Steps to Import
Customer Part Spreadsheets

Description

This document describes the process of importing a customer parts list from the template developed during the 4DX process. The document may be a Google Document and shared with the customer while they are filling it in, but will be exported to Excel to work with it.

The file format is very specifically defined. Among the rules are:

1. The first row is the column names.
2. The spelling of the column names must match exactly, including spaces.
3. The following column names are required in every tab:
    a. Import
    b. Active
    c. Part Name
    d. Part Description
    e. Unit Cost
4. The following columns are supported if present but are not required:
    a. Part To Clone
    b. Part Category
    c. Parent Part Category
    d. Units
    e. SKU
    f. Any UDF
5. If a Parent Part Category is given and the Part Category is not found under it, it will be created during the import.

6. If a Parent Part Category is given it must exist.
7. If a Part Category is given and no Parent Part Category is specified, the Part Category must exist.
8. The Units must match one of the existing standard or abbreviation unit formats found in the _Unit table. Each is used if no unit is given.
9. Avoid formatting with $.

Creating Import Files from The Spreadsheet

For the import, we will create a separate comma delimited file for each Tab in the spreadsheet. To make this simpler, follow these steps:

1. If you are using a Google Spreadsheet, export it to Excel.
2. Open the spreadsheet in Excel
3. Make sure the Spreadsheet is not in ReadOnly or Restriced mode.
4. Press Alt + F11 keys simultaneously to open the Microsoft Visual Basic Application window.
5. In the Microsoft Visual Basic Application window, click Insert > Module.
6. Copy and paste the following code routines into the Module window.

```
Sub ExportSheetsToCSV()
  Dim xWs As Worksheet
  Dim xPath As String
  xPath = "<<path>>"
  Dim xFile As String
  For Each xWs In ThisWorkbook.Worksheets
    xWs.Copy
    xFile = xPath & "\" & xWs.Name & ".csv"
    Application.ActiveWorkbook.SaveAs Filename:=xFile,
FileFormat:=xlCSV, CreateBackup:=False
    Application.ActiveWorkbook.Saved = True
    Application.ActiveWorkbook.Close
  Next
End Sub
```

For some reason, this VBA does not always work. If you get an error message, you can use this version

```
Sub ExportSheetsToCSV()
    Dim xPath As String
    Dim xFile As String
    xPath = "C:\Users\scott\Desktop\Customer Data\Tabs"
'Application.ActiveWorkbook.Path
```

```
    For Each xWs In ThisWorkbook.Worksheets
        xFile = xPath & "\" & xWs.Name & ".csv"
        xWs.SaveAs Filename:=xFile, FileFormat:=xlCSV,
CreateBackup:=False
    Next
    Application.ActiveWorkbook.Close
End Sub
```

7. Set the directory you want the Tab delimited files created in the **xPath** variable. I recommend you create a sub-directory called Tabs for these files
8. Press the F5 key to run the code.
9. The screen will blink a bunch. Don't worry, this is just old 1993 code so it's not the prettiest.
10. You will see all exported CSV files in the specified folder.

Importing into SQL

Now that you have the CSV files, you need to get them into the Database. This routine brings in all of the CSV files into their own tables called **PartImport.{filename}**.

You'll have to add a lot of stored procedures and functions to the database for this to run. See Scott or a consultant for the latest versions.

Enabling CMD Shell and Ad Hoc Distributed Query Calls in SQL

In order to retrieve the list of files, SQL must make a directory call to read the contents of the folder. By default, this access is denied in SQL since it could be used by malicious software to access the local drive.

To enable these features, run these SQL commands

```
EXEC sp_configure 'show advanced options', 1
GO
RECONFIGURE;
GO
sp_configure 'xp_cmdshell', 1;
GO
RECONFIGURE;
sp_configure 'Ad Hoc Distributed Queries', 1;
GO
RECONFIGURE;
```

You also need to download and install Access Database Engine(AccessDatabaseEngine_X64.exe). You can get the file using this link https://www.microsoft.com/en-us/download/details.aspx?id=39358.

Importing the CSV Files

To import the files as parts involves these steps:

1. Add the following procedures and functions to SQL if they don't exist (they probably won't)
    a. csp_ImportPartSpreadsheetTabs
    b. csp_ImportPart
    c. csp_ImportPartCategory
    d. csf_MergeXML
    e. csf_String_Split
    f. csf_TextToBit
    g. csf_TextToDecimal
    h. csf_TextToUnitID
    i. csf_TextToUnitType

2. BACKUP AND RESTORE A TEST DATABASE. While I show SQLs below to delete the import, it will delete a lot of other things to and you should only plan to import on the live database after you have successfully tested everything on the backup.

3. Run the following SQL command to execute the import. (You'll need to fill in the actual path first)

```
DECLARE
      @Path          VARCHAR(255) = '{**fill in**}'
    , @SkipRows      VARCHAR(255) = 3
    ;

EXEC csp_ImportPartSpreadsheetTabs
      @Path          = @Path
    , @SkipRows      = 3
    , @DontCommit    = 0  -- 1 to test only but not save, 0 to save
```

4. Note that the import produces a lot of single row table outputs (one for each part added).

5. **You MUST** click on the Messages tab in SQL Management Studio and scroll down the list to see any errors. Most errors will not appear on the table output and you will only know if there are problems if you scroll through that list looking for red output.

6. If the UDF names are not correct, there is no warning. You'll have to check the parts to find that out.

Backing Out the Import

If you made a mistake and want to re-import, you can delete all of the parts that were imported that day with this query, fix the data, and then run it again. This will delete anything done that day, whether by you or a user so only run this on a backup of the data.

```
DELETE from Part
where ModifiedbyUser = 'csp_ImportPartSpreadsheet'
DELETE from PricingElement
where ModifiedbyUser = 'csp_ImportPartSpreadsheet'
DELETE from PartInventoryConversion
where ModifiedbyUser = 'csp_ImportPartSpreadsheet'
DELETE from Inventory
where ModifiedbyUser = 'csp_ImportPartSpreadsheet'
DELETE from InventoryLog
where ModifiedbyUser = 'csp_ImportPartSpreadsheet'
```

Notes and Areas Not Implemented

Not all areas are currently implemented. These include:

- Default Units. All units must be specified on every column. A quick copy-and-paste is usually all that is needed for this. Any units not specified are treated as Each.
- Setting products Inactive with the Active = No value.

Other Notes:

- The routine does not delete the import files when it is done (though it will overwrite them if you re-import). You may want to clean them up when you are done.