**FAPI Inventory Import App**

The classes and code should be fairly well commented to explain what it does and how. Dot Net 4.51 and the Office 2007 interop are used and required for the app. Also, AccessDatabaseEngine needed to be installed prior to Manifest upload program.

**Modules/Forms:**

1. FormFAPI\_Import is the main windows form.
   1. Dropdowns are populated from tables in the PBIApplicationTables database. E.g the Warehouse dropdown data is from the ‘Famous\_Warehouse\_Data’ table.
      1. When selected the sWarehouse string variable is set.
   2. The menus are the following.
      1. File
         1. Open – opens manifest spreadsheet using the selected template to locate the data within the imported spreadsheet.
            1. Brings up a dialog box to select an excel file.
            2. After the initial opening of the excel file, a dialog box is presented to select a sheet from the workbook.
            3. The worksheet is opened using MS ACE OLEDB version 12
            4. Data Adaptor is used to fill the ImportedVesselDtaDataSet with the header vessel information which populate the variables and displays on the main form.
            5. Next the data part of the spreadsheet is put into the ImportedDataDataSet with the Data Adaptor object.
            6. Columns that are missing from the standard standard Pandol manifest are added to the ImportedDataDataSet to be used by the app during processing.
            7. Data Grid View of the main form is then filled with the importedDataDataSet data.
            8. Data stored in the ImportedDataDataSet is used throughout the app.
         2. Export – will export the imported data
         3. Exit
      2. Templates – opens the Import\_Template\_Selection form.
         1. Import\_Template\_Selection class
         2. Fills the TemplateSettings data row and returns it. This data row variable is used throughout the app to identify the location of the data in the spreadsheet. Data stored in the FAPI\_Import\_Templates table.
      3. Tools
         1. Update Translations
            1. Opens a FrmGrowerCodeTranslation form. This is the edit and creation form for the grower codes and translation tables used in translation of the manifest spreadsheets. Data stored in the GCV\_Information2 and Translation\_Details2 tables.
         2. Export Code Editor
            1. Opens a Adams\_Codes\_Editor form. This is the edit and creation form for the Adams translations. Data stored in the Adams\_Values table.
         3. Stone Fruit List
            1. Opens a StoneFruitList form to edit the list of stone fruit commodities. Data stored in the StoneFruitCommodities table.
         4. Template Creator
            1. Opens a TemplateCreator form. This form and the associated classes are used to create new templates for the imported manifest spreadsheets. Data stored in the FAPI\_Import\_Templates table.
         5. Update From Famous
            1. Updates the Famous tables in the database for the app. Runs the UpdateTablesFromFamous stored procedure and refreshes the app data where needed.
   3. The Vessel Information section displays the data pulled from the spreadsheet and allows editing if needed.
   4. The data Imported/Translated data grid view shows the imported data as it is being processed. It can also be edited here as well.
   5. <Translate> button begins the translation process.
      1. Instantiates a DataTranslatorByGrower object/class, passing the import data dataset, Importsettings, list of data to translate, grower block and the pallet prefix.
      2. Running the translate data method, translates and returns the translated data dataset.
      3. Displays the translated data in the data grid view in the main form.
   6. <Validate button>
      1. Validates the data through the DataValidation object/class with the validate method
      2. If data is valid then the export buttons are available to export the data.
   7. <Famous Export> button
      1. Runs the Export to Famous method in the main app
         1. Checks the variables from the dropdowns and vessel information
         2. Stops export and informs user of missing information.
         3. Or Checks to see if export should be in XML formation or text.
            1. Text is obsolete and no longer used but is still available.
         4. Instantiates a FamousXMLExporter object/class with ImportedDataDataset, ImportSettings and ExportDataList being passed to the exporter object.
         5. Gets the file name from the user through the SaveFileDialogBox object/class and stores it in the tempfile variable which is later passed to the
         6. Archives the data through the Archive\_Data object and Store\_Data method.
         7. Exports the data from the FamousXMLExporter through the ExportList method to the WriteExportTextFile object/class. Which then writes the data to a file. File name is passed with the WriteFile method.
   8. <Export> button (Adams Export)
      1. Instantiate an AdamsTranslator object/class. Passing ImportDataset, ImportSettings and AdamsList2Translate.
      2. Instantiate an AdamsExportToExcel object/class. Passing the AdamsTranslator object with the TranslateData method to run the translation, ImportSettings, exportFullDataList.
2. Famous XML Exporter.
   1. Holds the following.
      1. Data2Export dataset
      2. ExportStringlist stringlist
      3. ImportSettings object
   2. Uses the CombineMixedPallets object to select the dominate type produce for used each mixed pallet.
   3. Has the following methods
      1. ExportData()
         1. Creates the export list form the Data2Export dataset.
            1. Checks to see if the receipt number has already been used, via the IncrementRecieptNumber object
         2. Creates the export list
      2. ExportList
         1. Gets the export list from the exporter object
      3. GetReceiptNumber()
         1. Get the object’s receipt number
      4. GetReUseReceiptNumber()
         1. Get the receipt number to reuse
      5. ExportConversionSucceeded
         1. Checks to see if export process succeeded
3. Adams Exporter to Excel
   1. Uses the following objects during instantiation to create the Adams export excel sheet.
      1. Data2Export dataset
      2. ExportStringList stringlist
      3. ImportSettings object
   2. Uses the CombineMixedPallets object to select the dominate type produce for used each mixed pallet.
   3. Has the other following methods
      1. WasExported() Returns true of succeeded
      2. Excel FileName() Returns the filename used.
4. Data Translator by Grower
   1. Translates the data from the import data which was extracted from the manifest spreadsheet by grower and produce type.
   2. The following are passed during instantiation
      1. Data2Export dataset
      2. ExportStringlist stringlist
      3. ImportSettings object
      4. GrowerID string
      5. Pallet prefix string
   3. Then it gets the translation data from the database and stores it in a local table, TranslationTable, which is used for the translation process.
   4. The method TranslateData() translates the import data and returns a translated dataset which is used by the app for the export process
      1. It uses the Famous\_Grower\_Block\_Data and Famous\_Variety\_Data tables to create a local GrowerBlockTable. This is used to find the variety codes from the grower block information.
   5. Then completes the data translation process and returns the translated dataset.
5. Adams Translator
   1. Translates the importdata which has been translated to Famous form into Adams format for export.
   2. The following are passed during instantiation
      1. Data2Export dataset
      2. ExportStringlist stringlist
      3. ImportSettings object
      4. List2Translate stringlist (items to translate can be controlled by the main app if needed)
   3. The following methods are available and used by the app.
      1. TranslateData()
         1. First it gets the translation data from the Export\_Values table in the database and puts them in the local TranslationTable for use during the translation process.
         2. Runs the translation process.
         3. Returns the translated data dataset.
6. Data Validation
   1. Validates the data translated to Famous format with data from the Translation\_Validation\_Table in the database. This table is updated from the Famous Oracle database daily or when the update is run manually by the user through the app.
   2. The data from the database Translation\_Validation\_Table is used to create the local ValidationTable.
   3. Using the list of parameters to update, the routine runs the update process to create a list of invalid items and returns true if completed without errors.
   4. The method ListofInvalidItemLocations() returns the location of invalid translation data in the dataset.
7. Archive Data
   1. This class archives the import data to the import data warehouse database.
   2. The following parameters are passed during instantiation.
      1. DataToArchiveDataSet from the main imported dataset.
      2. ImportSetting object
      3. ArchiveStringList stringlist data from the app controls
      4. Receipt number
   3. Data is first inserted into the FAPI\_Import\_Data\_Archive\_Temp table in the database.
   4. Then the stored procedure sp\_UpdateManifestUploadArchiveInfoVersion2 is run to update the archive table from the temp table.
      1. Part of the process is to make additions for reporting.
8. Template Creator
   1. This is the windows form and class to allow the user to create templates for the import of the manifest spreadsheets from different farms and exporters who do not used the standard Pandol manifest spreadsheet.
   2. The routine walks the user through identifying the data locations in the spreadsheet and creating a template which is stored in the Spreadsheet\_Import\_Templates table in the database through the UpdateCreateTemplate stored procedure.
      1. If a template is already there by the same name, it is updated otherwise a new one is inserted into the table.
   3. A check is done at the end of the process and users are informed of missing data or errors.
   4. The restartClear() method with reset the creation process so the user can start over.
9. TranslationEditor
   1. This windows form allows the user to add and edit data translation information by exporter, commodity and variety.
   2. Famous codes are all populated from Famous data and offered in dropdowns.
   3. The data is retrieved for editing and stored in the GCV\_Information2 and TranslationDetails2 tables.
   4. Updated or new data is stored in the tables via the UpdateGCV\_InfoandTranslationDetails2 stored procedure.
10. Adams\_Codes\_Editor
    1. Windows form to allow the user to create or edit the Adams Export\_Values table which is used for the translation of Famous codes to Adams coded.