File Import Engine

# Goal:

Create a program that can load in files from FTP or API and import it into internal database. It will be configurable via tables. The process should be automated and self-correcting (meaning it should resolve basic issues (e.g. reload a file) or let us know of problems so we can fix it)

Backend:

MSFT

SQL Server 2019

(Also use: SSRS, Power BI)

Data Sources:

For this project the platform will be used to implement 4 data imports:

* API (gives data as JSON) There will be three data sources available by API for this initial project:
  1. <https://data.cms.gov/provider-data/dataset/4pq5-n9py>
  2. <https://data.cms.gov/provider-data/dataset/y2hd-n93e>
  3. <https://data.cms.gov/Special-Programs-Initiatives-COVID-19-Nursing-Home/COVID-19-Nursing-Home-Dataset/s2uc-8wxp>
* destination- new table (source and destination names are the same)
* FTP/CSV (and other flat file)
  1. 16 files- this will require special mapping from staging to prod
* Destination: Existing tables so mapping required

In your quote/when we discuss, please let me know what would be the process to technically create a new file import process and how much it would cost to build a new similar one. let’s say this one: [MDS Quality Measures](https://data.cms.gov/provider-data/dataset/djen-97ju).

Each file will be moved into staging first and then to production

# Process

User will use tables to setup/control an import (see the proposed “table structure” below. Feel free to suggest changes as this is just a guide).

Each job will have one or more steps depending on the goal. The user specifies the task and the appropriate parameters and the engine will perform it and log each step (we have a stored procedure for this- you just pass in parameters). Here’s the list of actions that should be supported

1. FTP- get data from an FTP to a local drive
2. Import API- - Download data via API into a Staging table
   1. use the mapping table configuration to put into staging table
   2. *Bonus*: If the staging table doesn’t exist- create it dynamically and create the mapping dynamically
3. Import delimited file- a parameter will specify delimiter (e.g. comma, pipe or tab) and escape character e.g. quotes
   1. use the mapping table configuration to put into staging table
   2. *Bonus*: If the staging table doesn’t exist- create it dynamically and create the mapping dynamically
4. Mapping Move- this is typically used to move staging to production- use the mapping table to map fields from source to destination.
5. Run SQL- this will just execute SQL code specified.
   1. E.g. Truncate table
6. Execute SSIS Package- name of SSIS package to execute and pass in parameters
7. Merge- given a source and destination and primary key move the new and updated records (probably use SSIS merge functionality)
8. Validation- we have stored procedures that run SQL commands. Here the process will execute the validation SQLs for this job (which stores results). Then query results table to determine if it should proceed or error event is triggered (and email is sent).
   1. Validations for these initial files and probably all include
      1. did it load?
      2. Is it duplicate of last load?

## Sample in action

**Sample 1. Import API**

1. Import API to Staging- Source: [API2](Http://5); Destination: StagingTable999, MapTable: 104
2. Validation- Did it load, duplication
3. Mapping Move: From Table: Staging5; To Table: Prod5, MapTable: 105
4. Validation- did it move?

**Sample 2. Import 16 csv file from FTP**

1. FTP- Connection:FTP1,FilesToGet:\*,Destination:c:\staging
2. Import delimited file- Source:File1, Destination:StagingFile1, Delimiter:csv, MapTable: 101
3. Import delimited file- Source:File2, Destination:StagingFile2, Delimiter:csv, MapTable: 102

… here you do the import step for all files

1. Validation- Did it load, duplication do for all files
2. Mapping Move: From Table: StagingFile1; To Table: Prod1, MapTable: 201
3. Mapping Move: From Table: StagingFile1; To Table: Prod1, MapTable: 202

… here you do the import step for all files

1. Validation- did it move

# General design:

1. Keep in mind: Scalability/modular
2. The process is driven by tables- limit hard coding. Let me know if it’s needed somewhere before doing it
3. At each step- log it using stored procedures I’ll provide (if you have a better suggestion for logging we can/should discuss it)
4. Alert on failure- email
5. When a run is missed it should be able to update on the next run
6. The program should fail gracefully and inform admin of issues

# Future considerations

These aren’t being done in this project- it may be in a future phase so keep it in the back of your mind.

1. Scheduler
2. GUI
3. Prepare files- unzip/rename etc.
4. Post load actions e.g. send email, send report
5. Conflicting run happening

## Table Structure

**Sample Import Job Table (C\_ImportJob)**

|  |  |
| --- | --- |
| **Import\_k** | **ImportName** |
| 1 | Provider |
| 2 | Owner |
| 3 | Covid |
| 4 | RCA HRPR |

**Sample Import Steps Table (C\_ImportSteps)**

For the parameters below I included the name for illustrative purposes. You actually don’t need it and will know based on the Action what each parameter number does

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Step\_K** | **Import\_k** | **StepOrder** | **Action** | **Parameter1** | **Parameter2** | **Paramater3** | **Parameter4** |
| 1 | 1 | 1 | Import API | Source: https://data.cms.gov/provider-data/api/1/metastore/schemas/dataset/items/4pq5-n9py | To Table: StagingTable999 | MapTable: 104 |  |
| 2 | 1 | 2 | Validation | 123 |  |  |  |
| 3 | 1 | 3 | Mapping Move | From Table: Staging5 | To Table: Prod5 | MapTable: 105 |  |
| 4 | 1 | 4 | Validation | 124 |  |  |  |
| 5 | 4 | 1 | FTP | Connection: 1 | FilesToGet:\* | Destination:c:\staging |  |
| 6 | 4 | 2 | Import delimited file | Source:File1 | Destination:StagingFile1 | Delimiter:, | MapTable: 101 |
| 7 | 4 | 3 | Import delimited file | Source:File2 | Destination:StagingFile2 | Delimiter:, | MapTable: 102 |
| 8 | 4 | 4 |  | *… here you do the import step for all files* |  |  |  |
| 9 | 4 | 5 | Validation | 125 |  |  |  |
| 10 | 4 | 6 | Mapping Move | From Table: StagingFile1 | To Table: Prod1 | MapTable: 201 |  |
| 11 | 4 | 7 | Mapping Move | From Table: StagingFile2 | To Table: Prod2 | MapTable: 202 |  |
| 12 | 4 | 8 |  | *… here you do the import step for all files* |  |  |  |
| 13 | 4 | 9 | Validation | 126 |  |  |  |

Connections Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Connection\_K** | **ConnectionName** | **Url** | **UserName** | **Password** |
| 1 | FTP1 | Sftp://ftp1 | User | guess |

Sample Mapping Table: (c\_importMapHeader)

|  |  |
| --- | --- |
| ImportMapHeader\_k | ImportMapName |
| 101 | Import5 ApI to Staging |
| 102 | Import5 Staging to Prod |

Sample Mapping Table: (c\_importMap)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Primarykey** | ImportMapHeader\_k | FromField | ToField | Order |
| 1 | 101 | Field1 | Name | 1 |
| 2 | 101 | Field2 | Telephone | 2 |
| 3 | 201 | Name | FullName | 1 |
| 4 | 201 | Telephone | phone | 2 |

## Existing Tech

The following technologies are already code you will just execute them with the right parameters etc:

* Logging- we have table structure and stored procedures to write to it that will be supplies
* Validation- we have stored procedures and view to execute sql scripts for validation. These sql rules write success and failure to tables. These tables can be used to determine what validations failed and based on their type the code here should determine if an error should be sent, process stops etc.