****

**NYC Checkbook2.0 ETL Implementation Approach**

**Prepared for**

**Friday, July 29, 2011**

**Version .0**

**Author**

**Athammai Thiagarajan**

Revision and Signoff Sheet

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** |  |
| 08/01/2011 |  | 0.1 |  |
| 08/05/2011 |  | 0.2 |  |

Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Version approved** | **Position** | **Date** |
| Athammai Thiagarajan | 0.1 | Principal DBA | 08/01 |
| Kishore Vuppala | 0.2 | Technical Lead | 08/05 |

**Table of Contents**

[Background 5](#_Toc300566315)

[Scope 5](#_Toc300566316)

[Overview 5](#_Toc300566317)

[ETL Related Tables 6](#_Toc300566318)

[ETL Process 8](#_Toc300566319)

[GetDirectoryListing 9](#_Toc300566320)

[IsEligibleForConsumption 9](#_Toc300566321)

[ProcessDataFile 9](#_Toc300566322)

[Stage and archive the data 10](#_Toc300566323)

[Validate data 10](#_Toc300566324)

[Process staged data 12](#_Toc300566325)

[Process agency 12](#_Toc300566326)

[Process department 12](#_Toc300566327)

[Process expenditure object 13](#_Toc300566328)

[Process location 13](#_Toc300566329)

[Process object Class 13](#_Toc300566330)

[Process FMSV 14](#_Toc300566331)

[Process MAG 14](#_Toc300566332)

[Process CON 15](#_Toc300566333)

[Process FMS 16](#_Toc300566334)

[Process Budget 18](#_Toc300566335)

[Process Revenue 18](#_Toc300566336)

[Questions 18](#_Toc300566337)

[Appendix A 19](#_Toc300566338)

[COA data validations 19](#_Toc300566339)

[FMSV data validations 20](#_Toc300566340)

[MAG data validations 21](#_Toc300566341)

[CON data validations 22](#_Toc300566342)

[FMS data validations 24](#_Toc300566343)

[PMS data validations 25](#_Toc300566344)

[Budget data validations 26](#_Toc300566345)

[Revenue data validations 27](#_Toc300566346)

## Background

NYC Checkbook2.0 is intended to develop a transparency dashboard for NYC financial reporting to integrate Budget, Contracts, Expense, Revenue, Capital and Payroll. Different data feeds namely COA, MAG, CON, FMSV, FMS, PMS, Revenue and Budget from data sources such as FISA and PMS is being used to provide data for this dashboard. ETL Implementation Approach process for NYC Checkbook2.0 is proposed after having done a preliminary analysis on the mentioned data feeds (excluding Capital, Payroll detailed information, Revenue Budget, Pension Funds and Pending Contracts) and based on the current implementation of ETL process which consumes the data from the various feeds and publishes the data to three SQL Server databases namely MyMoney, MyMoneyPublic & MyMoneyPublicR.

## Scope

This document provides an insight into the ETL Implementation Approach for NYC Checkbook2.0. ETL process starts with the consumption of raw data, validating, processing and posting it to the transaction tables.

## Overview

Kettle, Pentaho data integration tool will be used to integrate the various ETL processes to fetch files from remote server(s), loading data to database from flat files, data cleansing etc.

Master kettle job will be created which has several transformations and invokes several other jobs. This will be started around 5 AM when the files are published by FISA. Processing of the data begins only when the required files are available.

Assuming that MyMoney, MyMoneyPublic and MyMoneyPublicR will be obsolete with NYC Checkbook2.0 implementation, NYC Checkbook2.0 will be used as the source for MWBE application or for any reports that are being generated currently against MyMoney. Also since NYC Checkbook2.0 provides the history of contract modifications, there is a need to capture all the versions of a contract document. Disbursements which are not be displayed to public or which can be partially displayed only (masking its associated vendor, contract information) will be captured in the same database. For the above mentioned reasons, data which can be displayed to public in full/partial with masked information will be stored separately from the table which has all the data as from the feeds. This is covered in detail in ProcessDataFile section of this document.

Tables are organized in such a way that ETL related tables are stored in “etl” schema and the tables accessed by the Checkbook2.0 application are in the public schema. For most of the transaction tables corresponding database sequence is created in the public schema for generating the primary key.

Pictorial representation of ETL process is provided below.

Executed for each of the data file in the order of data source

Invoked every 15 min till 8 AM as per SLA for the data feeds where files are still not available.

The master job invokes the below transformations.

* GetDirectoryListing - For every 15 minutes, check for files from FISA server which need to be downloaded.
* IsEligibleForConsumption - Determine the files which need to be consumed as multiple files can be available for the same data feed
* ProcessDataFile - Process each of the files that can be downloaded (Highlighted in green).

## ETL Related Tables

Tables are organized in such a way that ETL specific tables which include certain reference tables, all the staging tables, archive tables, invalid tables (to store invalid/malformed records) are stored in the “etl”schema.

Few columns are added to the staging tables to identify each record, invalid\_flag, invalid reason, action\_flag which can be I(Insert)/ U(Update) to signify what kind of action need to be done in the transaction table, column corresponding to the primary key of the transaction table, columns corresponding to the FK columns of the transaction table etc.

Key reference/lookup tables for ETL includes ref\_data\_source, etl\_data\_load, etl\_data\_load\_file and ref\_validation\_rules, each of these tables is explained in detail below

**Table ref\_data\_source:**

Details about the various data sources are stored in ref\_data\_source table. This table provides key information such as the record identifiers for heterogeneous data which are considered for NYC Checkbook2.0, the name of the staging/archive/invalid tables where the data is staged and archived.

Columns of ref\_data\_source are data\_source\_code, document\_type, record\_identifier, staging\_table\_name, archive\_table\_name, invalid\_table\_name, sql\_condition, data\_source\_order. Possible data sources identified as of now includes Agency (A), Department (D), Expenditure Object (O), Location (L), FMSV (V), MAG (M), CON (C),FMS (F), PMS (P), Budget (B), Revenue (R). Document type (document\_type) column is strictly used for CON data as the format for various document types are different. It holds the values like CT1, CTA1 etc. Record identifier is used if the file contains heterogeneous data and the values are like H for header, W for award detail etc. Data from the file is dumped into one single table first and then each type of record is added to the specific table based on the SQL query generated using the information in ref\_column\_mapping in the order of table\_order. data\_source\_order column determines the sequence in which data files are downloaded.

**Table ref\_column\_mapping:**

Mapping between the columns in the data feed table and the staging table are captured in this table. Columns include data\_feed\_table\_name, data\_feed\_column\_name, data\_feed\_data\_type, staging\_table\_name, staging\_column\_name, staging\_data\_type, column\_order. This table is required as the data feed table usually has varchar as the data type for most of the columns due to heterogeneous data type and staging table has the appropriate data types. Values will have to be converted according to the staging table definition and then inserted into the table.

**Table ref\_ file\_name\_pattern:**

Provides the file name pattern for each of the data source. Columns include data\_source\_code, directory\_listing\_pattern and actual\_pattern. Directory\_listing\_pattern is used to list the files matching the criteria in the remote server and actual\_pattern is strictly used to validate the name of the file matches.

**Table etl\_data\_load:**

Table to capture the processing status of various data sources. It contains as many distinct data source codes as in ref\_data\_source\_table for the given job identifier associated to ETL process.

Columns are the job\_id associated with the master ETL process, load\_id (1 for each data source), publish\_start\_time, publish\_end\_time, files\_available\_flag (signifies if file is available for consumption).

**Table etl\_data\_load\_file:**

Stores the details of files for consumption for each of the load\_id in etl\_data\_load. Columns include load\_file\_id (Primary Key), file\_name, file\_timestamp, type\_of\_feed (M for monthly, D for daily, W for weekly), display\_type (used for FMS only), consume\_flag, pattern\_matched\_flag, and processed\_flag. File\_timestamp is the timestamp associated in the file\_name itself. Consume\_flag by default is N and is set to Y only if the file has to be consumed. Pattern\_matched\_flag will be set to Y when the name of the file matches the pattern as in ref\_file\_name\_pattern. Processed\_flag can be N/Y/E/C for not processed/ Processed/ Error/ Cancelled depending on the status of the processing.

**Table ref\_validation\_rules:**

This table lists the various validations done on each record in the data feed files consumed. Validations are broadly classified under 7 categories namely Missing key elements, Invalid/Inconsistent values, Missing parent entity, Duplicate records for the unique identifiers, Invalid components, Multiple records & Inter-load duplicates. Columns of this table are rule\_id (Primary Key), data\_source\_code, record\_identifier, document\_type, rule\_name, parent\_table\_name, component\_table\_name, staging\_column\_name, join\_condition, ref\_table\_name, ref\_column\_name, invalid\_condition, order. Document type is provided only for Contracts data feed. Rule\_name identifies the category, parent\_table\_name is the table associated with the corresponding header record and is provided only when category is having parent entity, component\_table\_name is the name of the table associated with the different components of a record ( business type, address, address type are components of vendor), staging\_column\_name is the field which is being validated, join\_condition provides the join condition to be used for records in the staging table and in the component tables, reference\_table\_name is the name of the reference table, reference\_column\_name is the name of the column in the reference table that is used for validation, invalid\_condition states the SQL condition to be executed. Order column determines the order in which validation needs to be done. This is very important as the check for invalid component should be done at the end only.

Sample records for the different ETL reference tables is attached below



## ETL Process

ETL process starts with the directory listing of FISA server and once the required details are obtained, each of the data file is processed & changes are posted in the transaction table in the order of COA/ FMSV, Revenue/Budget/MAG, CON, FMS/PMS. The only way to know if a vendor/ master agreement/ agreement should not be made available to the table accessed by the application is through disbursements which are being processed only after FMSV, MAG & CON data files. For this reason till FMS data file is processed sensitive information related to Contracts Data may be available in the public table. Probability of this sensitive information being exposed to the public by the application is very less since FMS data file is processed the same day. However if the FMS file is not available for processing and MAG/CON data files are available, the Contracts data may be exposed to the public.

Various transformations in the master job are explained in this section. If any error occurs in a transformation a DB function is executed by passing the load file identifier as a parameter to determine if ETL job can continue with the next data file or it has to abort the entire job.

* If any of the COA data file result in an error the entire job is aborted. Processed\_flag is set to E for the corresponding file and all data files which had to be consumed will be set to C for cancelled.
* If FMSV data file resulted in an error, processed\_flag is set to E for the corresponding file and all other FMSV/MAG/CON/FMS data files which had to be consumed will be set to C for cancelled.
* If MAG data file caused an error, processed\_flag is set to E for the corresponding file and all other MAG/CON/FMS data files which had to be consumed will be set to C for cancelled.
* If CON data file caused an error, processed\_flag is set to E for the corresponding file and all other CON/FMS data files which had to be consumed will be set to C for cancelled.
* If FMS/PMS/Revenue/Budget data file caused an error processed\_flag is set to E for the corresponding file and all other files from the same data source which had to be consumed will be set to C for cancelled.

### GetDirectoryListing

ETL process starts with this step where a record is created for each of the data sources in etl\_data\_load table with the job\_id specific to the entire ETL job and files\_available\_flag as N. Directory listing is obtained for each of the data sources every 15 minutes if the required files are not available until 8 AM. Output from the directory listing is inserted into the table etl\_data\_load\_file if there is no record in the table corresponding to this file name with processed\_flag other than E/C. Type\_of\_feed, file\_timestamp, display\_type (P/X/F for Partial, Not Displayed, Displayed in Full) are updated based on the file name. Consume\_yn and pattern\_matched\_yn is set to Y if it matches with the ref\_file\_name\_pattern.actual\_pattern and files\_available\_flag for the corresponding load\_id in etl\_data\_load is set to Y if pattern\_matched\_yn is 1.

**Note:** Exit criteria from this transformation are when files are available for all of the data sources before 8 AM or if it is 8 AM and at least COA files are available.

### IsEligibleForConsumption

Once all the required files are available, for each of the data source, files to be consumed are identified. For COA data sources, file with the latest timestamp is considered and others are not. As FMSV can have a combination of monthly (full), daily/weekly (incremental), monthly file with the latest timestamp and daily/weekly files submitted thereafter are only considered. Any files prior to the latest monthly submission will not be considered for consumption. For the remaining feeds all data files are considered.

### ProcessDataFile

Files which are to be consumed are processed in the order of the data source (as in ref\_data\_source table) in the order of file\_timestamp in etl\_data\_load\_file. Each of the file goes through several transformations before being made available in the public schema. Status of the file is verified to ensure processed\_flag is not C (Cancelled). Following transformations are done if the file can be processed.

#### Stage and archive the data

Function: stageandarchivedata

Input parameters: file identifier as in etl\_data\_load\_file

Return value: 1 for successful staging and 0 for any errors.

1. Data in the staging tables as identified in ref\_data\_source for the corresponding load is truncated.
2. File is downloaded to the local directory.
3. In the order of table\_order in ref\_data\_source data is loaded into the various staging tables. For instance MAG file is first loaded into stg\_mag\_data\_feed. Then data from stg\_mag\_data\_feed is inserted into stg\_mag\_header, stg\_mag\_award\_detail, stg\_mag\_vendor, stg\_mag\_commodity with the records having record\_identifier as H, W, V, C respectively using the SQL condition generated using ref\_column\_mapping. Also data is inserted into the respective archive tables with the additional information on job identifier, load identifier and file identifier.

#### Validate data

Function: validatedata

Input parameters: file identifier as in etl\_data\_load\_file

Return value: 1 for successful staging and 0 for any errors.

Data in the staging table is validated for different conditions each of which is mentioned in ref\_validation\_rules in the order of column by name order. If it is a bad/invalid record for the mentioned reason, the invalid flag is set to Y and the invalid reason is populated. Finally invalid records are copied to the corresponding invalid table and deleted from the staging table.

**Note:** Not all validations which are followed in NYC Checkbook1.0 are enforced. For instance all agencies, departments are considered in contrast to considering those which are associated with disbursements/ payroll records, not all data elements considered as mandatory in book1.0 are mandated in NYC Checkbook2.0 (e.g. discount amount has to be provided in NYC Checkbook1.0 however this field is not mandated in NYC Checkbook2.0 as these amounts are not being used in the application)

Details on how each kind of validation is done are listed below.

1. Missing key data elements – rule\_name is “Missing key elements”. Record is validated if any of the mentioned columns is not provided. As NULL and empty value should be treated the same way in this case and due to the different way of handling this check for data types like integer, varchar, date the SQL condition is mentioned in the column invalid\_condition.

For instance to check for key data elements for a disbursement record, corresponding rule in ref\_validation\_rule will have the following values

Staging\_column\_name - doc\_cd, doc\_dept\_cd, doc\_id, doc\_vers\_no

Invalid\_condition – COALESCE (doc\_cd,’’) =’’ OR COALESCE (doc\_dept\_cd,’’) =’’ OR COALESCE (doc\_id,’’) =’’ OR COALESCE (doc\_vers\_no, 0) =0

1. Duplicate records for the unique data elements – Rule name is “Duplicate”. Comma separated column names corresponding to unique data elements is provided in staging\_column\_name. Duplicate records for the given combination of columns are identified and the record with the least record identifier is retained and the rest are invalidated.
2. For instance to check for duplicate disbursement records, corresponding rule in ref\_validation\_rule will have the following values

Staging\_column\_name - doc\_cd, doc\_dept\_cd, doc\_id, doc\_vers\_no

Using this and ref\_data\_source table to get the staging\_table\_name below SQL query will be built

SELECT *doc\_cd, doc\_dept\_cd, doc\_id, doc\_vers\_no*, count(\*), min(record\_id)

FROM *stg\_fms\_header*

GROUP BY *doc\_cd, doc\_dept\_cd, doc\_id, doc\_vers\_no*

HAVING count(\*) > 1

1. Missing parent/component – Rule names starts with Missing followed by the parent/ component name. Parent/Component table name is provided in component\_table\_name and the condition to be verified is provided in invalid\_condition

For instance to check if disbursement has line items associated to it, corresponding rule in ref\_validation\_rule will have the following values

Component\_table\_name – stg\_fms\_accounting\_line

Invalid condition – stg\_fms\_header.doc\_id || ‘~’ stg\_fms\_header.doc\_dept\_cd || ‘~’ stg\_fms\_header.doc\_id || ‘~’ stg\_fms\_header.doc\_vers\_no NOT IN (SELECT stg\_fms\_accounting\_line.doc\_cd || ‘~’ stg\_fms\_accounting\_line.doc\_dept\_cd || ‘~’ stg\_fms\_accounting\_line.doc\_id || ‘~’ stg\_fms\_accounting\_line.doc\_vers\_no

FROM stg\_fms\_accounting\_line)

1. Invalid/Inconsistent values – Rule name starts with Invalid followed by an attribute name. If the value in the data field is to be validated against a lookup table, then name of the column in the staging table is provided against the staging\_column\_name and the lookup table name is provided against ref\_table\_name. Condition to validate is provided in invalid\_condition field. If a specific condition is to be checked then ref\_table\_name will not be provided.
2. Multiple records – Rule name is “Multiple records”. This is similar to “Duplicate records” however the difference is, Multiple records is the existence of more than 1 record for the given parent record. Comma separated column names corresponding to unique data elements of the parent is provided in staging\_column\_name. Records matching these criteria are all invalidated.
3. Inter-load duplicates – Rule name is “Inter-load duplicates”. This is to check if a record with the same set of unique data elements is already loaded in the transaction table. If it is then the record in the staging table is invalidated. Name of the transaction table is provided in transaction\_table\_name and the SQL condition is provided in invalid\_condition. Lookup table information is provided in ref\_table\_name if any lookup tables are required to connect the staging & transaction tables.
4. Associated to invalid components – Starts with “Invalid” and followed by the component name. Name of the component table is provided in component\_table\_name and the SQL condition is mentioned in invalid\_condition. These validations are done only after completing the validations of other types.

**Validations for each of the feeds are provided in appendix A.**

#### Process staged data

Function: processdata

Input parameters: file identifier as in etl\_data\_load\_file

Return value: 1 for successful staging and 0 for any errors.

Actual processing of the staged data is performed in this step and it is specific for each of the data source. Processing is done in the staging tables and once completed the records are posted in the transaction tables. Separate function is created for each of the data source which will be executed from within the main function namely processdata. Core processing done in each of the function is provided below.

##### Process agency

1. Agency\_id which is an additional column in the staging table stg\_agency is updated if a corresponding record exists in ref\_agency identified by agency\_code, agency\_name and action\_flag is set to U for update. For new agency records, agency\_id are generated using the corresponding sequence in the public schema and the action\_flag is set to I for insert.
2. Latest\_flag in ref\_agency is set to N for existing agency records for which name have been changed.
3. New records in the staging table identified by action\_flag as “I” are inserted into the transaction table.

##### Process department

1. Agency\_id column is updated from the ref\_agency table according to the agency\_code.
2. Fund\_class\_id is updated from the ref\_fund\_class table according to the fund\_class\_code.
3. For records where agency\_id/fund\_class\_id cannot be obtained, action\_flag is set to E for error record, invalid\_flag is set to Y and invalid\_reason is set to “Missing FK Values”. These records will not be loaded to the transaction table.
4. Department\_id which is an additional column in the staging table stg\_department is updated if a corresponding record exists in ref\_department identified by agency\_id, fund\_class\_id, fiscal\_year, department\_code, department\_name and action\_flag is set to U for update. For the rest of the records (excluding error records) department\_id is generated using the corresponding sequence in the public schema and the action\_flag is set to I for insert.
5. Latest\_flag in ref\_department is set to N for existing department records for which name have been changed.
6. New records in the staging table identified by action\_flag as “I” are inserted into the transaction table.

##### Process expenditure object

1. Expenditure\_object\_id which is an additional column in the staging table stg\_expenditure\_object\_id is updated if a corresponding record exists in ref\_expenditure\_object identified by expenditure\_object\_code, fiscal\_year, expenditure\_object\_name and action\_flag is set to U for update. For new expenditure object records, expenditure\_object\_id are generated using the corresponding sequence in the public schema and the action\_flag is set to I for insert.
2. Latest\_flag in ref\_expenditure\_object is set to N for existing expenditure object records for which name have been changed.
3. New records in the staging table identified by action\_flag as “I” are inserted into the transaction table.

##### Process location

1. Agency\_id column is updated from the ref\_agency table according to the agency\_code.
2. For records where agency\_id cannot be obtained, action\_flag is set to E for error record, invalid\_flag is set to Y and invalid\_reason is set to “Missing FK Values”. These records will not be loaded to the transaction table.
3. Location\_id which is an additional column in the staging table stg\_location is updated if a corresponding record exists in ref\_ location identified by agency\_id, location\_code, location\_name and action\_flag is set to U for update. For the rest of the records (excluding error records) location\_id is generated using the corresponding sequence in the public schema and the action\_flag is set to I for insert.
4. Latest\_flag in ref\_location is set to N for existing expenditure object records for which name have been changed.
5. New records in the staging table identified by action\_flag as “I” are inserted into the transaction table.

##### Process object Class

1. Object\_class\_id which is an additional column in the staging table stg\_object\_class is updated if a corresponding record exists in ref\_object\_class identified by object\_class\_code, object\_class\_name and action\_flag is set to U for update. For new object class records, object\_class\_id are generated using the corresponding sequence in the public schema and the action\_flag is set to I for insert.
2. Latest\_flag in ref\_location is set to N for existing location records for which name have been changed.
3. New records in the staging table identified by action\_flag as “I” are inserted into the transaction table.

##### Process FMSV

1. Address\_id is updated from address based on address\_line\_1, address\_line\_2, city, state, zip and country. For new addresses address\_id is generated and these details are captured in a different temporary table so that it can be added to the transaction table namely address.
2. For miscellaneous vendors identified by vendor\_customer\_code in ref\_miscellaneous\_vendor, unique vendor\_sub\_code is generated.
3. Vendor\_id is updated from vendor table based on vendor\_customer\_code, legal\_name, vendor\_sub\_code and action flag is set to U for update. For the rest of the records vendor\_id is generated using the corresponding sequence in the public schema and the action flag is set to I for insert.
4. Vendor\_id in the component tables namely stg\_fmsv\_business\_type, stg\_fmsv\_address, stg\_fmsv\_address\_type is updated based on vend\_cust\_cd in stg\_fmsv\_vendor table.
5. Business\_type\_id in stg\_fmsv\_business\_type is updated from ref\_business\_type based on business\_type\_code.
6. Address\_type\_id in stg\_fmsv\_address\_type is updated from ref\_address\_type based on address\_type\_code.
7. New address records are added to the address table.
8. Latest flag in vendor is updated for the vendors which are present in the staging table.
9. New vendors identified by the action flag as I will be inserted into the vendor table with the latest flag as 1. Corresponding business type records & address are added to vendor\_business & vendor\_address tables.
10. For vendors to be updated, alias name and miscellaneous flag are updated. Existing business type and vendor address records in the staging table are deleted and the new set is inserted.
11. It is not clear if identification of miscellaneous vendor should be based on the reference table as in checkbook1.0 or strictly by the miscellaneous flag in FMSV extract.

**Note:** Legal name for a vendor in NYC Checkbook1.0 was strictly from the vendor records associated with the disbursements. However the name as in FMSV extract will be considered for NYC Checkbook2.0.

##### Process MAG

1. FK fields such as document\_code\_id, agency\_id, award\_status\_id, document\_function\_code\_id are updated in etl.stg\_mag\_header from the corresponding reference tables for the columns namely doc\_cd, doc\_dept\_cd, cntrc\_sta, doc\_func\_cd. If agency\_id/department\_id are not available then the corresponding codes are added to the transaction tables.
2. FK fields such as award\_method\_id, award\_level\_id, agreement\_type\_id, award\_category\_id\_1m award\_category\_id\_2, award\_category\_id\_3, award\_category\_id\_4, award\_category\_id\_5 are updated in etl.stg\_mag\_award\_detail from the corresponding reference tables for the columns namely awd\_meth\_cd, awd\_lvl\_cd, cttyp\_cd, ctcat\_cd\_1, ctcat\_cd\_2, ctcat\_cd\_3, ctcat\_cd\_4 and ctcat\_cd\_5. If the lookup tables do not have the values i.e. when foreign key value cannot be obtained default value say 999 is considered. (999 in award category will have the description as ‘no matching category’)
3. FK fields such as vendor\_id are updated in etl.stg\_mag\_vendor from vendor based on vendor customer code.
4. FK fields such as commodity\_type\_id are updated in etl.stg\_mag\_commodity from ref\_commodity\_type.
5. Master agreement id is updated in etl.stg\_mag\_header from history\_all\_master\_agreement based on document\_code\_id, agency\_id, document\_id, document\_version and the action flag is set to U for update. For the rest of the master agreements, master\_agreement\_id is generated using the sequence in the public schema and the action flag is set to I. Similarly replacing master agreement id and replaced by master agreement id are updated.
6. Latest\_flag is set to N (default is Y) if the document in the staging table is not the latest version of the document by checking in all\_master\_agreement.
7. For records to be updated identified by action flag as U following is done

* If latest\_flag is Y, all\_master\_agreement is updated. Existing set of records in all\_agreement\_worksite and all\_agreement\_commodity corresponding to the master\_agreement is deleted and the new set is inserted. Same is done in master\_agreement, agreement\_worksite and agreement\_commodity and the related history tables.
* If latest\_flag is N, above operation are done with the related history tables.

1. For records to be inserted identified by action flag as I following is done

* If latest\_flag is Y, earlier version of the document is deleted from all\_master\_agreement, all\_agreement\_worksite, all\_agreement\_commodity, master\_agreement, agreement\_worksite, agreement\_commodity. Records from the staging table are then inserted into the respective transaction tables including the history tables. Mapping between the master agreement identifiers corresponding to latest version and previous version will be maintained prior to deletion in order to update the disbursement\_line\_item table with the latest agreement identifier.
* If latest\_flag is N, records from the staging table are inserted into the history tables.

**Note:**At this point of time master agreements associated with the disbursements which cannot be displayed or can only be partially displayed will also be available in master\_agreements. They will be deleted only when the corresponding disbursements are loaded. Alternate option is not to process the MAG data if FMS data is not available.

##### Process CON

1. FK fields such as document\_code\_id, agency\_id, award\_status\_id, document\_function\_code\_id are updated in the header tables namely stg\_con\_ct\_header, stg\_con\_po\_header, stg\_con\_do1\_header from the corresponding reference tables for the columns namely doc\_cd, doc\_dept\_cd, cntrc\_sta, doc\_func\_cd.
2. FK fields such as award\_method\_id, award\_level\_id, agreement\_type\_id, award\_category\_id\_1m award\_category\_id\_2, award\_category\_id\_3, award\_category\_id\_4, award\_category\_id\_5 are updated in stg\_con\_ct\_award\_detail & stg\_con\_po\_award\_detail from the corresponding reference tables for the columns namely awd\_meth\_cd, awd\_lvl\_cd, cttyp\_cd, ctcat\_cd\_1, ctcat\_cd\_2, ctcat\_cd\_3, ctcat\_cd\_4 and ctcat\_cd\_5.
3. FK fields such as vendor\_id are updated in stg\_con\_ct\_vendor, stg\_con\_po\_vendor, stg\_con\_do1\_vendor from vendor based on vendor customer code.
4. FK fields such as commodity\_type\_id are updated in stg\_con\_ct\_commodity, stg\_con\_po\_commodity, stg\_con\_do1\_commodity from ref\_commodity\_type.
5. FK fields such as fund\_class\_id, agency\_id, department\_id, expenditure\_object\_id, revenue\_source\_id, budget\_code\_id are updated in stg\_con\_ct\_accounting\_line, stg\_con\_po\_accounting\_line, stg\_con\_do1\_accounting\_line from the corresponding reference tables.
6. Master agreement id is updated in stg\_con\_ct\_header, stg\_con\_do1\_header from history\_all\_master\_agreement based on agree\_doc\_cd, agree\_doc\_dept\_cd, agree\_doc\_id (latest version to be considered).
7. Agreement id is updated in stg\_con\_ct\_header, stg\_con\_po\_header, stg\_con\_do1\_header from history\_all\_agreement based on document\_code\_id, agency\_id, document\_id, document\_version and the action flag is set to U for update. For the rest of the agreements, agreement\_id is generated using the sequence in the public schema and the action flag is set to I. Similarly replacing agreement id and replaced by agreement id are updated.
8. Latest\_flag is set to N (default is Y) if the document in the staging table is not the latest version of the document by checking in all\_agreement.
9. For records to be updated identified by action flag as U following is done

* If latest\_flag is Y, all\_agreement is updated. Existing set of records in all\_agreement\_worksite, all\_agreement\_commodity and all\_agreement\_accounting\_line corresponding to the agreement is deleted and the new set is inserted. Same is done in agreement, agreement\_worksite, agreement\_commodity, agreement\_accounting\_line and the related history tables.
* If latest\_flag is N, above operations are done with the related history tables.

1. For records to be inserted identified by action flag as I following is done

* If latest\_flag is Y, earlier version of the document is deleted from all\_agreement, all\_agreement\_worksite, all\_agreement\_commodity, all\_agreement\_accounting\_line, agreement, agreement\_worksite, agreement\_commodity, agreement\_accounting\_line. Records from the staging table are then inserted into the respective transaction tables including the history tables. Mapping between the agreement identifiers corresponding to latest version and previous version will be maintained prior to deletion in order to update the disbursement\_line\_item table with the latest agreement identifier.
* If latest\_flag is N, records from the staging table are inserted into the history tables.

**Note:** At this point of time agreements associated with the disbursements which cannot be displayed or can be partially displayed only will also be available in agreements. They will be deleted only when the corresponding disbursements are loaded. Alternate option is not to process the CON data if FMS data is not available.

##### Process FMS

1. FK fields such as document\_code\_id, agency\_id are updated in stg\_fms\_header from the corresponding reference tables for the columns namely doc\_cd and doc\_dept\_cd.
2. Display\_type column is set to N/P/F for not to be displayed, displayed partially and displayed in full based on the file name.
3. FK field such as vendor\_id is updated in stg\_fms\_vendor from vendor based on vendor customer code. For vendors associated with header records where the display type is N/P and doc\_cd is DC, public\_vendor\_id is set to a masked vendor with the vendor customer code as N/A and vendor legal name as N/A (PRIVACY/SECURITY). Rest of the vendors will have the same vendor id in vendor\_id and public\_vendor\_id. For miscellaneous vendor, unique vendor sub code is generated and the legal name from the vendor record of this extract will be considered and posted to the transaction table. If vendor id cannot be obtained from the vendor transaction table a record is created in the vendor table with the customer code and legal name information as in the vendor record of the disbursement.
4. FK fields such as fund\_class\_id, agency\_id, department\_id, expenditure\_object\_id, revenue\_source\_id, budget\_code\_id, fund\_id are updated in stg\_fms\_accounting\_line from the corresponding reference tables.
5. Disbursement status (expenditure\_status\_id) is set to cancelled for disbursements with version number greater than 1.
6. If dept\_cd is 096 in stg\_fms\_accounting\_line, public\_agency\_id is updated to the one corresponding to 069.
7. If dept\_cd is 098 in stg\_fms\_accounting\_line and associated to an agreement with the department code as 015 i.e. rqporf\_doc\_dept\_cd as 015, public\_agency\_id is updated to the one corresponding to 015.
8. If dept\_cd is 098 and obj\_cd is 4000/ 4140/ 6000/ 6130/ 6150/ 6220/ 6650/ 6710/ 6780/ 6810/ 6820/ 6830/ 6860, public\_agency\_id will be set to the one corresponding to rqporf\_doc\_dept\_cd.
9. Agreement id is updated in stg\_fms\_accounting\_line from history\_all\_master\_agreement based on rqporf\_doc\_id, rqporf\_doc\_cd, rqporf\_doc\_dept\_cd (latest version to be considered) if disbursement is associated to MAG. Public\_agreement\_id is set to a masked agreement with the document code as N/A, agency as N/A and document id as (PRIVACY/SECURITY).
10. Agreement id is updated in stg\_fms\_accounting\_line from history\_all\_agreement based on rqporf\_doc\_id, rqporf\_doc\_cd, rqporf\_doc\_dept\_cd (latest version to be considered) if disbursement is associated to CON.
11. If agreement id cannot be obtained (when MAG/CON data is not available in the transaction table), a record is created in all\_master\_agreement, master\_agreement, all\_agreement, agreement depending on the document type with the key information on document type, agency code and document id. Once the actual data is available from MAG/CON this forged record will be deleted.
12. All disbursements are considered as new records. Records from the staging tables are inserted into all\_disbursement, all\_disbursement\_line\_item. Same is done for disbursement and disbursement\_line\_item but public\_agency\_id, public\_vendor\_id and public\_agreement\_id will be used instead of agency\_id, vendor\_id and agreement\_id. Also the line item amount is negated for cancelled disbursements when inserting into disbursement\_line\_item.
13. Master agreements which are associated to disbursements related to document type DC and which cannot be displayed or partially displayed are deleted from master\_agreement, agreement\_worksite, agreement\_commodity and its history tables.
14. Agreements which are associated to disbursements related to document type DC and which cannot be displayed or partially displayed are deleted from agreement, agreement\_worksite, agreement\_commodity, agreement\_accounting\_line and its history tables.
15. Vendors which are associated to disbursements related to document type DC and which cannot be displayed or partially displayed are deleted from vendor, vendor\_business\_type, vendor\_address

##### Process Budget

1. FK fields such as fund\_class\_id, agency\_id, department\_id, budget\_code\_id, object\_class\_id are updated in the staging table namely stg\_budget.
2. Total\_expenditure\_amount is computed as the sum of pre\_encumbered\_amount, encumbered\_amount, accrued\_expense\_amount, cash\_expense\_amount & post\_closing\_adjustment\_amount.
3. Budget\_id is updated from budget based on budget\_fiscal\_year, fund\_class\_id, agency\_id, department\_id, budget\_code\_id, object\_class\_id and action flag is set to U for update. For the rest of the budget transactions, budget\_id is generated using the sequence in the public schema and the action flag is set to I.
4. Records identified as new records are inserted into the budget table and the modified records identified by action flag as U are updated.

##### Process Revenue

1. FK fields such as agency\_id, department\_id, object\_class\_id, funding\_source\_id, revenue\_category\_id, revenue\_class\_id, revenue\_source\_id are updated in the staging table namely stg\_revenue.
2. Revenue\_id is generated using the sequence in the public schema
3. All transactions are considered as new and are inserted into the transaction table namely revenue.

## Questions

1. What is the significance of miscellaneous flag from FMSV as disbursements associated with a vendor identified as miscellaneous (by checking against a standard list) are categorized as different vendors with the same vendor customer code but with different vendor sub code.
2. How the data files with some errors are being corrected? For instance if a FMSV daily file for July, 28th has some errors and it had to be recreated, will the data be made available in FTP server with the same name or as a different file with different timestamp.
3. How to handle master agreements/ agreements which are being replaced in order to avoid duplicate master agreements/agreements?

## Appendix A

#### COA data validations

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COA** | | | | | | | | | | | | |
| **Type of feed** | | Daily – Full | | | | | | | | | | |
| **File Name Pattern** | | AID2\_DLY\_COA\_DEPT\_20100601120000.asc  AID2\_DLY\_COA\_APPR\_20100601120000.asc  AID2\_DLY\_COA\_OBJ\_20100601120000.asc  AID2\_DLY\_COA\_LOC\_20100601120000.asc | | | | | | | | | | |
| **Number of files to be processed** | | One for each category. | | | | | | | | | | |
| **Tables** | | | | | | | | | | | | |
| **Entity Name** | **Entity Identifier** | | **Unique elements** | **Staging Table** | | **Transaction Table** | | **History Table** | | | | |
| Agency | NA | | agency\_code | ETL.stg\_agency | | ref\_agency | | NA | | | | |
| Department | NA | | agency\_code  fund\_class\_code  department\_code  fiscal\_year | ETL.stg\_department | | Ref\_department | | NA | | | | |
| Expenditure Object | NA | | expenditure\_object\_code  fiscal\_year | ETL.stg\_expenditure\_object | | Ref\_expenditure\_object | | NA | | | | |
| Location | NA | | location\_code  agency\_code | ETL.stg\_location | | Ref\_location | | NA | | | | |
| Object class | NA | | object\_class\_code | ETL.stg\_object\_class | | Ref\_object\_class | | NA | | | | |
| **Validation Rules** | | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | | **Duplicate** | **Invalid/Inconsistent Values** | **Missing Referenced Parent Entity** | | **Associated to invalid entities** | | | **Multiple Records** | |
| Agency | 1.agency\_code  2.agency\_name | | agency\_code |  |  | |  | | |  | |
| Department | 1.agency\_code  2.fund\_class\_code  3.department\_code  4.fiscal\_year | | agency\_code  fund\_class\_code  department\_code  fiscal\_year |  |  | |  | | |  | |
| Expenditure Object | 1.expenditure\_object\_code  2.fiscal\_year  3.expenditure\_object\_name | | expenditure\_object\_code  fiscal\_year |  |  | |  | | |  | |
| Location | 1.location\_code  2.location\_name  3.agency\_code | | location\_code  agency\_code |  |  | |  | |  | |
| Object class | 1.object\_class\_code  2.object\_class\_name | | object\_class\_code |  |  | |  | |  | |

#### FMSV data validations

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FMSV** | | | | | | | | | | |
| **Type of feed** | | Monthly – Full  Weekly  -  Incremental  Daily – Incremental | | | | | | | | |
| **File Name Pattern** | | AIV0\_DLY\_VEND\_CCYYMMDDHHMMSS.asc (Daily) AIV1\_WKLY\_VEND\_CCYYMMDDHHMMSS.asc (Weekly) AIV2\_MTHLY\_VEND\_CCYYMMDDHHMMSS.asc (Monthly) | | | | | | | | |
| **Number of files to be processed** | | Multiple files processed in the order in which it was created in FTP server. | | | | | | | | |
| **Tables** | | | | | | | | | | |
| **Entity Name** | **Entity Identifier** | **Unique elements** | | **Staging Table** | | **Transaction Table** | | **History Table** | | |
| Vendor | V | vend\_cust\_cd | | etl.stg\_fmsv\_vendor | | vendor | | NA | | |
| Vendor Business Type | M | vend\_cust\_cd  bus\_typ | | etl.stg\_fmsv\_business\_type | | vendor\_business\_type | | NA | | |
| Vendor Address | A | vend\_cust\_cd  ad\_id | | etl.stg\_fmsv\_address | | address  vendor\_address | | NA | | |
| Vendor Address Type | T | vend\_cust\_cd  ad\_id  ad\_typ | | etl. stg\_fmsv\_address\_type | | vendor\_address | | NA | | |
| **Validation Rules** | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | **Duplicate** | **Invalid/Inconsistent Values** | | **Missing Referenced Parent Entity** | | **Associated to invalid entities** | | **Multiple Records** | |
| Vendor | 1.vend\_cust\_cd  2.combination of lgl\_nm & alias\_nm | vend\_cust\_cd | 1.EFBGN\_DT is after EFEND\_DT | | NA | | 1.Invalid Business Type records  2.Invalid Address records  3.Invalid Address Type records | |  | |
| Vendor Business Type | 1.vend\_cust\_cd  2.bus\_typ  3.bus\_typ\_sta | vend\_cust\_cd  bus\_typ | 1. Invalid business type ( Not in ref\_business\_type )  2. Invalid business type status( Not in ref\_business\_type\_status)  3.Invalid minority type ( Not in ref\_minority\_type)  4. Minority type is not provided for the minority Business type with the status as accepted.  5. Minority type is provided but the business type is not MNRT | | Business type records without a vendor | | Invalid vendor  Invalid business type  Invalid address  Invalid address type | |  | |
| Vendor Address | 1.vend\_cust\_cd  2.ad\_id  3.ctry  4.str\_1\_nm  5.(ctry = US AND st/ zip/ city\_nm is NULL) | vend\_cust\_cd  ad\_id | EFBGN\_DT is after EFEND\_DT | | Vendor address records without a vendor | | Invalid vendor  Invalid business type  Invalid address  Invalid address type | |  | |
| Vendor Address Type | 1.vend\_cust\_cd  2.ad\_id  3.ad\_typ  4.efbgn\_dt | vend\_cust\_cd  ad\_id  ad\_typ | Invalid address type (Not in ref\_address\_type) | | 1. Vendor address type records without a vendor  2. Vendor address type records without vendor address | | Invalid vendor  Invalid business type  Invalid address  Invalid address type | |  |

#### MAG data validations

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MAG** | | | | | | | | | | | | | |
| **Type of feed** | | Monthly – Incremental  Daily – Incremental | | | | | | | | | | | |
| **File Name Pattern** | | AIDP\_DLY\_PCO\_MA\_CCYYMMDDHHMMSS.asc (Daily) AIMP\_MTHLY\_PCO\_MA\_CCYYMMDDHHMMSS.asc (Monthly) | | | | | | | | | | | |
| **Number of files to be processed** | | Multiple files processed in the order in which it was created in FTP server. | | | | | | | | | | | |
| **Tables** | | | | | | | | | | | | | |
| **Entity Name** | **Entity Identifier** | | **Unique elements** | | **Staging Table** | | **Transaction Table** | | **History Table** | | | | |
| MAG header | H | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | | etl.stg\_mag\_header | | master\_agreement  all\_master\_agreement | | history\_master\_agreement  history\_all\_master\_agreement | | | | |
| MAG award detail | W | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vers\_no  doc\_awddet\_ln\_no | | etl.stg\_mag\_award\_detail | | master\_agreement  all\_master\_agreement  agreement\_worksite  all\_agreement\_worksite | | history\_master\_agreement  history\_all\_master\_agreement  history\_agreement\_worksite  history\_all\_agreement\_worksite | | | | |
| MAG vendor | V | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | | etl.stg\_mag\_vendor | | master\_agreement  all\_master\_agreement | | history\_master\_agreement  history\_all\_master\_agreement | | | | |
| MAG commodity | C | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no  doc\_comm\_ln\_no | | etl.stg\_mag\_commodity | | agreement\_commodity  all\_agreement\_commodity | | history\_agreement\_commodity  history\_all\_agreement\_commodity | | | | |
| **Validation Rules** | | | | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | | **Duplicate** | **Invalid/Inconsistent Values** | | **Missing Parent/Child Entity** | | **Associated to invalid entities** | | **Multiple records** | | **Inter-Load duplicates** | |
| MAG header | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | 1. efbgn\_dt > efend\_dt  2. ma\_prch\_lmt\_am < 0 | | MAG header without award detail | | 1.Invalid MAG award detail  2. Invalid Vendor  3. Invalid commodity | |  | |  | |
| MAG award detail | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5.doc\_awddet\_ln\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_awddet\_ln\_no | 1. Invalid award method (Not in ref\_award\_method)  2. Invalid agreement type (Not in ref\_agreement\_type)  3. Invalid agreement category ( Not in ref\_award\_category) | | MAG award detail records without MAG header | | 1.Invalid MAG header  2. Invalid Vendor  3. Invalid commodity | | Multiple records for a MAG header. | |  | |
| MAG vendor | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5. doc\_vend\_ln\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no |  | | Vendor without a header | | 1.Invalid MAG header  2. Invalid award detail  3. Invalid commodity | | Multiple records for a MAG header. | |  | |
| MAG commodity | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5.doc\_vend\_ln\_no  6.doc\_comm\_ln\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no  doc\_comm\_ln\_no |  | | Commodity without a header | | 1.Invalid MAG header  2. Invalid award detail  3. Invalid vendor  4. Invalid commodity | |  |  | |

#### CON data validations

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CON** | | | | | | | | | | | | | |
| **Type of feed** | | Monthly – Incremental  Daily - Incremental | | | | | | | | | | | |
| **File Name Pattern** | | AIDP\_DLY\_PCO\_PO\_CCYYMMDDHHMMSS.asc (Daily) AIMP\_MTHLY\_PCO\_PO\_CCYYMMDDHHMMSS.asc (Monthly) | | | | | | | | | | | |
| **Number of files to be processed** | | Multiple files processed in the order in which it was created in FTP server. | | | | | | | | | | | |
| **Tables** | | | | | | | | | | | | | |
| **Entity Name** | **Document Type** | | **Entity Identifier** | **Unique elements** | | **Staging Table** | | **Transaction Table** | | **History Table** | | | |
| CON header |  | | H | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | | etl.stg\_con\_ct\_header(CT1, CTA1, CTA2)  etl.stg\_con\_po\_header (POC, PCC1, POD)  etl.stg\_con\_do1\_header (DO1) | | agreement  all\_agreement | | history\_agreement  history\_all\_agreement | | | |
| CON Award detail | All except DO1 | | W | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  awd\_meth\_cd | | etl.stg\_con\_ct\_award\_detail(CT1, CTA1, CTA2)  etl.stg\_con\_po\_award\_detail(POC, PCC1, POD) | | agreement  all\_agreement agreement\_worksite  all\_agreement\_worksite | | agreement  all\_agreement agreement\_worksite  all\_agreement\_worksite | | | |
| CON Vendor | All | | V | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no | | etl.stg\_con\_ct\_vendor (CT1, CTA1, CTA2)  etl.stg\_con\_po\_vendor (POC, PCC1, POD)  etl.stg\_con\_do1\_vendor (DO1) | | agreement  all\_agreement | | history\_agreement  history\_all\_agreement | | | |
| CON Commodity | All | | C | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no  doc\_vend\_ln\_no  doc\_comm\_ln\_no | | etl.stg\_con\_ct\_commodity(CT1, CTA1, CTA2)  etl.stg\_con\_po\_commodity(POC, PCC1, POD)  etl.stg\_con\_do1\_commodity(DO1) | | agreement\_commodity  all\_agreement\_commodity | | history\_agreement\_commodity  history\_all\_agreement\_commodity | | | |
| CON accounting line | All | | A | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no  doc\_vend\_ln\_no  doc\_comm\_ln\_no  doc\_actg\_ln\_no | | etl.stg\_con\_ct\_accounting\_line(CT1, CTA1, CTA2)  etl.stg\_con\_po\_accounting\_line(POC, PCC1, POD) etl.stg\_con\_do1\_accounting\_line(DO1) | | agreement\_accounting\_line  all\_agreement\_accounting\_line | | history\_agreement\_accounting\_line  history\_all\_agreement\_accounting\_line | | | |
| **Validation Rules** | | | | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | | **Duplicate** | | **Invalid/Inconsistent Values** | | **Missing Parent/child Entity** | | **Associated to invalid entities** | **Multiple records** | | **Inter-Load duplicates** | |
| Header | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | | 1.cntrct\_strt\_dt > cntrct\_end\_dt  2. max\_cntrc\_am < 0 | | Header without an award detail record (for document types other than DO1) | | 1. Invalid award detail record  2. Invalid vendor  3. Invalid commodity  4.Invalid accounting line |  | |  | |
| Award detail  (All except DO1) | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5.awd\_meth\_cd | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  awd\_meth\_cd | | 1.Invalid awd\_meth\_cd (Not in ref\_award\_method)  2.Invalid cttyp\_cd ( Not in ref\_agreement\_type)  3. Invalid ctcat\_cd\_1 (ref\_award\_category). Same for the other 4 award categories. | | Award detail records with a header | | 1.Invalid header  2.Invalid vendor  3.Invalid commodity  4.Invalid accounting line | Multiple award detail records for a header. | |  | |
| Vendor | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5. doc\_vend\_ln\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | |  | | Vendor record without a header | | 1.Invalid header  2.Invalid award detail  3.Invalid commodity  4.Invalid accounting line | Multiple vendor records for a header. | |  | |
| Commodity | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5.doc\_vend\_ln\_no  6.doc\_comm\_ln\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no  doc\_comm\_ln\_no | |  | | commodity record without a header | | 1.Invalid header  2.Invalid award detail  3.Invalid vendor  4.Invalid commodity  5.Invalid accounting line |  |  | |
| Accounting line | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5.doc\_vend\_ln\_no  6.doc\_comm\_ln\_no  7. doc\_actg\_ln\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no  doc\_comm\_ln\_no  doc\_actg\_ln\_no | |  | | Accounting line without a header | | 1.Invalid header  2.Invalid award detail  3.Invalid vendor  4.Invalid commodity  5.Invalid accounting line |  |  | |

#### FMS data validations

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FMS** | | | | | | | | | | | |
| **Type of feed** | | Daily – Incremental | | | | | | | | | |
| **File Name Pattern** | | AIDM\_DLY\_MMDSBP\_DC\_20100601120000.asc (Partially displayed)  AIDM\_DLY\_MMDSBX\_DC\_20100601120000.asc (Not displayed)  AIDM\_DLY\_MMDSBP\_DC\_20100601120000.asc (displayed fully) | | | | | | | | | |
| **Number of files to be processed** | | Multiple files processed in the order in which it was created in FTP server. | | | | | | | | | |
| **Tables** | | | | | | | | | | | |
| **Entity Name** | **Entity Identifier** | | **Unique elements** | **Staging Table** | | **Transaction Table** | | **History Table** | | | |
| Disbursement | H | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | etl.stg\_fms\_header | | expenditure  all\_expenditure | | NA | | | |
| Vendor | V | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no | etl.stg\_fms\_vendor | | expenditure  all\_expenditure | | NA | | | |
| Disbursement Accounting line | A | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_vend\_ln\_no  doc\_actg\_ln\_no | etl.stg\_fms\_accounting\_line | | expenditure\_line\_item  all\_expenditure\_line\_item | | NA | | | |
| **Validation Rules** | | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | | **Duplicate** | | **Invalid/Inconsistent Values** | | **Missing Parent/Child Entity** | | **Associated to invalid entities** | **Multiple records** | **Inter-Load duplicates** |
| Disbursement | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5.doc\_rec\_dt\_dc  6. doc\_bfy  7. doc\_fy\_dc  8. doc\_per\_dc  9. chk\_eft\_am  10.chk\_eft\_iss\_dt | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | |  | | 1.Disbursement without vendor  2. Disbursement without line items | | 1.Invalid vendor  2. Invalid disbursement line item |  | Invalidate the disbursement if there exists a disbursement in the transaction table for the combination of given unique elements. |
| Vendor | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.vend\_cust\_cd | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no | |  | |  | |  | Multiple vendors for a disbursement. |  |
| Disbursement Accounting line | 1.doc\_cd  2.doc\_dept\_cd  3.doc\_id  4.doc\_vers\_no  5.doc\_vend\_ln\_no  6.doc\_actg\_ln\_no  7. dept\_cd  8. bfy  9. fy\_dc  10. per\_dc  11. chk\_amt  12.rfed\_doc\_cd  13.rfed\_doc\_dept\_cd  14.rfed\_doc\_id  15.rfed\_actg\_ln\_no | | doc\_cd  doc\_dept\_cd  doc\_id  doc\_vers\_no  doc\_actg\_ln\_no | |  | | Disbursement line item without disbursement | | 1.Invalid disbursement  2.Invalid disbursement line item |  |  |

#### PMS data validations

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PMS** | | | | | | | | | | |
| **Type of feed** | | Daily – Incremental | | | | | | | | |
| **File Name Pattern** | | PAYROLL\_A015\_XGDVI680\_20100601120000.ASC | | | | | | | | |
| **Number of files to be processed** | | Multiple | | | | | | | | |
| **Tables** | | | | | | | | | | |
| **Entity Name** | **Entity Identifier** | | **Unique elements** | **Staging Table** | | **Transaction Table** | | **History Table** | | |
| Payroll Summary | NA | | pay\_cycle  pay\_date  pyrl\_no  uoa  fy  object  agency  bud\_code | etl.stg\_pms\_data\_feed | | payroll\_summary | | NA | | |
| **Validation Rules** | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | | **Duplicate** | **Invalid/Inconsistent Values** | **Missing Referenced Parent Entity** | | **Associated to invalid entities** | | **Multiple Records** | **Inter-Load duplicate** |
| Payroll Summary | 1.pay\_cycle  2.pay\_date  3.pyrl\_no  4.uoa  5.fy  6.object  7.agency  8.bud\_code  9. total\_amt | | pay\_cycle  pay\_date  pyrl\_no  uoa  fy  object  agency  bud\_code | Invalid pay\_cycle (Not in ref\_pay\_cycle) |  | |  | |  | Invalidate the record if a payroll summary record was already loaded in the system for the combination of given unique elements. |

#### Budget data validations

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Budget** | | | | | | | | | | |
| **Type of feed** | | Daily – Incremental | | | | | | | | |
| **File Name Pattern** | | AIB1\_DLY\_BUDSTRU92\_MM\_CCYYMMDDHHMISS.asc | | | | | | | | |
| **Number of files to be processed** | | Multiple | | | | | | | | |
| **Tables** | | | | | | | | | | |
| **Entity Name** | **Entity Identifier** | | **Unique elements** | **Staging Table** | | **Transaction Table** | | **History Table** | | |
| Budget transaction | NA | | BFY  Fund Class  Department  Appropriation Unit  Budget Code  Object | Etl.stg\_budget | | Budget | |  | | |
| **Validation Rules** | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | | **Duplicate** | **Invalid/Inconsistent Values** | **Missing Referenced Parent Entity** | | **Associated to invalid entities** | | **Multiple Records** | **Inter-Load duplicate** |
| Budget transaction | 1.BFY  2.Fund Class  3.Department  4.Appropriation Unit  5.Budget Code  6.Object | | BFY  Fund Class  Department  Appropriation Unit  Budget Code  Object |  |  | |  | |  | . |

#### Revenue data validations

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Revenue** | | | | | | | | | | |
| **Type of feed** | | Daily – Incremental  Monthly – Incremental | | | | | | | | |
| **File Name Pattern** | | AIDJ\_DLY\_JL\_JL\_032311015417.ASC  AIDJ\_MTHLY\_JL\_JL\_040111015417.ASC | | | | | | | | |
| **Number of files to be processed** | | Multiple | | | | | | | | |
| **Tables** | | | | | | | | | | |
| **Entity Name** | **Entity Identifier** | | **Unique elements** | **Staging Table** | | **Transaction Table** | | **History Table** | | |
| Revenue transaction | NA | | None | Etl.stg\_revenue | | Revenue | |  | | |
| **Validation Rules** | | | | | | | | | | |
| **Entity Name** | **Missing Key Values** | | **Duplicate** | **Invalid/Inconsistent Values** | **Missing Referenced Parent Entity** | | **Associated to invalid entities** | | **Multiple Records** | **Inter-Load duplicate** |
|  |  | |  |  |  | |  | |  | . |