****

**Summary of Understanding – ETL and Data Processing**

**Prepared for**

**Thursday, June 16, 2011**

**Version .0**

**Author**

**VINAY BEERAVOLU**

Revision and Signoff Sheet

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** |  |
| 06/16/2011 |  | 0.1 |  |
| 06/21/2011 |  | 0.2 |  |
| 06/22/2011 |  | 0.3 |  |

Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Version approved** | **Position** | **Date** |
| Vinay Beeravolu | 0.1 | DB Administrator | 06/16 |
| Athammai Thiagarajan | 0.2 | Principle DBA | 06/21 |
| Kishore Vuppala | 0.3 | Technical Lead | 06/22 |

Contents

[Background 4](#_Toc296547542)

[Scope 4](#_Toc296547543)

[Data Feeds 4](#_Toc296547544)

[ETL 5](#_Toc296547545)

[Overview 5](#_Toc296547546)

[GetIncomingFiles: 7](#_Toc296547547)

[COA Data Feed 8](#_Toc296547548)

[FMSV Data Feed 10](#_Toc296547549)

[CON Data Feed 14](#_Toc296547550)

[MAG Data Feed 17](#_Toc296547551)

[PMS Data Feed 19](#_Toc296547552)

[FMS Data Load 21](#_Toc296547553)

[Questions 25](#_Toc296547554)

## Background

To gain an understanding of the existing ETL (Extract transform and load) process for the data provided by FISA (Financial Information Systems Agency) and to incorporate it as part of Checkbook 2.0 initiative, REI Systems was provided with ETL SSIS packages that process data sets received from FISA. This document details out REI’s understanding based on the discussions, documents and data sets provided by NYC Office and FISA.

This document has feedback from REI regarding the provided documents including understanding, questions, concerns, and request for additional information, where applicable.

## Scope

Scope of the document is to provide an extensive level of detail of the consumption of various data feeds, processing of these and making it available to MyMoney database.

Document is organized to give information on the 6 data feeds, an overall picture of the ETL process followed by the detailed one for each of the 6 data feeds.

## Data Feeds

Six data feeds are provided by FISA to the Comptroller’s office for driving two applications namely Checkbook 1.0 and MWBE (Minority and Women Based Enterprise). Standard used in all feeds include pipe (|) as a field delimiter and carriage return/line feed character sequence as the record delimiter. However PMS data feed is an exception as line feed character is used as record delimiter. Data Feeds are not provided with header information. Header information is provided in the respective data mapping documents.

Details of each feed are provided below.

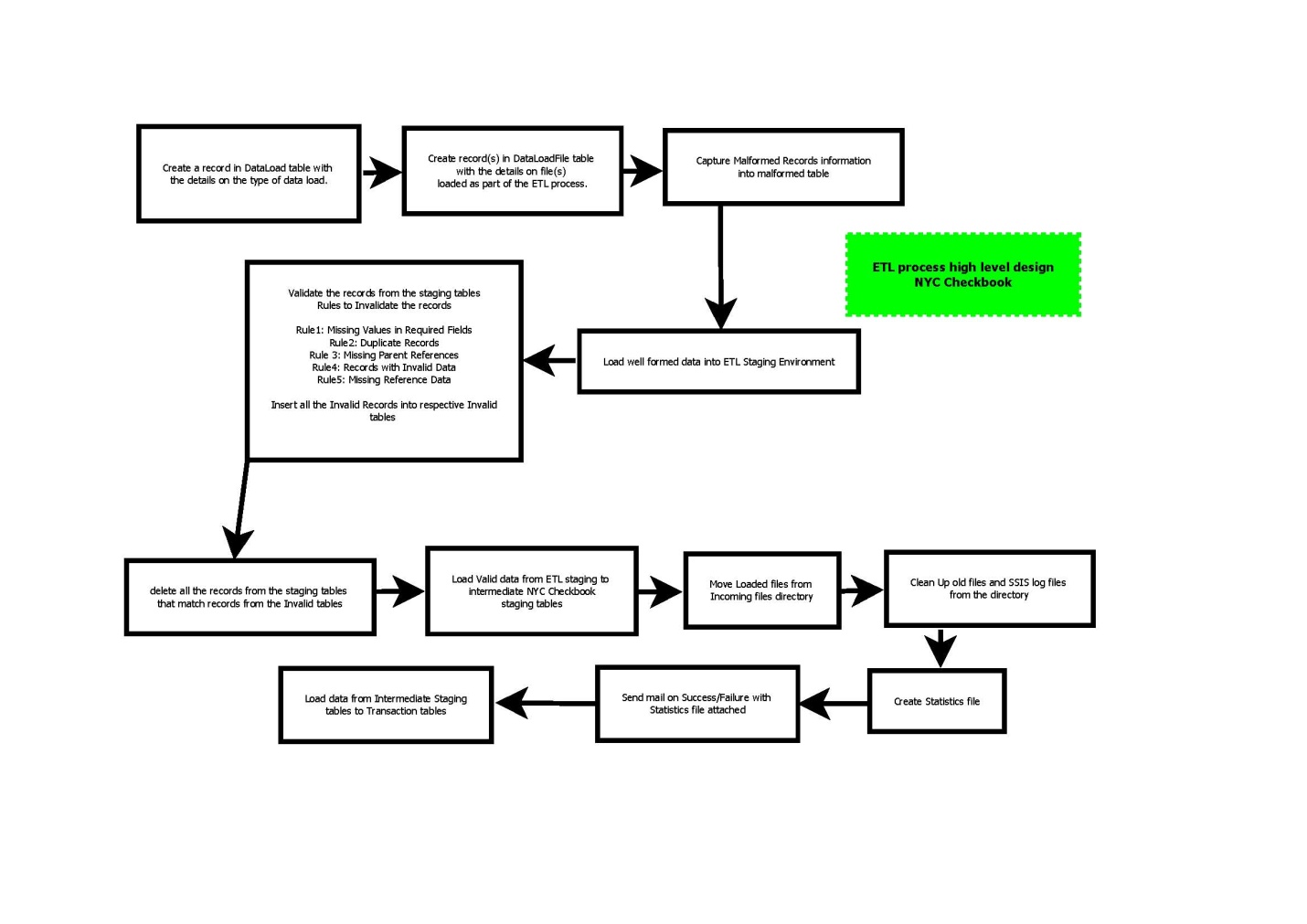
|  |  |  |
| --- | --- | --- |
| **Source Feed** | **File Name Pattern** | **Type of feed** |
| COA | AID[1-4]\_DLY\_COA\_DEPT\_CCYYMMDDHHMMSS.asc (Department). AID[1-4]\_DLY\_COA\_APPR\_CCYYMMDDHHMMSS.asc (Appropriation Unit). AID[1-4]\_DLY\_COA\_LOC\_CCYYMMDDHHMMSS.asc (Location Codes). AID[1-4]\_DLY\_COA\_OBJ\_CCYYMMDDHHMMSS.asc (Expenditure Object Codes). | Daily – Full (Last two year data) |
| FMSV | AIV0\_DLY\_VEND\_CCYYMMDDHHMMSS.asc (Daily) AIV1\_WKLY\_VEND\_CCYYMMDDHHMMSS.asc (Weekly) AIV2\_MTHLY\_VEND\_CCYYMMDDHHMMSS.asc (Monthly) | Daily/Weekly - Incremental  Monthly – Full |
| FMS | AI[DM|PM|XM]\_DLY\_MMDSB[|P|X]\_[AD|DC|MD]\_ CCYYMMDDHHMMSS.asc P: Data to be Partially Displayed X: Data Not to be Displayed Empty: Data to be Fully Displayed | Daily - Incremental |
| PMS | PAYROLL\_A015\_\*\_CCYYMMDDHHMMSS.ASC | Daily – Incremental |
| CON | AIDP\_DLY\_PCO\_PO\_CCYYMMDDHHMMSS.asc (Daily) AIMP\_MTHLY\_PCO\_PO\_CCYYMMDDHHMMSS.asc (Monthly) | Daily/Monthly – Incremental |
| MAG | AIDP\_DLY\_PCO\_MA\_CCYYMMDDHHMMSS.asc (Daily) AIMP\_MTHLY\_PCO\_MA\_CCYYMMDDHHMMSS.asc (Monthly) | Daily/Monthly – Incremental |

## ETL Overview

NYC Comptroller’s office uses Microsoft SQL Server Integration services (SSIS) to process the data files received from FISA. Controller package is executed with a SQL Server Agent job, scheduled to run every 30 minutes between 4:15 and 6:45 AM to load the data from FISA. This is a master package which executes the below packages in the mentioned order.

|  |  |  |
| --- | --- | --- |
| **Package name** | **Purpose** | **Prerequisite for the package execution** |
| *GetIncomingFiles* | FTP all eligible files for all the 6 data sources. | NA |
| *LoadCOADataLoad* | Load COA data for Department, Appropriation Unit, Expenditure Object and Location | NA |
| *LoadFMSVDataLoads* | Load Vendors | NA |
| *LoadCONDataLoads* | Load contracts which includes CT1, CTA1, CTA2, POC, PCC1, POD,DO1 | NA |
| *LoadMAGDataLoads* | Load master agreements | NA |
| *LoadFMSDataLoad* | Load Disbursements | *LoadCOADataLoad, LoadFMSVDataLoads, LoadCONDataLoads, LoadMAGDataLoads* |
| *LoadPMSDataLoad* | Load payroll summary | *LoadCOADataLoad,* |
| *PublishFMSDataLoadsM* | Publish Disbursements Data from MyMoney to MyMoneyPublic database on Rackspace for Checkbook 1.0 (Not discussed in detail in this document). | *LoadCOADataLoad, LoadFMSVDataLoads, LoadCONDataLoads, LoadMAGDataLoads, LoadFMSDataLoad,* |
| *PublishPMSDataLoadsM* | Publish Payroll Data from MyMoney to MyMoneyPublic database on Rackspace for Checkbook 1.0 (Not discussed in detail in this document). | *LoadCOADataLoad,  LoadPMSDataLoad* |
| *PublishFMSDataLoadsR* | Publish Disbursement Data from MyMoney to MyMoneyPublicR database on Rackspace for MWBE (Not discussed in detail in this document). | *LoadCOADataLoad, LoadFMSVDataLoads, LoadCONDataLoads, LoadMAGDataLoads, LoadFMSDataLoad,* |
| *PublishPMSDataLoadsR* | Publish Payroll Data from MyMoney to MyMoneyPublicR database on Rackspace for MWBE (Not discussed in detail in this document). | *LoadCOADataLoad,  LoadPMSDataLoad* |

Most of the execution steps are common across all the packages which are used to load the data. Pictorial representation of this is provided below. Also more detailed information is provided in the respective data feed sections.



### GetIncomingFiles:

The GetIncomingFiles package processes (COA, FMSV, CON, MAG, FMS and PMS) incoming files from FISA FTP Secure Server. A third party tool nsoftware is used for FTP-S to FISA server as SSIS doesn’t have the ability to connect to FTP-S server. The Task implemented to download is FTPReceive task. The files eligible to be downloaded are inserted into *ETL.FTPDirectoryEntry* table with the details on filename and timestamp on the file. If the same file name with the same timestamp exists and the files have been processed into MyMoney Database then files are not downloaded again. If the files have not yet been processed in the MyMoney database then files are re-downloaded every 30 minutes to make sure that file has the latest updated data.

For All data feeds to check if the file is eligible for download, the pattern of the file name is checked against *ETL.DataLoadFilenamePattern* table which contains the filename patterns for the various data feeds. If there is a problem with the *GetIncomingFiles* package, there is a provision to download the files manually using any FTP-S software and run the *LoadFMSDataLoad* and *LoadPMSDataLoad* manually to process the data feeds.

**Exclusions:**

**COA Data Feed:**

If there are multiple files for the same entity (Dept, appropriation unit etc) then the latest file is downloaded based on the most recent timestamp present on the filename.

**FMSV Data Feed:**

For FMSV data feed, if a monthly file exists then all the daily and weekly files with earlier timestamp will not be considered for processing, since monthly file is a full extract of all FMSV Vendor information.

### COA Data Feed

Chart of Accounts Data Feed referred to as COA Data Feed is the source for lookup tables for COA data elements which includes Department, Appropriation Unit, Expenditure Object and Location. This is a Full Data Feed generated daily.

* ***Execution steps as in LoadCOADataLoad.dtsx***

1. Check if any new files are available in the Incoming Files Directory.
2. Initialize the process of loading data by creating a record in ETL.dataload with the details on feed name (L for COA), type of feed (F for Full) and the status as S for started.
3. Drop foreign keys in the related staging tables, truncate the data and re-create the FKs. Refer to the procedure *ETL.spTruncateCOAStagingTables* for more details.
4. Check if the file name downloaded from FISA matches the expected pattern.
5. Create a new record in the ETL.*dataloadfile* table with the details of the file to be loaded.
6. Load data from the downloaded file for lookup data into staging tables according to the type of data element like DEPT etc. Change the status of load from started to T for staged.
7. Validate data in the Staging tables.(Refer: *ETL.spValidateCOADataLoad*)
8. Move the invalid records to the respective invalid tables for future analysis and delete the same from the staging tables. Change the status of load from staged to V for Validated.
9. Load Data from staging tables into the corresponding lookup tables in NYCCheckbook schema of MyMoney Database. Refer to the procedure *ETL.spLoadCOADataLoad* for more details. Change the status of load from Validated to L for Loaded.
10. Create a Statistics log files to be attached to the mail.
11. Move over all the loaded files from the existing location.
12. Delete Statistics SSIS log files any thing older than the current load processing date.
13. Send mail with success/failure notification.

* ***Business/Validation Rules for the staged data***

Each record is validated for the below rules and if one of them is true the record is invalidated with the appropriate reasoning by copying it to the corresponding invalid table and deleted from the staging table.

***Department:***

1. Missing values - DOC\_DEPT\_CD or DEPT\_NM is null.
2. Duplicate records - If any duplicate records based on DOC\_DEPT\_CD are found in the staging tables then the rule is to load one and to invalidate the rest.

***Appropriation Unit:***

1. Missing values - DEPT\_CD or FCLS\_CD or FY or APPR\_CD or UA\_ATTR\_NM is null.
2. Duplicate records - If any duplicate records based on DEPT\_CD, FCLS\_CD, FY & APPR\_CD are found in the staging tables then the rule is to load one and to invalidate the rest.

***Expenditure Object:***

1. Missing values - FY or OBJ\_CD or OBJ\_NM or OBJ\_SH\_NM is null.
2. Duplicate records - If any duplicate records based on FY &OBJ\_CD are found in the staging tables then the rule is to load one and to invalidate the rest..

***Location:***

1. Missing values - DOC\_DEPT\_CD or LOC\_CD or LOC\_NM or LOC\_SH\_NM or LOC\_NM\_UP is null.
2. Duplicate records - If any duplicate records based on DOC\_DEPT\_CD & LOC\_CD are found in the staging tables then the rule is to load one and to invalidate the rest.

* ***Rules to load data from Staging tables to NYCCheckBook Schema***

***Department:***

1. A row is inserted into *NYCCheckbook.FMSDepartment* table when a row in the staging table identified by DOC\_DEPT\_CD doesn’t match with the *NYCCheckbook.FMSDepartment* table identified by DeptCode.
2. A row is updated in  *NYCCheckbook.FMSDepartment* table when a row in the staging table identified by DOC\_DEPT\_CD matches with the *NYCCheckbook.FMSDepartment* table identified by DeptCode.
3. A row is not deleted from the *NYCCheckbook.FMSDepartment* table when a row in the FMSDepartment table identified by DeptCode is not available in *ETL.*FMSDepartmentStage table identified by DOC\_DEPT\_CD

***Appropriation Unit:***

1. A row is inserted into *NYCCheckbook. FMSAppropriationUnit* table when a row in the staging table identified by APPR\_CD, DEPT\_CD, FLSC\_CD and FY doesn’t match with the *NYCCheckbook. FMSAppropriationUnit* table identified by AppropriationUnitCode, DeptCode, FundClassCode and FiscalYear
2. A row is updated in *NYCCheckbook. FMSAppropriationUnit* table when a row in the staging table identified by APPR\_CD, DEPT\_CD, FLSC\_CD and FY matches with the *NYCCheckbook. FMSAppropriationUnit* table identified by AppropriationUnitCode, DeptCode, FundClassCode and FiscalYear.
3. A row is not deleted from the *NYCCheckbook. FMSAppropriationUnit* table when a row in the FMSAppropriationUnit table identified by AppropriationUnitCode, DeptCode, FundClassCode and FiscalYear is not available in *ETL.FMSAppropriationUnitStage* table identified by APPR\_CD, DEPT\_CD, FLSC\_CD & FY.

***Expenditure Object:***

1. A row is inserted into *NYCCheckbook.FMSExpenditureObject* table when a row in the staging table identified by OBJ\_CD and FY doesn’t match with the *NYCCheckbook.FMSExpenditureObject* table identified by ExpenditureObjCode and FiscalYear.
2. A row is updated in *NYCCheckbook.FMSExpenditureObject* table when a row in the staging table identified by OBJ\_CD and FY matches with the *NYCCheckbook.FMSExpenditureObject* table identified by ExpenditureObjCode and FiscalYear.
3. A row is not deleted from the *NYCCheckbook.FMSExpenditureObject* table when a row in the FMSExpenditureObject table identified by ExpenditureObjCode and FiscalYear is not available *ETL.FMSExpenditureObjectStage* table identified by OBJ\_CD and FY.

***Location:***

1. A row is inserted into *NYCCheckbook.FMSLocation* table when a row in the staging table identified by DOC\_DEPT\_CD and LOC\_CD doesn’t match with the *NYCCheckbook.FMSLocation* table identified by DeptCode and LocationCode.
2. A row is updated in *NYCCheckbook.FMSLocation* table when a row in the staging table identified by DOC\_DEPT\_CD and LOC\_CD matches with the *NYCCheckbook.FMSLocation* table identified by DeptCode and LocationCode.
3. A row is not deleted from the *NYCCheckbook.FMSLocation* table when a row in the FMSLocation table identified by DeptCode and LocationCode is not available in *ETL.FMSLocationStage* table identified by DOC\_DEPT\_CD and LOC\_CD.

### FMSV Data Feed

FMSV data feed provides information related to FMS Vendors. Data Feeds are provided as a Daily, Weekly and Monthly feeds. Daily and Weekly feeds have incremental vendor information that have changed in FMS/3 since the last extract of the same type but monthly data feed is a full extract of the active vendor Information. As a combination of these files could be received the same day, it is imperative to process these in the same order of creation based on the timestamp in the FMSV feed file name . For this purpose a data load is created for each file in the order of creation.

FMSV Data File Includes information related to Vendor, Vendor Business Type, Vendor Address & Address Type. These four sets of data is identified by record type field (V for vendor, M for business type, A for address and T for address type). For each Vendor record, there is zero, one, or more Vendor Business Type records, and one or more Vendor Address records. For each Vendor Address record, there are one or more Vendor Address Type records. There is 1 staging table and 1 invalid table for each of these 4 data sets.

Vendor business type determines if a vendor is identified within FMS/3 as a Woman-Owned Enterprise, a Minority-Owned Enterprise, a Local Business, or an Emerging Enterprise. Of these four business types, only the first two are considered to determine if a vendor is a MWBE vendor.

* ***Execution steps as in LoadFMSVDataLoads.dtsx***

1. Check if any new files are available in the Incoming files directory .
2. Initialize the process of loading data by creating a record in *ETL.dataload* with the details on feed name (V for FMSV), type of feed (F for Full, I for Incremental) and the status as S for Started.
3. Drop foreign keys in the related staging tables, truncate the data and re-create the FKs. Refer to the procedure *ETL.spTruncateFMSVStagingTables* for more details.
4. Check if the file name downloaded from FISA matches against the expected pattern.
5. Create a new record in the ETL.dataloadfile table with the details of the file to be loaded.
6. If the record is malformed then load file into malformed tables (*ETL.MalformedRecords*) for further Analysis.
7. Load data from the downloaded file into staging tables based on the record Type.
8. Validate the data in the Staging tables.(Refer: *ETL.spValidateFMSVDataLoad*)
9. Move the invalid records to the respective invalid tables for future analysis and delete the same from the staging tables.
10. Load Data from staging tables into the corresponding lookup tables in NYCCheckbook schema of MyMoney Database. (Refer*: ETL.spLoadFMSVDataLoad*).
11. Create a Statistics log files to be attached to the mail.
12. Move over all the loaded files from the existing location.
13. Delete Statistics SSIS log files any thing older than the current load processing date.
14. Send mail with success/failure notification.

* ***Business/Validation Rules for the staged data***

Each record is validated for the below rules and if one of them is true the record is invalidated with the appropriate reasoning by copying it to the corresponding invalid table and deleted from the staging table.

***Vendor:***

1. Missing key values - VEND\_CUST\_CD or VEND\_ACT\_STA or VEND\_APRV\_STA or ORG\_TYP is null or combination of the LGL\_NM, ALIAS\_NM, FIRST\_NM, MID\_NM, LAST\_NM, COMP\_NM is null.
2. Duplicate records - If any duplicate records based on VEND\_CUST\_CD are found in the staging tables then the rule is to load one and to invalidate the rest.
3. Invalid/Inconsistent values - EFFBGN\_DT greater than EFFEND\_DT.
4. Vendors associated to invalid related entities

* Vendors associated with vendor business type records that are invalidated for reasons other than ‘Duplicate’.
* Vendors associated with vendor address records that are invalidated for reasons other than ‘Duplicate’.
* Vendors associated with vendor address type records that are invalidated for reasons other than ‘Duplicate’.

***Vendor Business Type:***

1. Missing key values - VEND\_CUST\_CD or BUS\_TYP or BUS\_TYP\_STA or DISP\_CERT\_STRT\_DT is null.
2. Duplicate records - If any duplicate records based on VEND\_CUST\_CD and BUS\_TYP are found in the staging tables then the rule is to load one and to invalidate the rest.
3. Invalid/Inconsistent values

* BUS\_TYP (Business type) does not match with *NYCCheckbook.VendorBusinessType*.
* BUS\_TYP\_STA (Business type status) does not match with *NYCCheckbook.BusinessTypeStatus*.
* MIN\_TYP (Minority type) does not match with *NYCCheckbook.EthnicMinority.*
* Minority type(MIN\_TYPE) is not provided for an accepted (BUS\_TYP\_STA = 2) business type which is minority (MNRT).
* Minority type(MIN\_TYPE) is provided for a business type which is not minority (MNRT).
* DISP\_CERT\_STRT\_DT > CERT\_END\_DT

1. Vendor business types associated to invalid related entities

* Vendor Business Types associated with vendor records that are invalidated for reasons other than ‘Duplicate’.
* Vendor Business Types associated with business type records that are invalidated for reasons other than ‘Duplicate’.
* Vendor Business Types associated with address records that are invalidated for reasons other than ‘Duplicate’.
* Vendor Business Types associated with address type records that are invalidated for reasons other than ‘Duplicate’.

1. Missing referenced parent entity - Business type records whose vendor information is not available in the staging table related to vendor.

***Vendor Address:***

1. Missing key values - VEND\_CUST\_CD or AD\_ID or CTRY or STR\_1\_NM is null. CTRY=’US’ and ST/ZIP/CITY\_NM is null.
2. Duplicate records - If any duplicate records based on VEND\_CUST\_CD and AD\_ID are found in the staging tables then the rule is to load one and to invalidate the rest.
3. Invalid/Inconsistent values - EFBGN\_DT is after EFEND\_DT
4. Vendor address associated to invalid related entities

* Vendor address associated with vendor records that are invalidated for reasons other than ‘Duplicate’.
* Vendor address associated with business type records that are invalidated for reasons other than ‘Duplicate’.
* Vendor address associated with address records that are invalidated for reasons other than ‘Duplicate’.
* Vendor address associated with address type records that are invalidated for reasons other than ‘Duplicate’.

1. Missing referenced parent entity - address records whose vendor information is not available in the staging table related to vendor.

***Vendor Address Type:***

1. Missing key values - VEND\_CUST\_CD or AD\_ID or AD\_TYP or EFFBGN\_DT is null.
2. Duplicate records - If any duplicate records based on VEND\_CUST\_CD, AD\_ID and AD\_TYP are found in the staging tables then the rule is to load one and to invalidate the rest.
3. Invalid/Inconsistent values - AD\_TYP (Address type) does not match with YCCheckbook.AddressType
4. Vendor address type associated to invalid related entities

* Vendor address type associated with vendor records that are invalidated for reasons other than ‘Duplicate’.
* Vendor address type associated with business type records that are invalidated for reasons other than ‘Duplicate’.
* Vendor address type associated with address records that are invalidated for reasons other than ‘Duplicate’.
* Vendor address type associated with address type records that are invalidated for reasons other than ‘Duplicate’.

1. Missing referenced parent entity

* Address type records whose vendor information is not available in the staging table related to vendor.
* Address type records whose vendor address information is not available in the staging table related to vendor address.
* ***Rules to load data from Staging tables to NYCCheckBook Schema***

***Vendor:***

1. A row is inserted into *NYCCheckbook.FMSVendor* table when a row in the staging table *ETL.FMSVendorStage* identified by VEND\_CUST\_CD doesn’t match *NYCCheckbook.FMSVendor* identified by VendorCode.
2. A row is updated in *NYCCheckbook.FMSVendor* table when a row in the staging table ETL.FMSVendorStage identified by VEND\_CUST\_CD matches *NYCCheckbook.FMSVendor* identified by VendorCode.
3. A row is deleted from the NYCCheckbook.FMSVendor table (copied to *Graveyard. FMSVendor*) when a row in the NYCCheckbook.FMSVendor table identified by VendorCode is not available in ETL.FMSVendorStage table identified by VEND\_CUST\_CD only for data load corresponding to the monthly feed.

***Vendor Business Type:***

1. A row is inserted into *NYCCheckbook. FMSVendorBusinessType* table when a row in the staging table ETL. FMSVendorBusinessTypeStage identified by VEND\_CUST\_CD and BUS\_TYP doesn’t match *NYCCheckbook. FMSVendorBusinessType* identified by VendorCode and BusinessTypeCode.
2. A row is updated in *NYCCheckbook. FMSVendorBusinessType* table when a row in staging *table ETL. FMSVendorBusinessTypeStage* identified by VEND\_CUST\_CD and BUS\_TYP matches *NYCCheckbook. FMSVendorBusinessTyp*e identified by VendorCode and BusinessTypeCode.
3. A row is deleted from the *NYCCheckbook. FMSVendorBusinessType* table (copied to *Graveyard.* FMSVendorBusinessType) when a row in the *NYCCheckbook. FMSVendorBusinessType* table identified by VendorCode and BusinessTypeCode is not available in staging table ETL. FMSVendorBusinessTypeStage identified by VEND\_CUST\_CD and BUS\_TYP only for data load corresponding to the monthly feed.

***Vendor Address:***

1. A row is inserted into NYCCheckbook. FMSVendorAddress table when a row in the staging table *ETL. FMSVendorAddressStage* identified by VEND\_CUST\_CD and AD\_ID doesn’t match NYCCheckbook. FMSVendorAddress identified by VendorCode and AddressCode.
2. A row is updated in NYCCheckbook. FMSVendorAddress table when a row in staging *table* ETL. FMSVendorAddressStage identified by VEND\_CUST\_CD and AD\_ID *matches NYCCheckbook. FMSVendorAddress* identified by VendorCode and AddressCode.
3. A row is deleted from the NYCCheckbook. FMSVendorAddress table (copied to Graveyard. FMSVendorAddress) when a row in the *NYCCheckbook.* FMSVendorAddress table identified by VendorCode and AddressCode is not available in staging table *ETL. FMSVendorAddressStage* identified by VEND\_CUST\_CD and AD\_ID only for data load corresponding to the monthly feed.

***Vendor Address Type:***

1. A row is inserted into NYCCheckbook. FMSVendorAddressType table when a row in the staging table *ETL. FMSVendorAddressTypeStage* identified by VEND\_CUST\_CD, AD\_ID and AD\_TYP doesn’t match NYCCheckbook. FMSVendorAddressType identified by VendorCode, AddressCode and AddressTypeCode.
2. A row is updated in NYCCheckbook. FMSVendorAddressType table when a row in staging *table* ETL. *FMSVendorAddressTypeStage* identified by VEND\_CUST\_CD, AD\_ID and AD\_TYP *matches NYCCheckbook.* FMSVendorAddressType identified by VendorCode, AddressCode and AddressTypeCode.
3. A row is deleted from the NYCCheckbook. FMSVendorAddressType table (copied to Graveyard. FMSVendorAddressType) when a row in the *NYCCheckbook.* FMSVendorAddressType table identified by, AddressCode and AddressTypeCode is not available in staging table *ETL. FMSVendorAddressTypeStage* identified by VEND\_CUST\_CD, AD\_ID and AD\_TYP only for data load corresponding to the monthly feed.

### CON Data Feed

CON Data Feed provides information related to Contracts also referred to as Agreements. Data is provided as daily and monthly incremental feeds. As a combination of these files could be received the same day, it is imperative to process these in the same order of creation. For this purpose a data load is created for each file in the order of creation. Information related to documents of several types namely CT1, CTA1, CTA2, POC, PCC1, POD and DO1 are provided in the same feed. Layout for each of this is different.

Data is heterogeneous in nature and only Header and Detail records identified by record type H and W respectively are considered. Header record for all documents other than Delivery order (DO1) is staged in FMSContractStage, award detail records (DO1 does not have award detail record) are staged in FMSContractAwardStage. Header records for DO1 are staged in FMSDeliveryOrderStage.

* ***Execution steps as in LoadCONDataLoads.dtsx***

1. Check if any new files are available in the Incoming directory.
2. Initialize the process of loading data by creating a record in ETL.dataload with the details on feed name (C for CON), type of feed (I for incremental) and the status as S for started.
3. Drop foreign keys in the related staging tables, truncate the data and re-create the FKs. Refer to the procedure ETL. *spTruncateCONStagingTables* for more details.
4. Check if the file name downloaded from FISA matches againt the filename pattern.
5. Create a new record in the ETL.dataloadfile table with the details of the file to be loaded.
6. If the record is malformed then load file into malformed tables (*ETL.MalformedRecords*) for further Analysis.
7. Load data from the downloaded file into staging tables based on the record type. Change the status of load from started to T for staged.
8. Validate data in the Staging tables.(Refer: *ETL.spValidateCONDataLoad*).
9. Move the invalid records to the respective invalid tables for future analysis and delete the same from the staging tables. Change the status of load from staged to V for Validated.
10. Load Data from staging tables into the corresponding lookup tables in NYCCheckbook schema of MyMoney Database. Refer to the procedure *ETL. spLoadCONDataLoad* for more details. Change the status of load from Validated to L for Loaded.
11. Create a Statistics log files to be attached to the mail.
12. Move over all the loaded files from the existing location.
13. delete all Statistics SSIS log files
14. Send mail with success/failure notification.

* ***Business/Validation Rules for the staged data***

Each record is validated for the below rules and if one of them is true the record is invalidated with the appropriate reasoning by copying it to the corresponding invalid table and deleted from the staging table.

***FMS Contract:***

1. Missing values - DOC\_CD or DOC\_DEPT\_CD or DOC\_ID or DOC\_VERS\_NO is null.
2. Duplicate records - If any duplicate records based on DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO

are found in the staging tables then the rule is to load one and to invalidate the rest.

1. Missing award detail – Awards of type CT1, CTA1 and CTA2 not associated to any award detail.
2. Associated with multiple award detail – Awards of type non DO1 not associated to multiple award detail records.
3. Invalid/Inconsistent values

* CNTRCT\_STRT\_DT > CNTRCT\_END\_DT.
* MAX\_CNTRC\_AM < 0

1. Awards associated to invalid related entities – Awards associated with award detail records that are invalidated for reasons other than ‘Duplicate’

***FMS Contract Award:***

1. Missing values - DOC\_CD or DOC\_DEPT\_CD or DOC\_ID or DOC\_VERSION\_NO or AWD\_METH\_CD is null.
2. Duplicate records - If any duplicate records based on based on DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO and AWD\_METH\_CD are found in the staging tables then the rule is to load one and to invalidate the rest
3. Missing referenced parent entity. Award detail records whose award information is not available in the staging table corresponding to FMS Contract.
4. Invalid/Inconsistent values –

* AWD\_METH\_CD (Award method) does not match with NYCCheckbook.AgreementAwardMethod
* CTTYP\_CD (Contract type) not null and does not match with NYCCheckbook.AgreementType
* CTCAT\_CD\_1 (Contract Category) not null and does not match with NYCCheckbook. AgreementCategory.

1. Award detail records associated to invalid related entities – Award detail record associated with awards (header record) that are invalidated for reasons other than ‘Duplicate’.

***FMS Delivery Order:***

1. Missing key values - DOC\_CD or DOC\_DEPT\_CD or DOC\_ID or DOC\_VERSION\_NO is null then invalidate the record and move the record to the corresponding invalid tables.
2. Duplicate records - If any duplicate records based on DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO

are found in the staging tables then the rule is to load one and to invalidate the rest.

1. DO1 which are not associated with MAG (AGREE\_DOC\_CD/ AGREE\_DOC\_DEPT\_CD/ AGREE\_DOC\_ID) or any of the three values are missing.

* ***Rules to load data from Staging tables to NYCCheckBook Schema***

***FMS Contracts:***

1. A row is inserted into *NYCCheckbook.FMSContracts* table when a row in the staging table *ETL.FMSContractsStage* & *ETL.FMSContractsAwardStage* identified by DOC\_CD, DOC\_DEPT\_CD, DOC\_ID doesn’t match *NYCCheckbook.FMSContracts* identified by DocCode, DocDeptCode & DocID in addition to the below conditions
   1. At least one of the data elements namely required CNTRCT\_STRT\_DT, CNTRCT\_END\_DT, MAX\_CNTRC\_AM, DOC\_DSCR, DOC\_NM, AWD\_METH\_CD, CTTYP\_CD, CTCAT\_CD\_1 to have a value
   2. Consider the latest version of the document when multiple versions are in the staging table.
2. A row is updated in *NYCCheckbook.FMSContracts* table when a row in the staging table *ETL.FMSContractsStage* & *ETL.FMSContractsAwardStage* identified by DOC\_CD, DOC\_DEPT\_CD & DOC\_ID matches *NYCCheckbook.FMSContracts* identified by DocCode, DocDeptCode & DocID in addition to the 2 conditions mentioned above.
3. A row is not deleted from the *NYCCheckbook.FMSContracts* table when a row in the *NYCCheckbook.FMSContracts* table identified by DocCode, DocDeptCode & DocID is not available in *ETL.FMSContractsStage* and ETL.FMSContractsAwardStage identified by DOC\_CD, DOC\_DEPT\_CD & DOC\_ID.

***FMS Delivery Orders:***

1. A row is inserted into *NYCCheckbook.FMSDeliveryOrders* table when a row in the staging table *ETL. FMSDeliveryOrderStage* identified by DOC\_CD, DOC\_DEPT\_CD, DOC\_ID doesn’t match *NYCCheckbook.FMSDeliveryOrders* identified by DocCode, DocDeptCode & DocID.
2. A row is updated in *NYCCheckbook.FMSDeliveryOrders* table when a row in the staging table *ETL. FMSDeliveryOrderStage* identified by DOC\_CD, DOC\_DEPT\_CD and DOC\_ID matches *NYCCheckbook.FMSDeliveryOrders* table identified by DocCode, DocDeptCode & DocID.
3. A row is not deleted from the *NYCCheckbook. FMSDeliveryOrders* table when a row in *the NYCCheckbook. FMSDeliveryOrders* table identified by DocCode, DocDeptCode & DocID is not available in ETL. FMSDeliveryOrderStage identified by DOC\_CD, DOC\_DEPT\_CD & DOC\_ID.

### MAG Data Feed

The MAG Feed provides information related to Master Agreement. Data is provided as daily and monthly incremental feeds. As a combination of these files could be received the same day, it is imperative to process these in the same order of creation. For this purpose a data load is created for each file in the order of creation.

Data is heterogeneous in nature and only Header and Detail records identified by record type H and W respectively are considered. Header records are staged in *FMSMasterAgreementStage* and award detail records are staged in *FMSMasterAgreementAwardStage*.

* ***Execution steps as in* LoadMAGDataLoads*.dtsx***

1. Check if any new files are available to be downloaded from FISA.
2. Initialize the process of loading data by creating a record in ETL.dataload with the details on feed name (M for MAG), type of feed (I for incremental) and the status as S for started.
3. Drop foreign keys in the related staging tables, truncate the data and re-create the FKs. Refer to the procedure *ETL. spTruncateMAGStagingTables* for more details.
4. Check if the file downloaded from FISA is in proper format.
5. Create a new record in the ETL.dataloadfile table with the details of the file to be loaded.
6. If the record is malformed then load file into malformed tables (*ETL.MalformedRecords*) for further Analysis.
7. Load data from the downloaded file into staging tables based on the record type. Change the status of load from started to T for staged.
8. Validate data in the Staging tables.(Refer: *ETL.spValidateMAGDataLoad*).
9. Move the invalid records to the respective invalid tables for future analysis and delete the same from the staging tables. Change the status of load from staged to V for Validated.
10. Load Data from staging tables into the corresponding lookup tables in NYCCheckbook schema of MyMoney Database. Refer to the procedure *ETL. spLoadMAGDataLoad* for more details. Change the status of load from Validated to L for Loaded.
11. Create a Statistics log files to be attached to the mail.
12. Move over all the loaded files from the existing location.
13. delete all Statistics SSIS log files
14. Send mail with success/failure notification.

* ***Business/Validation Rules for the staged data***

Each record is validated for the below rules and if one of them is true the record is invalidated with the appropriate reasoning by copying it to the corresponding invalid table and deleted from the staging table.

***FMS Master Agreement:***

1. Missing key values - DOC\_CD or DOC\_DEPT\_CD or DOC\_ID or DOC\_VERS\_NO is null.
2. Duplicate records - If any duplicate records based on DOC\_CD, DOC\_DEPT\_CD, DOC\_ID and DOC\_VERS\_NO are found in the staging tables then the rule is to load one and to invalidate the rest.
3. Missing MAG award detail – MAG record without an award detail record.
4. Multiple MAG award detail – MAG record associated to more than one one award detail record.
5. Invalid/Inconsistent values -

* EFBGN\_DT > EFEND\_DT.
* MA\_PRCH\_LMT\_AM < 0

1. MAG associated to invalid MAG award detail record.
2. Invalid/Inconsistent values -

* EFBGN\_DT > EFEND\_DT.
* MA\_PRCH\_LMT\_AM < 0

***FMS Master Agreement Award:***

1. Missing values - DOC\_CD or DOC\_DEPT\_CD or DOC\_ID or DOC\_VERS\_NO or DOC\_AWDDET\_LN\_NO is null.
2. Duplicate records - If any duplicate records based onDOC\_CD, DOC\_DEPT\_CD, DOC\_ID and DOC\_VERS\_NO and DOC\_AWDDET\_LN\_NO are found in the staging tables then the rule is to load one and to invalidate the rest
3. Missing referenced parent entity. MAG detail records whose MAG information is not available in the staging table corresponding to MAG header.
4. Invalid/Inconsistent values

* AWD\_METH\_CD (Award method) not null and does not match with NYCCheckbook.AgreementAwardMethod
* CTTYP\_CD (Master agreement type) not null and does not match with NYCCheckbook.AgreementType.
* CTCAT\_CD\_1 (Master agreement category) not null and does not match with NYCCheckbook. AgreementCategory.

1. MAG detail records associated to invalid related entities – MAG detail record associated with MAG (header record) that are invalidated for reasons other than ‘Duplicate’.

* ***Rules to load data from Staging tables to NYCCheckBook Schema***

***FMS Master Agreement:***

1. A row is inserted into *NYCCheckbook. FMSMasterAgreement* table when a row in the staging table *ETL. FMSMasterAgreementStage* *& ETL. FMSMasterAgreementAwardStage* identified by DOC\_CD, DOC\_DEPT\_CD, DOC\_ID doesn’t match NYCCheckbook. FMSMasterAgreement identified by DocCode, DocDeptCode & DocID in addition to the below conditions
   1. At least one of the data elements namely EFFBGN\_DT or EFFEND\_DT or MA\_PRCH\_LMT\_AM or DOC\_DSCR or DOC\_NM or AWD\_METH\_CD or CTTYP\_CD or CTCAT\_CD\_1 to have a value
   2. Consider the latest version of the document when multiple versions are in the staging table.
2. A row is updated in *NYCCheckbook. FMSMasterAgreement* table when a row in the staging table *ETL. FMSMasterAgreementStage* *& ETL. FMSMasterAgreementAwardStage* identified by DOC\_CD, DOC\_DEPT\_CD & DOC\_ID matches *NYCCheckbook. FMSMasterAgreement* identified by DocCode, DocDeptCode & DocID in addition to the 2 conditions mentioned above.
3. A row is not deleted from the *NYCCheckbook. FMSMasterAgreement* table when a row in the *NYCCheckbook. FMSMasterAgreement* table identified by DocCode, DocDeptCode & DocID is not available in *ETL. FMSMasterAgreementStage* and ETL. *FMSMasterAgreementAwardStage* identified by DOC\_CD, DOC\_DEPT\_CD & DOC\_ID.

### PMS Data Feed

The PMS Feed provides information related to Payroll Management System. The data provided by FISA is at a summarized level and is a incremental feed. Payroll summary information is provided only for general fund (Fund Class 001).

* ***Execution steps as in* LoadPMSDataLoads*.dtsx***

1. Check if any new files are available to be downloaded from FISA.
2. Initialize the process of loading data by creating a record in ETL.dataload with the details on feed name (P for PMS), type of feed (I for Incremental) and the status as S for started.
3. Drop foreign keys in the related staging tables, truncate the data and re-create the FKs. Refer to the procedure *ETL.* *spTruncatePMSStagingTables* for more details.
4. Check if the file downloaded from FISA is in proper format.
5. Create a new record in the ETL.*dataloadfile* table with the details of the files to be loaded.
6. Load data from the downloaded files into the staging table. Change the status of load from started to T for staged.
7. Validate data in the Staging tables.(Refer: *ETL.spValidatePMSDataLoad*)
8. Move the invalid records to the respective invalid tables for future analysis and delete the same from the staging tables. Change the status of load from staged to V for Validated.
9. Load Data from staging tables into the corresponding lookup tables in NYCCheckbook schema of MyMoney Database. Refer to the procedure *ETL.spLoadPMSDataLoad* for more details. Change the status of load from Validated to L for Loaded.
10. Create a Statistics log files to be attached to the mail.
11. Move over all the loaded files from the existing location.
12. Delete all Statistics SSIS log files.
13. Send mail with success/failure notification.

* ***Business/Validation Rules for the staged data***

Each record is validated for the below rules and if one of them is true the record is invalidated with the appropriate reasoning by copying it to the corresponding invalid table and deleted from the staging table.

1. Missing key values - PayCycleCode/ PayDate/ PayrollNumCode/ PayrollNumName/ AppropriationUnitCode / FiscalYear/ ExpenditureObjCode/ DeptCode/ BudgetCode/ BudgetName/ PayrollAmount is null.
2. Duplicate records.

* Intra load duplicate records - If any duplicate records based on PayCycleCode,PayDate,PayrollNumCode,AppropriationUnitCode, FiscalYear,ExpenditureObjCode,DeptCode,BudgetCode are found in the staging tables then the rule is to load one and to invalidate the rest.
* Inter load duplicate records – If a record was already loaded from previous files (Presence of a matching record in NYCCheckbook.PayrollSummary for the same PayCycleCode,PayDate,PayrollNumCode,AppropriationUnitCode, FiscalYear,ExpenditureObjCode,DeptCode,BudgetCode) then the record in staging table is invalidated in order to avoid duplicates in the transaction table

1. Invalid/Inconsistent values – PayCycleCode does not match with NYCCheckbook.PayCycle

* ***Rules to load data from Staging tables to NYCCheckBook Schema***

1. Insert distinct payroll number information from *ETL.PayrollSummary* table which are not available in *NYCCheckbook.PayrollNumber* identified byPayrollNumCode and PayrollNumName.
2. Insert distinct BudgetCode from *ETL.PayrollSummary* table which are not available in *NYCCheckbook.Budget* identified by BudgetCode and BudgetName .
3. Insert distinct Department information from *ETL.PayrollSummaryStage* and *NYCCheckbook.FMSDepartment* which are not available in *NYCCheckbook.Department* table with DeptCode from the PMS feed and the Department Name as “<Unknown Department>”.
4. Insert distinct AppropriationUnit from *ETL.PayrollSummaryStage* and *NYCCheckbook.FMSAppropriationUnit* which are not available *in NYCCheckbook.AppropriationUnit* table based on AppropriationUnitCode ,DeptCode, FundClassCode(001 - General Fund), and FiscalYear.
5. Insert distinct ExpenditureObjCode from *ETL.PayrollSummaryStage* and *NYCCheckbook. FMSExpenditureObject* which are not available in *NYCCheckbook. ExpenditureObject* table based on ExpenditureObjCode and FiscalYear.
6. Insert New distinct Payrollnumber from *ETL.PayrollSummaryStage* table based on PayrollNumCode,PayrollnumberName into NYCCheckbook. Payrollnumber reference table.
7. Insert the records from *ETL.PayrollSummaryStage* to *NYCCheckbook.PayrollSummary* table, by matching values with lookup tables for DepartmentId, AppropriationUnitID, ExpenditureObjectID, PayrollNumID, and BudgetID from respective tables in NYCCheckbook schema.

### FMS Data Load

The FMS Feed provides information related to disbursement data as daily incremental files. Filename carries the information on the visibility of disbursements which can be Fully Displayed, Partially displayed or Not Displayed.

Data is heterogeneous in nature and the different types of records are identified by the record type. Feed contains one Header record, one vendor record and one or more accounting line records. Header records identified by H are staged in *ETL.DisbursementStage*, vendor records identified by record type V are staged in ETL.vendorstage and disbursement line items identified by record type as A are staged in *ETL.DisbLineItemStage.*

* ***Execution steps as in* LoadFMSDataLoads*.dtsx***

1. Check if any new files are available in the Incoming files directory.
2. Initialize the process of loading data by creating a record in ETL.dataload with the details on feed name (F for FMS), type of feed (I for incremental) and the status as S for started.
3. Drop foreign keys in the related staging tables, truncate the data and re-create the FKs. Refer to the procedure *ETL. spTruncateFMSStagingTables* for more details.
4. Check if the filename downloaded from FISA is matching the file pattern.
5. Create a new record in the ETL.dataloadfile table with the details of the file to be loaded.
6. If the record is malformed then load file into malformed tables (*ETL.MalformedRecords*) for further Analysis.
7. Load data from the downloaded file into staging tables based on the record type. Change the status of load from started to T for staged.
8. Validate data in the Staging tables.(Refer: *ETL.spValidateFMSDataLoad*).
9. Move the invalid records to the respective invalid tables for future analysis and delete the same from the staging tables. Change the status of load from staged to V for Validated.
10. Load Data from staging tables into the corresponding tables in NYCCheckbook schema of MyMoney Database. Refer to the procedure *ETL. spLoadFMSDataLoad* for more details. Change the status of load from Validated to L for Loaded.
11. Create a Statistics log files to be attached to the mail.
12. Move over all the loaded files from the existing location.
13. delete all Statistics SSIS log files
14. Send mail with success/failure notification.

* ***Business/Validation Rules for the staged data***

***Disbursement:***

1. Missing key values - DOC\_CD / DOC\_DEPT\_CD / DOC\_ID / DOC\_VERS\_NO / DOC\_REC\_DT\_DC / DOC\_BFY/ DOC\_FY\_DC/ DOC\_PER\_DC/ CHK\_EFT\_AM / CHK\_EFT\_ISS\_DT/ DISC\_AM/ INTR\_AM/ BKUP\_WHLD\_AM/ INCT\_AM/ RTG\_AM is null.
2. Duplicate records.

* Intra load duplicate records - If any duplicate records based on DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO are found in the staging tables then the rule is to load one and to invalidate the rest.
* Inter load duplicate records – If a record was already loaded from previous files (Presence of a matching record in *NYCCheckbook.Disbursement for* the same DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO) then the record in staging table is invalidated in order to avoid duplicates in the transaction table.

1. Missing vendor – Disbursements without any vendor information in the *ETL.vendorstage.*
2. Multiple vendors – Disbursements which are associated to multiple vendors in the *ETL.vendorstage.*
3. Missing disbursement line item - Disbursements without any disbursement line items in the *ETL.DisbLineItemStage.*
4. Disbursements associated to invalid related entities

* Disbursements associated with vendor records that are invalidated for reasons other than ‘Duplicate’.
* Disbursements associated with disbursement line item records that are invalidated for reasons other than ‘Duplicate’.

***Vendor:***

1. Missing key values - DOC\_CD/ DOC\_DEPT\_CD / DOC\_ID/ DOC\_VERS\_NO/VEND\_CUST\_CD/ LGL\_NM/ DOC\_VEND\_LN\_NO/ ORG\_CLS/MISC\_ACCT\_FL is null.
2. Duplicate records.

* Intra load duplicate records - If any duplicate records based on DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO are found in the staging tables then the rule is to load one and to invalidate the rest.
* Inter load duplicate records – If a record was already loaded from previous files (Presence of a matching record in *NYCCheckbook.Disbursement for* the same DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO) then the record in staging table is invalidated in order to avoid duplicates in the transaction table.

1. Missing referenced parent entity - Vendor records whose disbursement information is not available in the staging table related to disbursement.
2. Invalid/Inconsistent values - ORG\_CLS (Organizational classification) does not match with *NYCCheckbook.VendorOrgClassification.*
3. Vendors associated to invalid related entities

* Vendors associated with disbursement records that are invalidated for reasons other than ‘Duplicate’.
* Vendors associated with disbursement line item records that are invalidated for reasons other than ‘Duplicate’.

***Disbursement Line Items:***

1. Missing values - DOC\_CD/ DOC\_DEPT\_CD/ DOC\_ID/ DOC\_VERS\_NO/ DOC\_VEND\_LN\_NO/ DOC\_ACTG\_LN\_NO/ DEPT\_CD/ BFY/ FY\_DC/ PER\_DC/ CHK\_AM/ BKUP\_WHLD\_LN\_AM/ DISC\_LN\_AM/ INCT\_LN\_AM/ INTR\_LN\_AM/ RTG\_LN\_AM/ RFED\_DOC\_CD/ RFED\_DOC\_DEPT\_CD/ RFED\_DOC\_ID/ RFED\_VEND\_LN\_NO/ RFED\_ACTG\_LN\_NO is null.
2. Duplicate records.

* Intra load duplicate records - If any duplicate records based on DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO & DOC\_ACTG\_LN\_NO are found in the staging tables then the rule is to load one and to invalidate the rest.
* Inter load duplicate records – If a record was already loaded from previous files (Presence of a matching record in *NYCCheckbook.* DisbursementLineItem *for* the same DOC\_CD, DOC\_DEPT\_CD, DOC\_ID, DOC\_VERS\_NO & DOC\_ACTG\_LN\_NO) then the record in staging table is invalidated in order to avoid duplicates in the transaction table.

1. Missing referenced parent entity – Disbursement line item records whose disbursement information is not available in the staging table related to disbursement.
2. Disbursement line items associated to invalid related entities

* Disbursement line items associated with disbursement records that are invalidated for reasons other than ‘Duplicate’.
* Disbursement line item associated with vendor records that are invalidated for reasons other than ‘Duplicate’.
* ***Rules to load data from Staging tables to NYCCheckBook Schema***

Insert vendor information from VendorStage which are not available in the NYCCheckbook.Vendor identified by VendorCode, vendorsubcode and VendorName. Vendorsubcode is applicable only for miscellaneous vendors identified by a non-zero value, which are available in ETL.MiscVendo table For any disbursement made to a miscellaneous vendor, insert a new distinct row in the NYCCheckbook.Vendor table, generating an arbitrary value for the VendorSubCode column.

1. Insert distinct Department information from *ETL.* DisbLineItemStage and *NYCCheckbook.FMSDepartment* which are not available in *NYCCheckbook.Department* table with DeptCode from the FMS feed and the Department Name as “<Unknown Department>”..
2. Insert distinct AppropriationUnit from *ETL.*DisbLineItemStage and *NYCCheckbook.FMSAppropriationUnit* joined on {APPR\_CD, DEPT\_CD, FCLS\_CD, FY\_DC} which are not available *in NYCCheckbook.AppropriationUnit* table with the AppropriationUnitCode ,DeptCode, FundClassCode, and FiscalYear from FMS feed and the AppropriationUnitName as “Non Applicable Appropriation Unit”
3. <expenditure object rule>
4. Insert distinct Location from *ETL.*DisbLineItemStage and *NYCCheckbook.*FMSLocation joined on {LOC\_CD, DEPT\_CD} which are not available *in NYCCheckbook.*Locationtable with the LocationCode andDeptCode from the FMS feed and LocationName as “Unknown Location”.
5. Insert distinct agreement rows from ETL.Disblineitem table identified by RQPORF\_DOC\_CD, RQPORF\_DOC\_DEPT\_CD, RQPORF\_DOC\_ID into NYCCheckbook.Agreement table based on AgreementDocCode, AgreementDocDeptCode and AgreementDocID when all three values are not null.
6. Insert records from ETL.DisbursementStage into NYCCheckbook.disbursement table. Disbursement status is set to ‘C’ for any disbursements with version number greater than 1.
7. Insert records from ETL.DisbLineItemStage into NYCCheckbook.disbursementLineItem table by looking up the necessary foreign key (DepartmentID, AppropriationUnitID, and ExpenditureObjectID from respective tables in NYCCheckbook schema).
8. Insert records from ETL.DisbLineItemStage and NYCCheckbook.Location into NYCCheckbook. DisbursementLineItemLocation table.
9. Insert records from ETL.DisbLineItemStage and NYCCheckbook.Agreement into NYCCheckbook. DisbursementLineItemAgreement table.
10. Insert distinct address into NYCCheckbook.VendorAddress from NYCCheckbook.FMSVendorAddress for the loaded disbursements by considering only the ordering address (identified by address type as ‘PR’) when it is not already available in NYCCheckbook.VendorAddress
11. Insert records into NYCCheckbook.DisbursementVendorAddress for the loaded disbursements.
12. Insert records into NYCCheckbook.DisbursementBusinessType for the loaded disbursements with the Vendor Business Type Status Code is 2 - “Accepted”.
13. Insert records into NYCCheckbook. DisbursementMinorityfor the loaded disbursements with the Vendor Business Type is “MNRT” and Vendor Business Type Status Code is 2 - “Accepted”.
14. Insert records from FMSContracts into NYCCheckbook.AgreementDetail for the loaded disbursements by matching FMSContract.{DocCode, DocDeptCode, DocID} = Agreement.{AgreementDocCode, AgreementDocDeptCode, AgreementDocID}. Agreement detail attributes unchanged compared to those in the most recent version of agreement detail will not be inserted.
15. Insert records from FMSContract joining with NYCCheckbook.Agreement table and NYCCheckbook. DisbursementLineItemAgreement table to popluated NYCCheckbook. DisbursementLineItemAgreementDetail table.
16. Insert records from FMSMasterAgreement joining FMSDeliveryOrder to populate NYCCheckbook. MasterAgreementDetail table by matching FMSDeliveryOrder.{DocCode, DocDeptCode, DocID} = Agreement.{AgreementDocCode, AgreementDocDeptCode, AgreementDocID}. Master Agreement detail attributes unchanged compared to those in the most recent version of master agreement detail will not be inserted.
17. Insert records from FMSDeliveryOrder joining NYCCheckbook.DisbursementLineItem and NYCCheckbook.DisbursementLineItemAgreement and NYCCheckbook.Agreement to popluated NYCCheckbook.DisbLineItemMasterAgrDetail.

## Questions

|  |  |
| --- | --- |
| **#** | **Question** |
| 1 | What are the next step after Invalid records are found for the correction? |
| 2 | What are the next step after malformed records are found for the correction? |
| 3 | Are there any Internal/External Applications that are dependent on MyMoney, MyMoneyPublic and MyMoneyPublicR Databases other than Checkbook 1.0 and MWBE? |
| 4 | What is the significance of LoadAgreementIndustryType.dtsx  and ImportLookups.dtsx as these are not part of the Controller Package? |
| 5 | As only new disbursements/payroll summary records are being processed and no updates/deletes are done as per the current ETL implementation, how are data corrections done in FISA being handled to reflect the same in MyMoney database? For instance the data correction can be modifying the appropriation unit for a disbursement which was earlier provided with an incorrect/missing appropriation unit. |
| 6 | Is there any standard process defined to load lookup tables like  AgreementCategory,VendorOrgClassification,BusinessType,AgreementAwardMethod, AgreementType, VendorAddressType etc? |
| 7 | As per Dmitri there is a rare possibility that a PMS Records might be updated , though not available in the ETL process. If the possibility exists we see that duplicate records are just ignored in the ETL process.  **Note:** We haven’t seen any record in the MyMoney Database with the specific scenario. |
| 8 | How is “Clearview” database data being populated on Rackspace? – We have not found any ETL SSIS packages related to Clearview applicationd data |
| 9 | No contract amount field is mapped for the Delivery Orders, is there any business reason for not mapping any contract dollar amount for Delivery Orders data? |