Supplier Cost Refresh Technical Design

Problem

Over the last several months, supplier catalogs have been extracted from the BlueCube production database and imported into the future ESO Production database. This is done to prepare for a pilot phase and system cutover from BlueCube to ESO. This process established the catalogs with the cost at the time of the extraction from BlueCube, so any cost changes made after the extraction are not reflected in ESO. There is not one specific point in time that can use used to determine what price changes have been missed because incremental new items and new or missed suppliers must be accounted for, however, this can be approximated using the logs of the extraction processed and internal timestamps.

The JDA Catalog import does not support multiple prices for a supplier cost level, so it is also the case that promotions and future price changes have not been imported into ESO correctly.

There is no JDA import for cost change events, however, it is possible to leverage the JDA catalog import to simulate cost change events by importing multiple catalogs for distinct cost changes.

To avoid excessive data entry and errors, a process will be implemented to extract the missing cost changes from BlueCube and import the changes into ESO. This will be one time per supplier as the supplier is needed for stores cutting over to ESO.

Solution

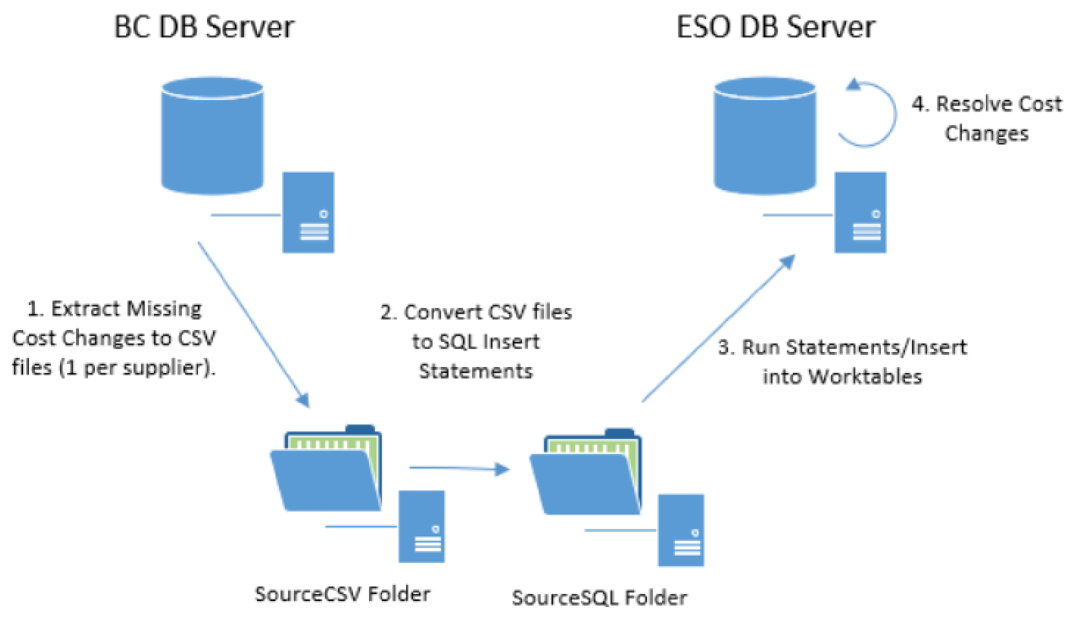
The desired solution will pull costs from the source BlueCube system that were not included in the original load of the supplier catalog files. This data will include costs that are currently in effect and are in effect in the future for both normal cost changes and promotional cost changes. Once extracted from the BlueCube system these costs will be imported into staging tables. The final step will be to extract the supplier data along with the missing costs into excel spreadsheets or csv using the existing catalog extract and import framework. This will allow the generation and import of standard JDA catalog import files.

The solution has two main sets of steps. The first set of steps extracts data from BlueCube and imports it into ESO staging tables. The second set of steps creates catalog import files with the item details and distinct cost imported from BlueCube.

Cost Refresh - Sync DBs

Process Overview

The Sync DB process is based on a custom extract of the missing cost changes from BlueCube and a custom import of the changes into ESO. The result will be a missing supplier item cost worktable in the ESO database.


BC DB Server
ESO DB Server
4. Resolve Cost Changes
1. Extract Missing Cost Changes to CSV files (1 per supplier).
2. Convert CSV files to SQL Insert Statements
3. Run Statements/Insert into Worktables
SourceCSV Folder
SourceSQL Folder

Detailed Steps

Extract Missing Cost Changes. This is executed on the source BlueCube server using the command file “cmdStep1 - DB - To – SourceCSV” located in the data extraction and loading folder named “Cost Refresh – Sync DBs”. It will create csv files import folder structure in the folder “SourceCSV”.

Although this extract produces files by supplier, it is driven off a list of business units. If the supplier is assigned to a business unit specified in the script, the supplier costs will be included in the csv file out. In the sql script file “SupplierItemCostRefreshCountForExtract” there is a list of business units. This list should be edited to include the appropriate business units using the business unit code.

```
DECLARE @supplier TABLE (supplier_id INT, name nvarchar(128))
DECLARE @business_unit TABLE (business_unit_id INT)

INSERT @business_unit
SELECT data_accessor_id
FROM rad_sys_data_accessor AS rsda
/* Insert the list of business units */
WHERE rsda.name IN (
'0006953',
'0007001',
'0007640',
'0005402',
'0006303',
'0006096',
'0006031',
'0006523')
```

Convert CSV data to SQL statements. This step does not require a DB connection. The command file “cmdStep2 - SourceCSV - To – SourceSQL” will convert the csv data in insert statements on a file by file basis and place the files in the “SourceSQL” folder.

Insert missing cost data into ESO worktables. This step is executed on the ESO DB server using the command file “cmdStep3 - SourceSQL - To - Dest DB” located in the data extraction and loading folder named “Cost Refresh – Sync DBs”. It will place data into the ESO cost import worktables. This step will set the status of the import to “imported.”

The final step is to resolve the cost changes. This step is executed on the ESO DB server using the command file “cmdStep4 - Resolve Cost Changes” located in the data extraction and loading folder named “Cost Refresh – Sync DBs”. It will update data within the ESO cost import worktables. Id values for supplier, supplier items, and cost levels will be resolved. This step will also mark any cost changes that are not needed as invalid, this is added to limit unneeded data. This step will set the status of the import as “ready for export.”

Worktables

BlueCube Database

The extract from BlueCube requires a worktable to keep track of prior exports. This will allow the process to be rerun but also keep a log of what criteria was used for the export and the time the export was executed. This table should be created in the bcssa_custom_integration database using the sql script file SupplierItemCostRefreshCreateBCTables.

Table bc_extract_cost_export.

Column Name	Data Type	Notes
ExportId	int	Identity
SupplierId	int	The BlueCube supplier Id
SupplierXrefCode	nvarchar	The BlueCube supplier Xref Code
EffectiveDate	smalldatetime	The Effective Date – This is what was used to determine which cost changes to export. Those that are still in effect or in effect in the future as of this date are included.
ExecutionDate	smalldatetime	Records the execution date and is used as the next effective date if the same supplier is exported again.
ExportedTimeStamp	smalldatetime	The server time that the csv file was created.

ESO Database

The ESO database requires two tables, the bc_extract_cost_import is somewhat of a mirror image of the export table in BlueCube. The bc_extract_cost_import_supplier_item is a child table of the import table and contains the imported costs from BlueCube. Both tables should be created in the VP60_Spwy database using the sql script file SupplierItemCostRefreshCreateESOTables.

Table bc_extract_cost_import.

Column Name	Data Type	Notes
ImportId	int	Matches the ExportId from the imported data
SupplierXrefCode	nvarchar	Matches the Xref Code from the imported data
ResolvedSupplierId	int	Matches the ESO id, resolved during the resolve cost changes step.
StatusCode	nvarchar	Has 3 possible values, i = Imported from BlueCube, r = ready for export to CSV, e = exported to CSV.
EffectiveDate	smalldatetime	The effective date of the changes associated with the import.
ExecutionDate	smalldatetime	The date the data was generated.
ExportedTimeStamp	smalldatetime	The date-time the data was generated.
ImportedTimeStamp	smalldatetime	The date time the data was imported into ESO

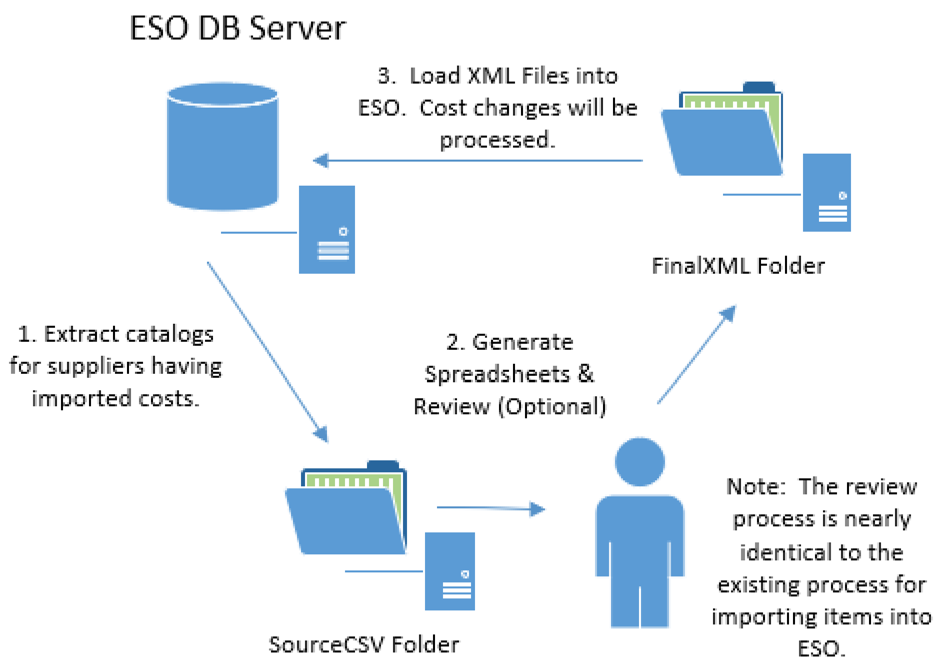
Table bc_extract_cost_import_supplier_item.

Column Name	Data Type	Notes
ImportId	int	Matches the ExportId from the imported data
SequenceNumber	int	The unique identifier of a cost change per supplier item and cost level. This is needed because the catalog import does not allow multiple cost changes per cost level. The number of distinct sequence values for each supplier will match the number of files produced when the data is exported.
SupplierItemCode	nvarchar	Matches the imported data.
ResolvedSupplierItemId	int	Matches the ESO id, resolved during the resolve cost changes step.
CostLevelName	nvarchar	Matches the imported data.
ResolvedCostLevelId	int	Matches the ESO id, resolved during the resolve cost changes step.
SupplierPrice	smallmoney	Matches the imported data.
SupplierAllowance	smallmoney	Matches the imported data.
StartDate	smalldatetime	Matches the imported data.
EndDate	smalldatetime	Matches the imported data.
PromoFlag	nvarchar	Matches the imported data.
IsValid	nvarchar	Defaults to “y” (True). Can be marked as invalid during the resolve cost changes step.

Cost Refresh - Items

Process Overview

The Items process is very similar to the process used to extract supplier catalog data from BlueCube and load the catalogs into ESO using the JDA import process. The major difference is that these catalogs are extracted from the ESO database with the exception that the cost is taken from the import cost worktable. This means that the xml from the catalogs can be created using the same base configuration data for the supplier items, however the catalog import files will contain costs based upon BlueCube data.


ESO DB Server
3. Load XML Files into ESO. Cost changes will be processed.
FinalXML Folder
1. Extract catalogs for suppliers having imported costs.
2. Generate Spreadsheets & Review (Optional)
SourceCSV Folder
Note: The review process is nearly identical to the existing process for importing items into ESO.

File naming convention and sequence

Every file created during this process will have a naming convention of “Cost Refresh - Items-***supplier xref***-***supplier name_sequence number***”. At least one file with sequence number 1 will be generated if the supplier has any unprocessed imported cost changes. The sequence number is required in order to generate unique files if more than one cost change is available for a supplier item cost level. This is due to a limitation in the JDA import. For example, if item Coke, Case 24 had a normal missing cost change on August 24 and a promotional cost change on September 1, two files would be created with a sequence of 1 and 2. When completely the final step of loading the XML into ESO, it is advisable to load the files in order by sequence. Process all files with sequence 1 first and then proceed with any remaining files. This will ensure consistency with the BlueCube historical sequence of cost changes.

Detailed Steps

Extract Missing Cost Changes. This is executed on the ESO server using the command file “cmdStep1 - DB - To - SourceCSV” located in the data extraction and loading folder named “Cost Refresh – Items”. It will create csv files in the import folder structure within the folder “SourceCSV”. This step will set the status of the import as “exported”. If for some reason this process fails, the status code in the bc_extract_cost_import table can be reset to “r”, ready for export and the command can be executed again.

Review Steps. (Optional)

Note: The review spreadsheets use the same format that is used for catalog import spreadsheets, however some information such as the barcodes and keywords for availability in the item description (i.e., “*INACTIVE*”) have been omitted.

Execute the command file “cmdStep2 - SourceCSV - To - WorkingXLS” located in the “Cost Refresh – Items” folder. This will create excel spreadsheets in the “WorkingXLS” directory.

After the files in the WorkingXLS directory are reviewed and edited, place the files in the “FinalXLS” folder.

Execute the command file “cmdStep3 - FinalXLS - To - FinalCSV” located in the “Cost Refresh – Items” folder. This will create excel spreadsheets in the “FinalCSV” directory.

Note: More complete information regarding this process can be found in a document located in the documentation folder of the data extraction and loading package titled “BC to ESO - Process Overview and Spreadsheet Instructions”.

Without Review

Copy the contents of the “SourceCSV” folder to the “FinalCSV” folder.

Create the XML. Once files are available in the “FinalCSV” folder execution the command “cmdStep4 - FinalCSV - To – FinalXML” to create files in the “FinalXML” folders.

Import the XML. Once the XML is generated, import the files using the standard JDA ESO import method. Note, like all catalog imports consideration must be given to the state of the “review catalog import” flag on the supplier. In order by bypass this step, the flag must be set to false when the catalogs with the updated costs are imported.