHold() | Updates Quote status to On Hold |
| GQPolicyDetailsEntered() | Updates Quote status to Policy Details Entered |
| GQUWComplete() | Updates Quote Status to UW Complete |
| GQUWUndo() | Updates Quote status to Underwriting Undone |
| GQPeerReviewed() | Updates Quote status to Peer Reviewed |
| GQAuthorised() | Updates Quote status to Authorised |
| GQAllocateUW() | Assign underwriter to Opportunity |
| GQDeclined() | Updates Quote status to Declined |
| GQDeclineAuthorised() | Updates Quote status to Decline Authorised |
| GQDeclineUndo() | Updates Quote status to Decline Undo |
| GQPreAuthGEB() | Updates Quote status to Quote Sent to GEB |
| GQFinalQuoteGEB() | Updates Quote status to Authorisation Received from GEB |
| GQCopyAuditTable() | Copies certain columns from Audit table to GIP Quote screen |
| GQExportQuoteToWordDoc() | Copies Quote Lite screen to a new word document |

|  |  |
| --- | --- |
| M19\_Queries | This module contains details needed for Notes functionality - this creates 'little opportunities' which can't have quotes against them |
| LoadQueriesTab() | Creates a new Opportunity note |
| NTSave() | Saves note |
| NTComplete() | Completes and closes an Opportunity note |

## Utility

### Worksheets

|  |  |
| --- | --- |
| Sheet1 - Reference data (RD) |  |

|  |  |
| --- | --- |
| Sheet6 - Hidden data (HD) |  |

|  |  |
| --- | --- |
| Sheet2 – Rate Update (RU) |  |

|  |  |
| --- | --- |
| Sheet3 – Other Rates |  |

|  |  |
| --- | --- |
| Sheet4 –Broker (BU) | This contains all the worksheet code for Broker sheet |
| ComboBox1\_MouseDown() | Controls use of combobox |

|  |  |
| --- | --- |
| Sheet5 - Users (UU) |  |

|  |  |
| --- | --- |
| Sheet7 - Application Control (AU) |  |

|  |  |
| --- | --- |
| Sheet10 – PLOpportunity |  |

|  |  |
| --- | --- |
| Sheet11 – PLQuote |  |

|  |  |
| --- | --- |
| Sheet9 – 2017 Open Quotes |  |

|  |  |
| --- | --- |
| ThisWorkbook | This contains all the worksheet code for the workbook |
| Workbook\_Open() | Controls what happens upon opening the workbook |
| Workbook\_BeforeClose() | Controls what happens upon closing the workbook |

### Forms

1. IFAView

### Modules

|  |  |
| --- | --- |
| M0\_UtilityGeneral | This module contains all the General code which is specific for the Utility WB |
| GenSetupSpreadsheet() | initial setup for workbook, populate Hidden Data fields etc |

|  |  |
| --- | --- |
| M1\_ApplicationControl | This module contains all the code for the Application Control screen |
| TestAppControl() | Manages application control |

|  |  |
| --- | --- |
| M2\_BrokerUpdate | This module contains all the code for the Broker Update screen |
| BUUploadBrokers() | upload records to the DB, insert or update as appropriate |
| BUDownloadBrokers() | download records from the DB |
| BUClearTable() | Removes data from BrokerView\_Table |
| BUPutBrokers() | Download all the Brokers from the DB and place them on the BrokerView table. |
| PopulateBrokerComboBox() | Populate broker combo box with list of broker names |
| GetBrokerInfo() | Get broker details |

|  |  |
| --- | --- |
| M3\_RateUpdate | This module contains all the code for the Rate Update screen |
| RUClearTable() | Removes data from the RateView Table |
| DownloadRates() | Download rates from DB and display them in RateView Table |
| ValidateRatesUpload() | validates rates for upload to DB |
| DeleteCurrentRates() | remove current rates from DB |
| UploadRates() | upload rates to DB |

|  |  |
| --- | --- |
| M4\_User | This module contains all the code for the Users Update screen |
| UploadUsers() | upload records to the DB, insert or update as appropriate |
| DownloadUsers() | download records from the DB |
| UUClearTable() | Remove all data from the UserView table |
| UUPutUsers() | Download all the Users from the DB and place them on the UserView table |

|  |  |
| --- | --- |
| M5\_PLOpportunityUpload | Module for batch uploading Opportunities from the postlist in the Utility Module |
| PLUploadOpportunities() | upload records to the DB, insert or update as appropriate |
| PLOppCopyColumns() | copies required columns over from postlist |
| PLOppConvertData() | Sets Opportunity Status field, converts usernames/initials and broker name to ID's for DB etc |
| PLOppGetOpportunityStatus | figure out status of opportunity from postlist and enter it in Opportunity View table |
| PLOppCleanseData() | remove unwanted characters from postlist data |
| PLOppClearFormatting() | removes cell formatting from postlist |
| HasInformationToQuote | check postlist for whether initial review is completed |
| OtherActiveOppInfo() | Determine if other active Opportunities, get Opp info |
| SetChaserID | set AllocatedChaserID based on Opportunity status |
| PLOppClearTable() | Removes data rows from table |

|  |  |
| --- | --- |
| M6\_PLQuoteUpload | Module for batch uploading Quotes from the postlist in the Utility Module |
| PLUploadQuotes() | upload records to the DB, insert or update as appropriate |
| PLQuoteCopyColumns() | copies required columns over from postlist |
| PLQuoteConvertData() | Sets Quote Status field, converts usernames/initials and to ID's for DB etc |
| PLQuoteGetQuoteStatus | figure out status of quote from postlist and enter it in Quote View table |
| CanOrCannot | sets cell text based on boolean value |
| PLQuoteCleanseData() | remove unwanted characters from postlist data |
| PLQuoteRemoveEmptyEntries() | removes rows that don't contain Quote data |
| PLQuoteClearFormatting() | removes cell formatting from postlist |
| ExtractQuoteVersion | gets quote version number from quote reference |
| SetQuoteType | sets quote type based on quote version column |
| ExtractOriginalQuoteNumber | gets original quote number from quote reference |
| SetDeadlineDate | if empty sets deadline date to value from corresponding Opportunity |
| SetDeadlineType | if empty sets deadline type to value from corresponding Opportunity |
| PLQuotecleartable() | Removes data rows from table |

## Reporting

### Worksheets

|  |  |
| --- | --- |
| Sheet4 - Reference Data (RD) |  |

|  |  |
| --- | --- |
| Sheet5 - Hidden Data (HD) |  |

|  |  |
| --- | --- |
| Sheet1 - Filtered(FR) |  |

|  |  |
| --- | --- |
| Sheet2 – Timed(TR) |  |

|  |  |
| --- | --- |
| Sheet3 - ChaseOpportunities |  |

|  |  |
| --- | --- |
| Sheets6 – ChaseQuotes |  |

|  |  |
| --- | --- |
| ThisWorkbook | This contains all the worksheet code for the workbook |
| Workbook\_Open() | Calls routines required upon opening this workbook |
| Workbook\_BeforeClose() | Manages what routines are called on workbook close action |

### Modules

|  |  |
| --- | --- |
| M1\_General |  |
| SetUpSpreadsheet() | Manages application control, users and protection on initial workbook open |
| ProtectSheets() | Protects editing of worksheets |
| UnProtectSheets() | enables editing of worksheets |

|  |  |
| --- | --- |
| M2\_Filtered |  |
| FRGenerateReports() | For interface, manages calling of other subroutines. |
| FRGetOpportunityData() | Download Opportunity report data and place into Excel table |
| FRGetQuoteData() | Download Quote report data and place into Excel table |

|  |  |
| --- | --- |
| M3\_Timed |  |
| FRGetTimedData() | Download Timed opportunity report data and place into Excel table |

|  |  |
| --- | --- |
| M4\_Chaser | This module manages production of the reports on the Chaser sheet of the workbook. |
| OCRGetOppReport() | Download Opportunity report data and place into Excel table |
| OCRGetQuoteReport() | Download Quote report data and place into Excel table |

## Rating

### Worksheets

|  |  |
| --- | --- |
| Sheet1 – Reference Data(RD) |  |

|  |  |
| --- | --- |
| Sheet10 – Member Data |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Change() | Checks for changes on the member table |

|  |  |
| --- | --- |
| Sheet11 – Data Validation |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Change() | Manages cell height for text cells |

|  |  |
| --- | --- |
| Sheet12 – Hidden Data(HD) |  |

|  |  |
| --- | --- |
| Sheet13 – Opportunity List |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_BeforeDoubleClick() | When a user double clicks on an opportunity it is selected. |

|  |  |
| --- | --- |
| Sheet14 – Letter Info |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |

|  |  |
| --- | --- |
| Sheet15 – Front Sheet and Audit |  |

|  |  |
| --- | --- |
| Sheet16 – Premium Calculation |  |

|  |  |
| --- | --- |
| Sheet17 – Policy Conditions |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Change() | Resets PCValidatedDate cell on sheet |

|  |  |
| --- | --- |
| Sheet18 – Quote List |  |
| Worksheet\_Activate() | calls a sub responsible for populating the Quote List Table |
| Worksheet\_BeforeDoubleClick() | When a user double clicks on an opportunity it is selected. |

|  |  |
| --- | --- |
| Sheet2 – Initial Checks |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Change() | Manages cell height for text cells |

|  |  |
| --- | --- |
| Sheet3 – Letter Info2 |  |

|  |  |
| --- | --- |
| Sheet4 – Working - other stuff |  |
| Worksheet\_Change() |  |

|  |  |
| --- | --- |
| Sheet5 – Opportunity Audit |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |

|  |  |
| --- | --- |
| Sheet6 – Calc Review |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_Change() | Manages cell height for text cells |

|  |  |
| --- | --- |
| Sheet7 – Claims Calculation |  |
| Worksheet\_Change() | resets fields on sheet if table is updated |

|  |  |
| --- | --- |
| Sheet8 – UW Notes |  |
| Worksheet\_Activate() | Calls routines required upon switching to this worksheet |
| Worksheet\_BeforeDoubleClick() | When a user double clicks on an UWnote Row it is selected. |
| Worksheet\_Change() | Manages cell height for text cells |

|  |  |
| --- | --- |
| Sheet9 – Category Conditions |  |
| Worksheet\_Change() | resets fields on sheet if table is updated |

|  |  |
| --- | --- |
| ThisWorkbook |  |
| Workbook\_Open() | Run when the Workbook is opened |

### Forms

1. OccupationCodeSelection

### **Modules**

|  |  |
| --- | --- |
| M30\_GeneralQuotes | This module contains lots of general functions used through out the Quote Module Only |
| ProtectSheetV2() | sets protection on the workbook |
| ProtectSheets () | This module is called each time the workbook is opened and when the data is reset. |
| UnProtectSheets() | unprotects sheets |
| SetUpSpreadsheet() | Manages application control, users and protection on initial workbook open |

|  |  |
| --- | --- |
| M31\_InitialChecks | This module contains all the code for the Initial Checks sheet |
| ICDisplayButtons() | button management on worksheet |
| ICPopulateScreen() | Populate the screen with Opportunity and Quote details |
| ICClearScreen() | resets the screen, clearing out existing data |
| SendQuoteToGEB() | This module updates the Interaction with GEB section. |
| ReferenceDataClick() | sets reference data tab to visible |
| ICSave() | Saves the quote information entered |
| ICShowBrokerInfo() | Display information for selected broker |
| ICAllocateUW() | Assign underwriter to Opportunity |

|  |  |
| --- | --- |
| M32\_Policy | This module contains all the code for the Policy data calculations and validation used on the policy tab |
| AddPolicyRow() | adds a blank row to the policy table |
| ValidatePolicyRows() | validates data on policy table |
| UpdateExistingFCL() | If the scheme is brand new the Existing FCL should be updated |

|  |  |
| --- | --- |
| M33\_Categories | This module contains all the code for the Category data calculations and validation used on the Cat tab |
| AddCategoryRow() | adds a blank row to the category table |
| PopulateCategoryRows() | manages populating of category table |
| ValidateCategoryInfo() | validation for data on the category table |
| PopulateCategoryInfo() | this subroutine auto populates rows on the Category table |
| GetTerminationAge | function to lookup termination age |
| GetPolicyInfo() | function to lookup early retirement and redundancy policy info |

|  |  |
| --- | --- |
| M34\_Member | This module contains all the code for the Member data calculations functionality used on the Member data tab |
| AddMemberRow() | adds a number of blank rows to the member table |
| PopulateAllMemberRows() | populates all rows of the member table |
| PopulateErroredMemberRows() | populates errored rows |
| PopulateMemberRows() | This procedure calculates all fields on Member table in one pass. |
| ValidateMemberRow() | validate a row on the member table |
| PopulateMemberRow() | populates a row of the member table |
| PostPopulationValidation() | validate a populated row of the member table |
| GetRowErrorMsg | returns error value from a row |
| ValidatePolicyCategory() | validates policy and category data for a row |
| CollectCategoryInfo() | get data from category and policy tables for member table |
| GetAgeNextBirthday | calculate age next birthday from DOB and PCEffectiveDate field |
| GetReasonForExclusion | logic for deciding the reason for exclusion |
| GetDISPType | This gets the DISP type from the category table |
| GetGLARate | This function collects the GLA rate for a member row |
| GetEscalation | This function calculates the escalation |
| GetGeneraliCapFac | looks up the Cap Factor for a member row |
| GetGeneraliTCSA | looks up the Generali TCSA for a member row |
| GetDISPRate | looks up the DISP rate for a member row |
| GetDISPPrem | calculates DISP premium from rate and DISP |
| GetPostCapSalary | Get the salary to use in the calculations |
| GetPostMaxBenefitLSDB | Limit the amount of LSDB depending on the Maximum benefit limits |
| GetExistingTCSA | Calculate Existing TCSA |
| GetHardCapFactor | get the Hard Cap Factor if required |
| GetHardTCSA | Calculate Existing hard TCSA |
| GetHardCapFactorValue | get the Hard Cap Factor value |
| DoICalculateHardCap | determine whether hard cap needs calculating |
| GetMUWFlag | determine MUW flag |
| GetDeferredRetiree | determine if member is over retirement age |
| PopulateOccupations() | This module calculates and populates the Occupation codes |
| PopulateLocations() | This function populates the location code |
| PopulateLocation() | populate a location for a row |
| DeleteDifficultOccupations() | This module deletes the 'difficult occupations table' on the reference tab |
| HideAlwaysHiddenColumns() | This procedure hides all of the columns in the member data table which are not required for all products |
| ResetReports() | resets the report section of the worksheet |
| DisplayLTSReport() | displays the LTS report |
| DisplayEarlyRetReport() | displays the early retiree report, basically the member data with a few columns missing and filtered. |
| DisplayMUWReport() | displays the MUW report, basically the member data with a few columns missing and advanced filtered. |
| DisplayGLAOnly() | Display GLA without DISP |
| DisplayGLADISP() | Display GLA and DISP |
| DisplayMemberButtons() | button management for worksheet |

|  |  |
| --- | --- |
| M35\_DataValidation | This module contains all the code for the DataValidation calculations functionality used on the data Validation data tab |
| DVDisplayButtons() | button management for worksheet |
| DataValidationComplete() | validates and completes a policy |
| UpdateLocationTCSAPivotTable() | This function refreshes a pivot table of member data, looks at the result  and returns the number of rows (postcodes) which have a TCSA greater than a certain value. |
| UpdateExcludedLives() | this module counts the number of excluded lives for DISP and GLA. |
| UpdateOverMaxBenDOBs() | This sub brings together all the DOBs above maximum benefit to be displayed on the calc review sheet & Data Validation. |
| PopulateReferrals() | Populates the referrals table |

|  |  |
| --- | --- |
| M36\_ClaimCalc | This module contains all the code used in the claims calculation modules |
| ClearClaimsCalcTable() | clears entries on ClaimsCalculation table |
| AddClaimsInputRow() | adds a blank row to the Claims Input table |
| ValidateClaimsInputTable() | validates entries on the ClaimsInput table |
| PopulateClaimsInputTable() | populates the ClaimsInput table |
| PopulateClaimsCalculationTable() | Populates the ClaimsCalculation\_Table table |
| IsClaimsInputRowBlank | this tests to see if a row on the claims input table is blank or not |
| UnHideCredibilityRows() | displays credibility rows on worksheet |
| HideCredibilityRows() | Hides credibility rows on worksheet |
| GetNextRow() | this function returns the next row on the Claims Input table in an array. |
| UpdateClaimsUpdatedDate() | update date |

|  |  |
| --- | --- |
| M37\_PremCalc | This module contains all the code for the Premium Calculations used on the Premium Calculation tab |
| ClearPolicySummaryTable() | delete rows from PolicySummary table |
| ClearCategorySummaryTable() | delete rows from CategorySummary table |
| PopulateSummaries() | manages populating of Policy and Category Summary tables |
| ValidatePopulateSummaries() | validate summaries |
| PopulatePolicySummaryTable() | This routine populates the Policy Summary table,it also calculates the Policy enhancements Percentage. |
| PopulateCategorySummaryTable() | populates the CategorySummary table |
| GetGLAClosedLoad | gets GLA closed load from base rate |
| GetDISPClosedLoad | gets DISP closed load from base rate and escalation |

|  |  |
| --- | --- |
| M38\_CalcReview | This module contains all the code for the Calc Review tab, includes button presses on the calc review tab and on member tab doen at this time of process |
| CRDisplayButtons() | button management for worksheet |
| PostAssessedMUWRqd() | sets the MDPostAssedRqd status of an entry |
| UpdatePostAssessedMUW() | This routine updates the member table during the Calc Review process |
| CalcReviewUnUnderwritten() | Undoes the underwriting |
| CalcReviewUnderwritten() | updates a quote to underwritten |
| CalcReviewPeerReview() | updates a quote to peer reviewed |
| CalcReviewAuthorised() | updates a quote to authorised |
| CalcReviewQuoteCopied() | creates a copied quote |
| UWDecline() | updates a quote to declined |
| UWDeclineAuth() | updates a quote to decline authorised |
| UWDeclineUndo() | updates a quote to decline undone |
| M39\_LetterInfo | This module contains all the code used by the LetterInfo tab |
| GenerateCostingSheets() | generates the costing sheets and populates with data |
| GenerateCategoryLetterInfo() | generates category letter information |
| ClearLetterInfo() | removes letter data and CategoryLetterInfo table |
| GetBenefitLimit | logic to calculate benefit limit |
| GetEscalationDesc | This function calculates the escalation |
| GetSalaryDef | gets salary definition |
| GetTerminationAge | gets termination age |
| GetIllness | gets illness info |
| GetTempAbsAnyOther | gets reason for temp abs |
| GetTempAbsIncrease | gets reason for temp abs increase |
| GetRetireCover | get retirement cover |
| GetRedundancy | get redundancy cover |

|  |  |
| --- | --- |
| M41\_OpportunityList\_copy  (Copy of module on Opportunity workbook) | This module contains details needed for Audit Table Excel Screen |
| OLDisplayButtons() | This module controls button display on this sheet |
| OLUpdateOppList() | update the list of Opportunities shown |
| OLUnSelectRecord() | Unselect any Opportunity which has previously been selected |

|  |  |
| --- | --- |
| M42\_QuoteList\_copy  (Copy of module on Opportunity workbook) | This module contains details needed for Quote Table Excel Screen |
| QLDisplayButtons() | This module controls button display on this sheet |
| QLUpdateQuoteList() | update the list of quotes shown. |
| QLUnselectRecord() | Unselect any Quote which has previously been selected |
| QLCreateInitialQuote() | Enables creation of a new initial quote |
| QLCreateCopyQuote() | enable creation of a copied quote |

|  |  |
| --- | --- |
| M43\_AuditList\_copy | This module contains details needed for Audit Table Excel Screen |
| OADisplayButtons() | This module controls button display on this sheet |
| OAUpdateAuditTable() | Manages populating of audit table |
| OAShowAllOppTransactions() | allows you to see transactions for quote and Opportunity |

|  |  |
| --- | --- |
| M44\_UWnotes | This module contains details needed for UW Notes Excel Screen |
| UNClearPreviousEntry() | Clears data from previous UW Note entry |
| UNSave() | Update the UW Notes table |
| UNSelectAndCopy() | Copies new entry to previous entry holding cells |
| UNClearTable() | Removes data from UWNote\_Table |
| UNPopulateTable() | populates UWNote\_Table from SQL |

|  |  |
| --- | --- |
| M96\_Workflow | This code is not called |
| CreateButtons() |  |

|  |  |
| --- | --- |
| M98\_DevelopmentOnly | This module contains lots of general functions used only in development, it is not need when system is live |
| ResetAll() | This module should only ever be used once. It removes all the calculated values from the sheets and sets a few initial values up. This should only need to be done before saving the spreadsheet for the first time before use. |
| LiveRefData() | populates Reference Data tab |
| TestRefData() | populates Reference Data tab |

|  |  |
| --- | --- |
| M99\_ExampleCode | example code not used |
| SelectingPartOfTable() |  |
| Macro1() |  |
| Set\_AsField() | This macro sets the selcted field so that it can be edited |
| Set\_AsBackground() | Set the selected field as a grey background |
| Set\_AsBackground2() | Set the selected field as a grey background |
| ProtectSheet() |  |

# Transactions

## Opportunity Transaction/Status

|  |  |
| --- | --- |
| Transaction | Next Status |
| Initial Checks Started | Awaiting Initial Checks |
| Saved | Awaiting Initial Checks |
| Saved Later | Awaiting Initial Checks |
| Underwriter Allocated | Awaiting Initial Checks |
| Initial Checks Complete | Awaiting Initial Checks Review |
| Initial Checks Reviewed | Awaiting Quote(s) Production |
| More Quote(s) Requested | Awaiting Quote(s) Production |
| UWNote Update | Awaiting Quote(s) Production |
| Quote Production Complete | Awaiting Quote Pack Sending |
| Quotes Sent | Issued |
| Update Chase | Issued |
| Quotes Chased | Issued |
| Won | Awaiting Entry on Phoenix |
| Retained | Awaiting Entry on Phoenix |
| Lost | Closed |
| Withdrawn | Closed |
| Not Taken Up | Closed |
| Entered On Phoenix | Closed - Won |
| Declined | Awaiting Decline Authorisation |
| Decline Authorised | Declined |
| Decline Undone | Awaiting Initial Checks |
| Spec Sent to GEB | SAME AS BEFORE |
| Requested GEB Info | SAME AS BEFORE |
| Summary GEB Info Sent | SAME AS BEFORE |
| Cancelled | Cancelled |
| Create Note | SAME AS BEFORE |
| Update Issued Status | SAME AS BEFORE |

|  |  |
| --- | --- |
| Opportunity Statuses | |
| Awaiting Initial Checks | |
| Awaiting Initial Checks Review | |
| Awaiting Quote(s) Production | |
| Awaiting Quote Pack Sending | |
| Issued | |
| Awaiting Entry on Phoenix | Awaiting Decline Authorisation |
| Closed – Won | Declined |
| Closed | |
| Cancelled | |

## Quote Transaction/Status

|  |  |
| --- | --- |
| Transaction | Next Status |
| Quote Created | Awaiting Policy Entry Detail |
| Policy Details Entered | Awaiting Underwriting |
| UW Complete | Awaiting Peer review - Authorising |
| Peer Reviewed | Quote Complete |
| Authorised | Quote Complete |
| Underwriting Undone | Awaiting Underwriting |
| Won | Awaiting Entry on Phoenix |
| Retained | Awaiting Entry on Phoenix |
| Lost | Closed |
| Withdrawn | Closed |
| Not Taken Up | Closed |
| Declined | Awaiting Decline Authorisation |
| Decline Authorised | Declined |
| Decline Undone | Awaiting Policy Entry Detail |
| Quote Chased | Issued |
| Quote Sent | Issued |
| Entered On Phoenix | Closed – Won |
| Cancelled | Cancelled |
| Quote Sent to GEB | SAME AS BEFORE |
| Authorisation Received from GEB | SAME AS BEFORE |
| On Hold | On Hold |
| Underwriter Allocated | SAME AS BEFORE |

|  |  |
| --- | --- |
| Quote Statuses | |
| Awaiting Policy Entry Detail | |
| Awaiting Underwriting | |
| Awaiting Peer review - Authorising | |
| Quote Complete | |
| Awaiting Entry on Phoenix | Awaiting Decline Authorisation |
| Closed – Won | Declined |
| Closed | |
| Cancelled | |
| On Hold | |

# DB Diagram

**Guide**

|  |  |
| --- | --- |
| Table | Purpose |
| Opportunity | (potential) piece of business |
| Quote | Each Opportunity can have any number of Quotes |
| Chaser | An Opportunity and/or Quote that has been chased up by the business |
| Broker | list of data used in the creation of Opportunities/Quotes |
| Rate | list of data used in the creation of Opportunities/Quotes |
| UWNote | Opportunities can have notes attached to them containing further information |
| Users | contains all information regarding users of the system |
| TransactionAudit | Each step in the process of Opportunity management is logged in this table for audit purposes |

# Release Instructions

* Users should access the latest version from the network, then save a copy to their local machine to use on a regular basis.
* Admin can manage versioning using the Reference Data tabs on the workbooks and the Application Control table that resides in Production.
* Prior to a new release a new entry should be added to the Application Control table containing the new VersionIdentifier, which is also saved in the workbook.
* Old entries should be removed if no longer valid.
* The workbook AddIn references should be updated to the new version of the AddIn, if applicable.
* Users should be advised to obtain a copy of the new workbook from the network.

# Add-In Installation Instructions

For Users on the new VDI (Virtual desktop built in 2016) the Add In needs to be installed the first time before using.

The instructions for carrying out this task are as follows:

* Open the (Opportunity) Workbook
* File > Options > Add-Ins
* Choose Drop down: “Manage Excel Add-ins”, click – Go
  + Browse to the Add-In file on the network and select
* It will give you a choice to Install the Add-In on the local machine – choose No.
* Close the workbook & all of Excel
* Reopen Excel

Remember to add Users to the User table…

# Postlist Upload Instructions (Utility Workbook – Opportunity/Quote)

Most of the functionality is wrapped in the two buttons, ‘Import Opportunities/Quotes’, which will get a list of Opportunities/Quotes from the Postlist data tab (‘2017 Open Quotes’), and ‘Upload Opportunities/Quotes’, which will insert them into the DB. However, there are some manual stages;

1. Copy data over from Postlist workbook into ‘2017 Open Quotes’ tab on Utility workbook.
2. Ensure that the ‘Postlist\_Table’ is maintained on that tab
3. Click ‘Import Opportunities’ on ‘PLOpportunity’ tab
4. Review data in columns highlighted in red – ensure these contain IDs that exist in the DB.
5. On the PLOpportunityView\_Table - BrokerID column ‘FIX ME’ indicates that the Broker doesn’t exist in the DB. Either replace this with a valid BrokerID or use the Broker tab to insert new brokers into the DB, and then run PLOppConvertData() again to populate this column.
6. Once all data is ready you should be able to press ‘Upload Opportunities’.
7. Repeat steps 3,4,6 for Quotes, it’s important to upload all Opportunities first.

Upload will fall over if:

* Columns that refer to DB Foreign Keys contain data that cannot be found in the DB, i.e. missing UserIDs, or uploading a Quote with an OpportunityID that doesn’t yet exist.
* Columns that contain incorrect data, e.g. date columns that contain text and haven’t been cleansed during the import process
* Cells violate data type or size restrictions, i.e. ProductType is restricted to VARCHAR(20)
* If an Opportunity/Quote you’re trying to upload already exists in the DB.

# **Generali Excel VBA Standards**

## Naming Standards

|  |  |  |
| --- | --- | --- |
| Type | Format | Example |
| Modules | M#\_<MeaningfulName> | M08\_Quote (Unique numbers across entire system) |
| Classes | C<MeaningfulName> | CQuote |
| Excel Tables | <MeaningfulName>\_Table | Opportunity\_Table |
| Labels | <2 char reference to tab><MeaningfulName> | RD – reference data, MD – Member data, RU – Rate Upload |
| Variables | Prefix private class variables with ‘p’ | pOpportunityID |
| Buttons | <2 Char reference to tab><MeaningfulName>\_Btn | Sheet3.Shapes.Range(Array("ICStarted\_Btn")) (always with a full reference) |
| Class Instance | <2 char reference to class> | OP (“Dim OP As EBGeneralAddIn.COpportunity”) |

## Language

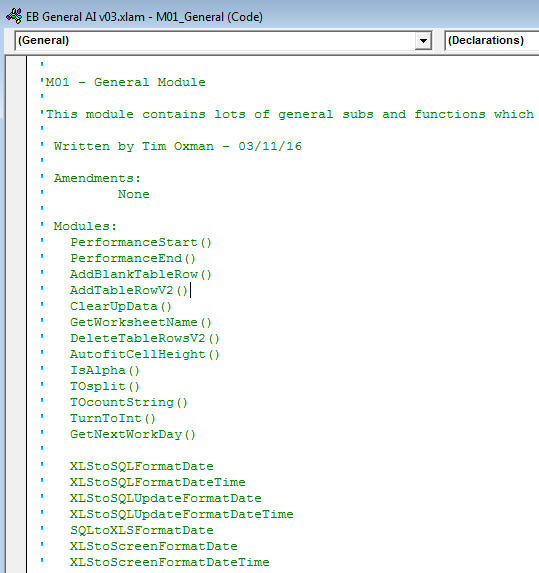
* Use ‘Put’ and ‘Get’ to transfer data to and from Excel tables
* Use ‘Insert’ and ‘Select’ to transfer data to and from SQL tables
* Refer to Visual Basic/code as ‘VBA’, SQL DB as ‘SQL’ and Excel as ‘XLS’
* Database interactions via ADO, use following variables:
  + LinkToServer = ADODB connection
  + LoginAccess = Database connection string
  + SQLSyntax = SQL query string
  + Basket = Database recordset container
* ‘Rw’ and ‘Row’ used as counters in loops through data
* ‘OkToProceed’ (returns true/false) as a validation check

## Worksheets

* Branded according to Generali standards
  + Use of red stripe
  + Logo on first page
  + Official colour scheme

## Code Modules

* Comment block inside Declarations stating Module title, explanation, author, amendments and modules.



* Then declare ‘Option Explicit’ followed by constants and variables (global and Module-scoped)
* Sub/function in CamelCase based on what it does/returns e.g. ‘AddBlankTableRow()’ / ’GetWorksheetName’ followed by one line comment to explain
* Prefix subs/functions with <MeaningfulName> e.g. ‘ChaserClearTransactions()’ not ‘ClearChaserTransactions’
* Functions can return an error message as a string to indicate failure, or an empty string to indicate successful operation
* Functions named ‘Get\*’ when they return something or ‘Has\*/Is\*/Are\*’ if they return a boolean value.
* Variable names in CamelCase in consistent format i.e. ‘ReceivedDate’, ‘ChasedDate’ and matching DB column names
* Declare variable types, including variant where required, at start of sub/function.
* 2 blank lines between each subroutine/function. Single lines within subs/functions to separate into meaningful sections and to aid readability
* Avoid acronyms/jargon as much as possible
* Don’t use GoTo statements, avoid selecting objects where possible
* Repeated code extracted to common modules where feasible or
* EBGeneralAddIn for code used across multiple places/workbooks and Class modules
* Code stored in modules with only calls to modules stored in Excel objects
* Single task in each subroutine where possible
* Use of ‘[ ]’ notation to reference worksheet cells, instead of using “Range….”
* Use “Range…” notation for referencing cells in tables

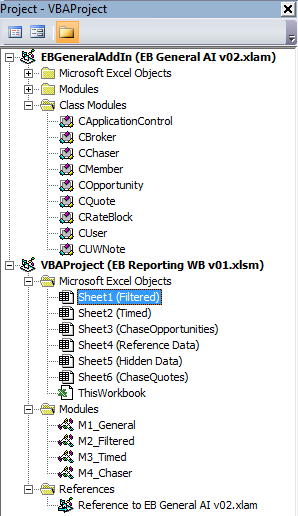
## Standard Modules

* Buttons – button manipulation is managed on a per-module basis as functionality is similar but operation is situation-specific. These subroutines are named ‘<2 char reference to tab>DisplayButtons’. Buttons are generally set to an initial status of inactive, with colours being determined by the Generali colour scheme.
* Button per ‘action’ with state change from active to inactive, don’t give a button complex/multiple functionality.
* Date Manipulation – Functions to convert dates into the required formats across Excel, VBA and SQL are stored in the EBGeneralAddIn and called where required elsewhere in the code. Dates shouldn’t be formatted on-the-fly.

# VBA Cheat Sheet

## Developer Window

Accessed by pressing ‘ALT + F11’, this opens a VBA project window, with a navigation pane on the left-hand side, as below. Listed in the navigation are the objects currently associated with the VBA project, these can be the worksheets that make up the workbook, modules, forms, add-ins and references.



## Recording Macros

It is possible to record a macro and have VBA generate the required code for you, however it’s not advised as tasks can generally be completed much more succinctly by writing the code yourself. However, it’s useful as a first step when trying to write some functionality for the first time.

On the Developer tab (or in the bottom left of the Excel window) there is a ‘Record Macro’ button. Hitting this will open a dialogue box for you to name the macro, and choose where to store it. Once you select ‘Ok’ on this dialogue box, Excel begins recording of the macro. Here, all your interactions with the worksheet will be recorded, until you select the ‘Stop Recording’ button (from either the Developer menu or bottom left-hand corner of the Excel window). You can then view this code by selecting its name from the navigation pane within the VBA editor, or via the ‘Macros’ button on the Developer menu (select the macro and choose ‘Step Into’ or ‘Edit’ to view the code).

## Controls

The Controls section of the Developer tab gives us a number of useful features. We can view our VBA code and we can add controls to our worksheets, such as buttons, that will run a macro assigned to them. Design Mode allows us to edit a control, and the Properties option shows us the properties for the selected control.

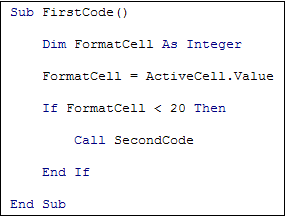
## Dot Notation

Like many other programming languages, VBA uses dot notation for property assigning and method calling. For example “tv.size = ’55 inch’” would add a size property (with a value of ’55 inch’) to our tv object. Methods are called on the object in a similar fashion, e.g. ‘tv.buy’ would call the tv’s buy method. Parameters can be added to methods with a space separating them from the method, followed by ‘:=’ and the parameter name, i.e. ‘tv.buy PaymentType:=Cash Discount:=No’.

## Subroutines and Functions

### Subs

Subroutines, or ‘Subs’ don’t return a value, they simply execute some code. A sub can be made private by adding the word ‘Private’ before ‘Sub’, meaning it won’t be available globally.



This example above executes a second sub (SecondCode) from within the initial FirstCode sub. You can pass parameters inside the brackets of a Sub declaration, e.g.

**Sub FirstCode(Value As Boolean, Name As String, Optional ByVal Age As Integer)**

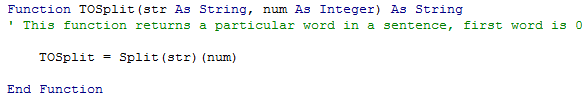
You can then call this Sub with applicable arguments (explicitly or as variables), i.e.

**Call FirstCode(True, ‘Tim’)** - True/1 , False/0

The third parameter in this subroutine is set as an optional argument. These should be passed with the correct ‘passing mechanism’ – ‘ByVal’ where the value should not be modified by the routine, or ‘ByRef’ where you expect it to be, e.g. if it was the resulting value from a calculation.

### Functions

If you want a chunk of code to return some sort of answer then you use a ‘Function’. The result will be stored as the value of the function name, and the data type must match what you established it as when you setup the function (or default to Variant).



You can access Excel functions from within VBA by prepending ‘WorksheetFunction’ e.g 

### Application.WorksheetFunction.function vs Application. function

Because VBA is used by many applications beyond Excel, the Excel worksheet functions are not part of the VBA language itself. However, you can call worksheet functions directly through the Application object or through the Application.WorksheetFunctions class. The difference between using or omitting the WorksheetFunctions reference is how errors are handled. If you do include the WorksheetFunction property, errors will manifest themselves as runtime errors and cause the application to stop running. WorksheetFunction is generally slower as it involves interaction with the workbook, and you should consider backwards compatibility when using WorksheetFunction.

## Variables

Variables are created with the following syntax ‘<Keyword> <VariableName> As <Variable Type>’ e.g. ‘Dim MyNumber As Integer’. VBA is a static language, therefore you must state the variable type at the stage of creation. Failure to set up the variable without assigning a type will result in the variable being set to a data type called a ‘Variant’, which should be avoided as it will cause slow down at run time.

You can then assign that variable a value, ‘MyNumber = 10’. Variables cannot start with a number, nor contain a space or certain characters (!, %, ?, #, $).

Adding ‘Option Explicit’ at the very top of the coding window will instruct VBA to check all of your variables for you, and communicate errors via error messages. To set this on as default go to the Tools menu > Options and tick ‘Require Variable Declaration’ under the Code Settings section of the Editor tab.

This top area is reserved for ‘Declarations’, and anything here, including variables, will be scoped to the entire document. Otherwise variables are local to whatever sub or function you established them in.

As Integer – whole number between -32, 768 and 32,767.

As Long – whole number between -2,147,483,648 and 2,147,483,647

As Single – floating point number up to 4 bytes of data

As Double – floating point number up to 8 bytes of data

The keyword ‘Set’ is used to create new object variables;



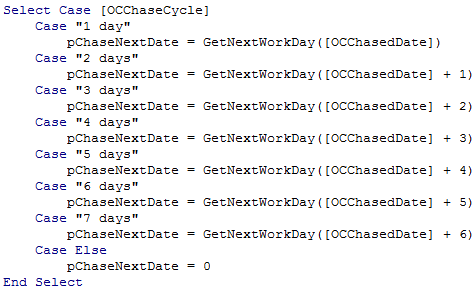
## Conditional Logic

|  |  |
| --- | --- |
| **Multiple Line IF Statement** | **Single Line IF Statement / Inline IF Statement** |
|  | If User\_Choice = “R” Then Null End |
| [IONumberOfQuotes] = IIf(pNumberOfQuotes = 0, "", pNumberOfQuotes) |

If statements can also be nested, and it’s good practice to indent these to keep them readable.

Case statements

Case statements can be used as an alternative.



### Operators

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| **=** | Has a value of |
| **<** | Less than |
| **>** | Greater than |
| **<=** | Less than or equal to |
| **>=** | Greater than or equal to |
| **<>** | Not equal to |
| **Not** | Test if value is NOT something |
| **And** | Test for more than one condition (only TRUE is all evaluate to true) |
| **Or** | Test if the value is either OR something (TRUE if any evaluate to true) |
| **Xor** | Test if one and only one value is true |

## Loops and Iteration

|  |  |
| --- | --- |
| For Loops | While Loops |
|  |  |
| Do While | Do Until |
|  |  |

## Basic Mathematics

This can be done with the standard mathematic operators, +, -, \* and /. You can pass values directly or values stored as variables.

## Modules and code structure

## Comments

These are added with a single quote mark preceding the comment text, and will display in green in the editor window. Buttons to comment/uncomment a block of code are found on the Edit toolbar.

## Excel interaction

Excel ranges are referenced with the ‘Range’ property, where the range is represented as the top left and bottom right cells as a string, separated by a colon. For example, ‘Range(“A1:B7”). The Range property has a number of methods available for it, with ‘Select’ being probably the most common one. Range itself is a property of the Worksheets object, and it is implied to be of the current worksheet unless one is stated, e.g. “Worksheets(“one”).Range(“A1:B7”).Select”. Cells can also be targeted using row and column number references, e.g. “Sheet1.Cells(1,2)” – which would refer to cell B1 in this example.

## Excel syntax within VBA

As alluded to above Excel functions can be used within VBA, where VBA doesn’t already have a version itself. For example, CountIf;



## Debugging VBA

There are numerous ways to debug errors in VBA code;

* Keyboard button F8 will allow you to step through your code line by line
* Clicking to the left of a line of code will add a breakpoint and stop the code execution at that point
* The Watch window shows the current values of selected variables or expressions and the current property settings for selected objects. You can use the Watch window to monitor the status of variables and objects as you step through a procedure.
* The Locals window allows you to view the value of all the variables in a procedure when you are stepping through the procedure. To display the Locals window, choose Locals Window from the View menu. Using the Locals window is easier to display variable values than examining the value from the Immediate window. For simple variable types (e.g., Long and String variables), the value is displayed on one line. For complex types or objects (e.g., a Range variable), its properties are displayed in a collapsible tree-like structure.
* “Debug.print” will output whatever input you give it, e.g. a variable value, to the Immediate Window (accessible with CTRL + G)
* “MsgBox X” can also be used to generate a pop-up message box containing the value of “X”.

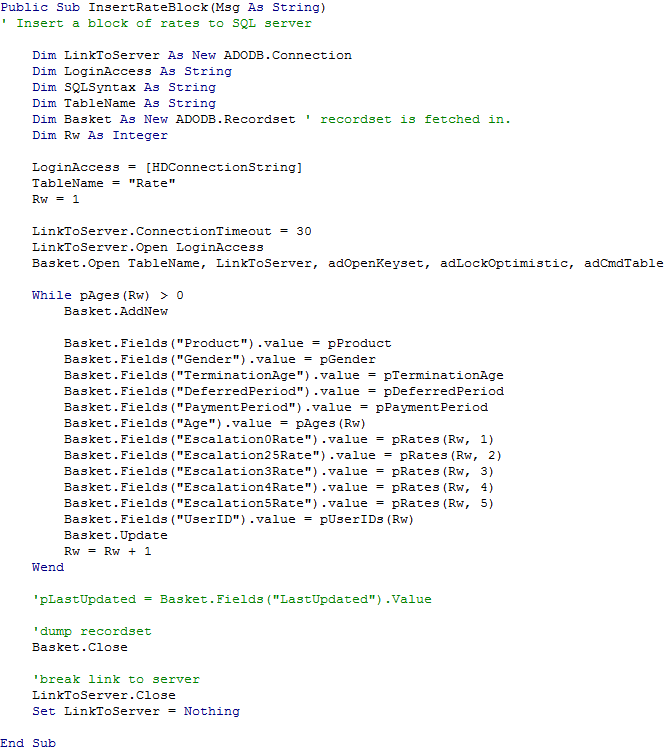
## Useful Excel skills

Naming cells or tables enables you to easily reference them in VBA, without having to state their exact address. For example, “[HDOpportunity]” refers to a particular cell on Sheet2(“Hidden Data”). So long as names are not duplicated VBA will find them in the workbook.

It’s always worth considering whether you need VBA code to achieve your objective, or whether functionality built-in to Excel will do the job, or perhaps both, when validating data input for example. Excel has reasonable data validation, along with common formatting functionality.

## Advanced VBA

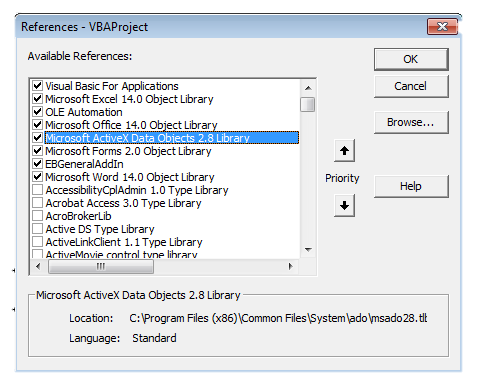
### Interacting with SQL



#### HDConnectionString

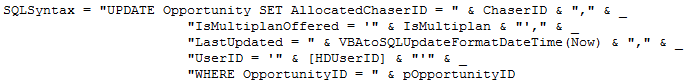
|  |  |
| --- | --- |
|  | Provider=SQLOLEDB; Integrated Security=SSPI; PeBsktist Security Info=True; Data Source=TC2-SQL08\INSTANCE01; Initial Catalog=EBSchemeUW\_DEV; |

ADODB.Connection = connection object to the DB via ADO (ActiveX Data Objects Library). This needs to be added as a reference from within your VBA project. From the Tools menu, select References and then find the necessary reference from the list of available ones.



Login Access = connection string containing the database type (SQLOLEDB), connection security settings (using SSPI, or windows authentication, in this example), the address of the server instance and the name of the database.

SQLSyntax = query string for communicating the changes you intend to make to the data stored in the database. For example;



This must be written in valid VBA string format that will also be able to be read by your SQL software. Key things to consider are to include spaces between SQL operators, and how different data types need to be accommodated, e.g. strings, variables, dates etc. The above example has an integer variable, a string variable, a datetime object (that is being formatted for SQL) and a string referenced directly from the Excel spreadsheet ([HDUserID]).

“UPDATE” is the keyword used for updating a record in a SQL database. “INSERT” is used for creating new records, and “SELECT” for retrieving existing records.

It can be useful to use ‘Debug.print SQLSyntax’ to see the full output string, and this can be copied into your SQL software to compare.

ADODB.Recordset = ADO returns database records in an ADODB.Recordset object.

#### Basic How To

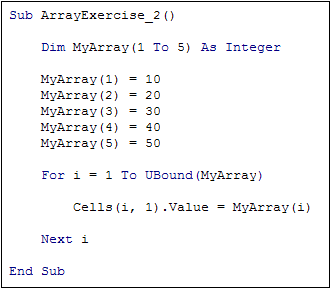
1. Establish connection to server with connection string
2. Open ADODB.Recordset object
3. Do DB transaction with SQL query
4. Close ADO.Recordset object
5. Terminate connection to server

### Arrays

Dim MyArray(4) as Integer – establishes an array of five integers.

MyArray(0) = 10 – sets the first array element to a value of 10

ArrayValue = MyArray(0) – retrieves the first array element and assigns it to a variable called ArrayValue



Dim MyArray(1 to 5) As Integer – establishes an array of 5 integer elements with a starting position of 1 (rather than 0).

For i = 1 to UBound(MyArray) – convention to use i in a loop. UBound(MyArray) sets the upper boundary of the loop to the last number in the array. (LBound will get the lowest value).

Cells(i, 1).Value = MyArray(i) – i updates with each iteration, updating the position of the cell and the value of MyArray.

### Working with Files

Open <FileName> For <Mode> As <FileNumber>

FileName = “path to file” e.g. “C:\Users\Owner\VBA\authors.csv”

Mode:

Append – used for adding data to an already existing file

Output – used to write to a file

Input – used to read a file

Binary – used to read or write data in byte format

Random – used to place characters in a buffer of a set size

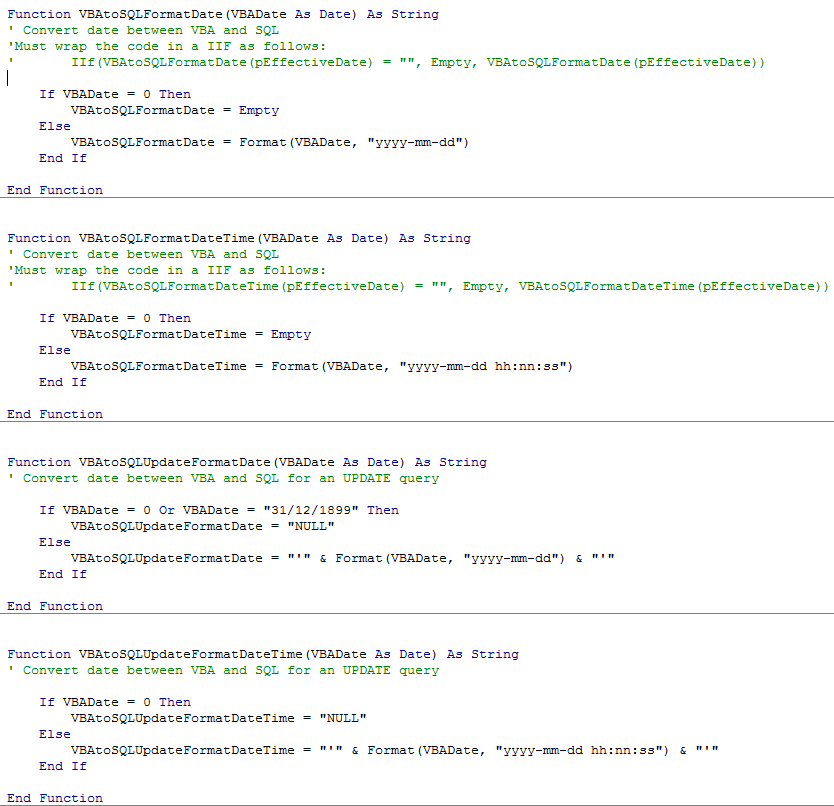
FileNumber – Any number between 1 and 511, first file should be 1 and then so on, always preceeded by #.

EOF = End of File, used to read through a file (in a loop probably) until there are no more lines to read from.

## Dates in VBA

Dealing with date formats between Excel, VBA and SQL is probably the most painful thing you’ll ever do. Within M01\_General of the AddIn we have a number of functions established to do the correct version based on the circumstance, i.e. taking a date entered on the worksheet and storing it in VBA as a variable, or using it in an SQL update query.

Some examples;



## Using Buttons

The application uses shapes from Excel’s Insert > Illustrations menu to act as buttons, as they are customisable and therefore visually more appealing than macro control buttons. These are coloured in a standard Generali red when active, and a grey when inactive. Example code to manage buttons for a particular sheet is shown below.

