mi.claims - upload of data moving forward

ClaimCenter

0. Requirements

IntelliJ configured for Development: Configuring IntelliJ for Apac Development

7Zip or some other unzip tool that can handle password protected zip files.

Claim Center

1. Load Claim Center Dump

The ClaimCenter file is a .BAK file delivered as a .ZIP file via a Secure file Download link. It is downloaded to the following folder location:
S:_Group\Projects\Project MIR\5 - Claims\
A password is also received which is needed to access the ClaimCenter file. The password can be obtained from the PMO Manager.
Copy the Claim Center Dump DB to the SQL server instance below. The DB must be named ClaimCenterDump.
Local - APANZHC10DB08DV
Dev - APANZHC10DB08DV
Test - APANZHC10DB08DV
Prod - APANZHC10DB08VP

How to change database:

(Insert steps to restore DB from backup here...)

2. Check for Database

There is a test in mi.claims.test called 'processClaimCenterImport.feature'. Run this on the data. It will report any clients ids that could not be located, or policy numbers with more than one client. These are ignored by the production import process anyway (as per the requirements). If the mismatch assertion fires, the data was not loaded correctly and needs to be fixed. Note: this is run in Dev or locally

3. Run import on dev/test/prod. Instructions here: mi.claims - initial database load

4. Get the number of expected entries from either step 2 or 3. Add it to the runsheet.

MGA/Vero

1. Check the file for windows style endlines. It should be sufficient to open it in notepad. If there is not one record per line, then it does not have windows endlines. If it fails, send it back and have the vendor fix it.
2. Make sure the csv file has the same number of columns, the same columns names, and the same column data format as the original file that was loaded. Compare it with the original file. Make sure to check the claim number format carefully. If it's not the same, then either have the vendor fix it, or if it's a required change, you have to do a full database reload.
3. Copy the vero csv file to src/test/resources in the mi.claims.test project. Call it vero-prod.csv.
4. Run the processVeroImport.feature cucumber test. If there is a failure parsing the csv file, then send it back. This can be done on dev or locally. If it fails with an assertion on a mismatched client id, the data was not loaded correctly and needs to be fixed. You can examine the output. It will probably output some discrepancies in the data. These are for informational purposes and a sanity check. The requirements for the production import are to import the data as is.
5. Run the import on dev/test/prod. Instructions here: mi.claims - initial database load
6. Get the number of expected entries from either step 4 or 5 and add to the runsheet.

PULSE/NZI

1. Check the file for windows style endlines. It should be sufficient to open it in notepad. If there is not one record per line, then it does not have windows endlines. If it fails, send it back and have the vendor fix it.
2. Make sure the csv file has the same number of columns, the same columns names, and the same column data format as the original file that was loaded. Compare it with the original file. Make sure to check the claim number format carefully. If it's not the same, then either have the vendor fix it, or if it's a required change, you have to do a full database reload.
3. Copy the pulse csv file to src/test/resources in the mi.claims.test project. Call it pulse-prod.csv.
4. Run the processPulseImportPreprocess.feature cucumber test. If there is a failure parsing the csv file, then send it back. This can be done on dev or locally. If the mismatch assertion fires, it means the data did not load correctly and needs to be fixed. You can examine the output. It will probably output some discrepancies in the data. These are for informational purposes and a sanity check. The requirements for the production import are to import the data as is.

5. Run the import on dev/test/prod. Instructions here: mi.claims - initial database load
6. To get statistics for the prod upload, eg, number of new entries: Get the number of entries from step 4 or 5. Add it to the runsheet.

Full load or incremental load

It is possible to do an incremental load (only new claims are loaded), but only if the new data does not change any existing claims. If any of the new data has changes to existing claims, then you must do a full reload.

It looks like at least the vero csv file will contain updates to current claims, so it seems a full reload will be required.

If a full load is not required, then you can follow these steps to get the number of new entries:

1. Empty Dev mi.claims database (proj1dev_claims)
2. Copy prod_claims database into proj1dev_claims
3. Run the import for Dev - and note the number of new entries -> mi.claims - initial database load

To Test Locally

Run mi.claims.api on your local.

Create file structure as outlined in mule.test/src/test/resources/WORKSTATION.test.config.properties.

```
import.vero.input=c:/crombie/claims/vero/input
import.vero.processed=c:/crombie/claims/vero/processed
import.pulse.input=c:/crombie/claims/pulse/input
import.pulse.processed=c:/crombie/claims/pulse/processed
```