MeterDataLoader

Purpose

The MeterDataLoader application is a tool to process interval meter data files and load into the MeterDataDB database. The application supports industry-standard NEM12 and a fully flexible range of CSV formats. It is structured to allow the addition of extra handlers for other file types if required.

As at May 2017 the main 15- and 30-minute meter data tables contain over 200m records representing meter data for around 2100 meters.

Technical Details

This application is a SavvyPlus-developed console application written in Python.

It runs on MEL-SVR-AP-001 and is started by a scheduled task. It is designed to run continuously.

The application is located at E:\SavvyApps\MeterDataLoader\MeterDataLoader.exe and writes log files to E:\ApplicationLogs

The application writes to and reads from the MeterDataDB database. Configuration table is MeterDataLoaderJobs and file processing records are stored in MeterDataLoaderFiles

Operation

On opening, the application reads a list of loading jobs from table MeterDataLoaderJobs. Each job specifies a source folder, a handler and various parameters to pass to it, an error folder for failed files and archive folder for successfully processed files. It also specifies a retention time for successfully processed files.

During operation the application continually cycles through the following steps:

1. Refreshes the jobs list if it has been longer than the specified time interval (e.g. 60seconds) since last done
2. Performs a directory listing for each source folder if it has been longer than the specified time (e.g. 5 seconds) since last done, and identifies files matching each job's file name mask
3. Determines the priority order for processing files based on each job's priority rank
4. Processes the highest-priority file in the queue
    a. passes the file name/location to the specified handler
    b. if handler returns successfully, move file to archive folder
    c. if handler returns and error, move file to error folder
    d. write result to MeterDataLoaderFiles table
5. Purge archive files according to each job's file mask and retention time setting

Monitoring and Troubleshooting

There is no explicit monitoring on this application but performance can be checked by looking at the various Queue folders for a buildup of files.

In normal operation the application is highly reliable and re-start is not required.

For advanced trouble-shooting consult the log file, or run the application from a dos box to see error messages produced.

Application Configuration

The following parameters can be set in the config file MeterDataLoader.cfg located in the deployment folder:

Database Connection

- odbcconnectionstring: ODBC database connection string to connect to database
- retry_time_seconds: time (seconds) to wait between database connection re-tries (e.g. 5)

Refresh Times

- source_locations_refresh_seconds: time interval (seconds) between refreshing the list of source locations to load files from (e.g. 60)
- source_files_refresh_seconds: time interval (seconds) between polling the source folders to look for new files (e.g. 5)
- archive_purge_delay_minutes: time interval (minutes) at which to purge archive folders (e.g. 1)

Configuring Meter Data Loading Jobs

Jobs are managed through the MeterDataLoaderJobs table. The fields are as follows:

Field	Purpose
source_folder	The folder to watch for files to process. This should be a local folder on the server running the application as reliable access is assumed.
success_folder	The folder in which to place files after successful processing. This is also the location against which the file purging process will operate.
fail_folder	The folder in which to place files that are not successfully processed.
priority	The job priority. Jobs with lower numbers will be processed before those with higher numbers
active_flag	1 = active, 0 = inactive. Use to temporarily disable jobs without deleting configuration
filename_pattern	dos-style file name matching string e.g. *.zip
handler	text matching an entry in the table of handlers below
handler_params	Text that evaluates to a Python dict containing parameters to pass to the handler. Description/examples below
success_retention_days	Number of days to retain archived files. Files in archive older than this date will be purged. -1 indicates keep files forever
commodity	Text to indicate commodity (e.g. ELEC) for which metering data is loaded.
organisation	Text to indicate which organisation/client the loaded data is supposed to be for.

Existing Handlers

Handler	Role
spmdf_handler	Highly configurable handler to load interval meter data in just about any CSV-like format. See detail below.
move_only	File is moved from source to archive folder without further processing
unzip	Unzips the file in-place i.e. extracts the contents of the zipped file to the same location.
nem12_handler	Load meter data in NEM12 format.

spmdf_handler

This handler can be used to process a large number of CSV files and variants without modifying the original file so that ongoing meter data feeds in any CSV-like format can be accepted.

The processing steps can be summarised as follows:

> ***Raw format SPMDF format Staging table Final table***

To work with a new file format you use the flexibility of the MeterDataLoader's (and pandas' read_csv) functions to map the original format to an acceptable SPMDF format by passing appropriate parameters to the handler. This generally involves selecting, adding & renaming columns, and sometimes applying certain rules to particular columns as well as stripping off header and footer information and interpreting dates.

The requirements of the target SPMDF format are as follows:

1. Valid field names (case-sensitive) are: MeterRef, NMI, StreamRef, MeterSerialNumber, Date, Time, PeriodID, Timestamp, TimestampType, IntervalLength, Net_KWH, Net_KVARH, Exp_KWH, Imp_KWH, Exp_KVARH, Imp_KVARH, KW, KVA, MDPUpdateDateTime, QualityCode
2. IntervalLength field must exist and be populated with values of 15 or 30
3. Meter reading time must be specified as one of Timestamp+TimestampType, Date+PeriodID, or Date+Time+TimestampType
4. Meter must be specified either as MeterRef or as NMI+StreamRef
5. At least one of the following fields must be supplied: Net_KWH, Net_KVARH, Exp_KWH, Imp_KWH, Exp_KVARH, Imp_KVARH, KW, KVA
6. QualityCode must be present and contain only the following values: ,'A','E','F','I','S','X'
7. All Exp_KWH, Imp_KWH, Exp_KVARH, Imp_KVARH and KVA values must be non-negative

Parameter Name	Usage
header_end_text	Text that marks the end of the header text to be stripped away leaving data in CSV form. Useful when number of lines of header text is unknown but the header ends in a known character sequence. Optional parameter
footer_start_text	Text that marks the start of the footer text to be stripped away leaving data in CSV form. Useful when number of lines of footer text is unknown but the footer starts with a known character sequence. Optional parameter
fixed_column_vals	Creates additional columns that contain a particular fixed value. Uses include: • to specify IntervalLength when not provided but you know the resolution to expect (30 or 15) • to specify QualityCode when not provided (e.g. to assign unknown quality code 'X') • to specify the TimestampType i.e. PE (period-ending) or PS (period-starting)
truncateNMI	Truncates the MeterRef or NMI field to the first 10 characters. Use if the NMI/MeterRef has a checksum digit or other information appended.
map_col_names	Dict of source and target column names. Use to convert the supplied column names to valid SPMDF field names.
flip_signs	List of field names whose values should have their sign flipped. Use if e.g. import kwh in file is shown as a negative value.

In addition to the parameters above you can also use any of the arguments accepted by the pandas library's read_csv function. Documentation is at http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html

Examples:

handler_params value
{'parse_dates':['Timestamp'],'dayfirst':True}
{'usecols':['NMI','REPORT_DATE','REPORT_TIME','KVARH_CON','KVARH_GEN','KWH_CON','KWH_GEN'],'parse_dates':['REPORT_DATE'],'d 'REPORT_TIME':'Time', 'KVARH_CON':'Exp_KVARH', 'KVARH_GEN':'Imp_KVARH', 'KWH_CON':'Exp_KWH', 'KWH_GEN':'Imp_KWH'}}

Compiling

The application is built using Python library PyInstaller. A text file named compile_instructions.txt in the application's source code folder contains instructions to compile the application.

Compilation requires Python and a number of additional libraries to be installed. For details of libraries/versions installed see Python Libraries pag
e.

Notes

When spmdf_handler is used:

- data loads into table SPMDF_Staging

- stored proc MergeSPMDF_Staging is called by the MeterDataLoader
- MergeSPMDF_Staging loads data for each NMI-StreamRef is loaded into IMD_Staging table
- MergeIMD_Staging is then called to merge all meter data straight away into final tables

When nem12_handler is used:

- data loads into a series of staging tables NEMMDF_Staging_*
- on an hourly schedule stored proc MergeAllNEMMDF_Staging_Files runs, calling MergeNEMMDF_Staging once per source file
- by default data is loaded at NMI resolution only, but if NMI-X exists in IMD_MeterPoint then the individual meter's data will also load into the final tables. So e.g. if VAAA000123-1 exists in IMD_MeterPoint, and data is loaded for VAAA000123 with E1,E2,Q1,Q2 then the aggregate of E1E2Q1Q2 will load as VAAA000123 and the aggregate of E1Q1 will load as VAAA000123-1
- MergeNEMMDF_Staging aggregates data with previously loaded streams and loads into IMD_Staging
- MergeIMD_Staging is called to merge all meter data into final tables