Data Loader Automation tool

Contents

[1 Preface 2](#_Toc392374081)

[2 Solution overview 2](#_Toc392374082)

[2.1 Functionality 2](#_Toc392374083)

[3 Prerequisites 3](#_Toc392374084)

[4 Installation 3](#_Toc392374085)

[5 Directory structure 3](#_Toc392374086)

[5.1 Configuration directory (configs) 4](#_Toc392374087)

[5.1.1 Standard configs 5](#_Toc392374088)

[5.2 Mapping directory 7](#_Toc392374089)

[5.2.1 Data loader mapping files 7](#_Toc392374090)

[5.2.2 FART mapping files 7](#_Toc392374091)

[5.3 Dataloaderconfig directory 7](#_Toc392374092)

[5.4 Root directory 7](#_Toc392374093)

[5.4.1 Mainscript.bat 7](#_Toc392374094)

[5.4.2 Mappingscript 9](#_Toc392374095)

[5.4.3 Paramdeletescript 9](#_Toc392374096)

[5.4.4 Deletescript 9](#_Toc392374097)

[5.4.5 Other files 9](#_Toc392374098)

[6 Dentsply org migration tasks 9](#_Toc392374099)

# Preface

This tool for data loader automation is a collection of scripts which were written in large part to facilitate a large-scale data migration operation, i.e. a data export from one org, export file manipulation and subsequent import into a target org.

Besides the core functionality, the tool also contains some helper scripts which automate some tasks that were encountered during the process.

Please be aware that as such, this tool has undergone an evolution along lifecycle of the project for which it was put together, and therefore, there may be some places where e.g. scripts are not factored, parameterization is sloppy, etc. As always in such cases, the time to properly wrap and package this tool has not yet materialized ☹

# Solution overview

This tool is designed with an org-to-org data migration task in mind. Therefore, it has features to accomplish the following things easier than with the standard available tools:

* Export data flexibly and repeatably
* Remap Salesforce IDs in exported data efficiently and non-manually through the use of Excel or similar
* Import data flexibly and repeatably
* Facilitate easy deletes of erroneously imported data

The following bits of technology are used as part of the tool:

Apex Data Loader – from your SF org – please use Data Loader for API version 29.0 (inlcuded in package), since issues have been seen using Data Loader 30.0

PostgreSQL database – www.postgresql.org

Saxon XSLT parser - <http://saxon.sourceforge.net/> (JAR provided in the package)

FART (Find And Replace Text) - <http://fart-it.sourceforge.net/>

Baretail – tail utility for Windows (provided in package)

Apart from this, there is a fair bit of Windows batch scripts and not much else.

## Functionality

The main script of this tool automates the tasks below:

* Data export from an object in the source org
* Truncation of data in the postgres database
* Mass search & replace of text in the export file
* Load of the export file into a table in the database
* Replace of predefined columns in a given database table based on a mapping (so a field containing an ID of an object in the source org can be replaced with the ID of the corresponding object in the target org)
* Unload of the modified data from the database
* Insert or upsert of the data into the target org
* Update of the mapping table of the object just loaded in the database

# Prerequisites

Windows PC or VM – the ability to run Windows batch scripts is essential

Functioning Postgres installation (preferably local)

Enough disk space to hold the required data that will be migrated x3 (in principle, when running final load, data will reside in an export file, in the database and in an import file before it can be deleted if import succeeds).

# Installation

Assuming Postgres is installed, the installation consists of unzipping the provided package in directory of your choice, on a disk which satisfies the disk space requirements.

After that, the PSQLCMD variable must be updated in the various scripts (see 5.4.1 and 5.4.2), replacing this:

SET PSQLCMD="C:\Program Files\PostgreSQL\9.3\bin\psql.exe"

with whatever the correct path on your system is.

Similarly, the variable DLPATH (line 60 or thereabouts) must be updated to point to wherever the Apex Data Loader resides:

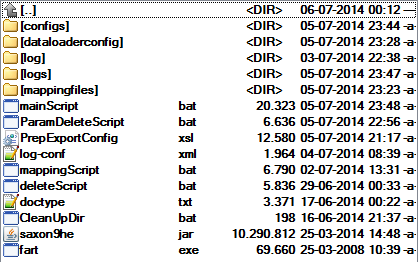
SET DLPATH=c:\Dev\ApexDataLoader

Once this is done, the setup should be complete, and the tool is ready to start work.

The last step is to import the provided database dump which contains the database structure of the database so the tool has someplace to load the required data. After the database dump is loaded, the postgres parameters should be updated as needed in the Mainscript (see 5.4.1 for details).

# Directory structure

This section will briefly outline the directory structure of the tool. It will look something like this:



The individual files and directories are briefly explained below.

## Configuration directory (configs)

This directory contains the configuration files that direct the data load. There are three distinct types of files: standard configs, mapping configs and delete configs. For convenience, a naming convention is used that implies that standard configs are files with the name format *xxx.xxx*.abc.config, where xxx are digits (which can be used for sorting and thus to dictate load order when migrating); mapping configs, since these are often prerequisites to the actual load itself, are named pre*xxx*.config; delete configs are named d*xxx*.config.

As an example, there could be several files to accomplish the actual load of Contact objects:

002.001.contacts\_bare.config  
002.001.contacts\_bare.config

In addition, there could be a delete file for contacts:

d002.001.contacts\_delete.config

And a mapping file would be named like this:

pre001.Report\_mapping.config

Each file type uses the following notation convention:

* any line starting with # is a comment
* parameters are given in the format: *paramname*:*paramvalue* – note the separator character is a colon, not an equal sign as could be expected. This is due to the parsing of parameters in Windows batch scripts. In addition, be careful with special characters:
  + any exclamation marks (!) will most likely break the script – do not use
  + if you need to have a percentage sign (%) in a parameter, it must be doubled, so e.g. a parameter like UNLOADWHERE:WHERE username LIKE ‘Peter%%’ will have two %s, which will be collapsed to one by the parsing and handed off correctly to the database.
* any line that is not blank and not a comment must parse in the context of a Windows batch file (in practice, this means that the only thing except for parameters and comments in the file can be blank lines)

The content and structure of each individual file is explained below.

### Standard configs

Standard configs are what is used to drive the actual data migration. A sample standard config file is shown below; the individual parameters are explained in comments using Cambria font it:

# Events\_bare.config

EXP:1

FARTMAP:1

TRUNC:1

LOADPGSQL:1

REMAP:1

UNLOADPGSQL:1

IMP:1

EXPMAP:0

# Export from source system

# Uses OBJECT, ENTITY, SOQL, FILENAME

# Generates/overwrites FILENAME using SOQL

#EXP:[1|0]

# Do any remaps required on local file

# Uses FILENAME, FARTMAPPING

#FARTMAP: [1|0]

# Truncate table in local database

# Uses OBJECT

#TRUNC: [1|0]

#

# Load into table in local database

# Uses OBJECT, FIELDSTRING, FILENAME

# Reads FILENAME

#LOADPGSQL: [1|0]

#

# Do any remaps required in local database

# Uses OBJECT, FIELDSTOREMAP

#REMAP: [1|0]

#

# Unload from table in local database to file

# Uses OBJECT, FIELDSTRING, MAPPEDFILENAME

# generates MAPPEDFILENAME

#UNLOADPGSQL: [1|0]

#

# Import into target system

# Uses OBJECT, ENTITY, SFMAPPINGFILE, MAPPEDFILENAME

# Loads MAPPEDFILENAME

#IMP: [1|0]

#

# Export mapping from target system into local database

# Uses OBJECT, ENTITY, MAPPINGSOQL, OLDNEWIDFILENAME

# Uses MAPPINGSOQL to generate OLDNEWIDFILENAME, then loads/updates it into mapping table in local database

#

#EXPMAP: [1|0]

# Whether the Data Loader is to use the Bulk API or not

#BULKAPI:[true|false]

BULKAPI:true

# Data Loader batch size. If Bulk API is not use, max is 200

# BATCHSIZE:[0:10000]

BATCHSIZE:10000

#Limit on number of rows for an extract operation

#LIMIT:LIMIT [0-n (integer)]

#LIMIT:LIMIT 100

#Name of the file to be output by the export operation

FILENAME:%BASEFILEDIR%Event\Event.csv

#SF Object to operate on

OBJECT:Event

#SF Entity to operate on (most likely redundant) – should be same as Object in 99,9% of cases

ENTITY:Event

#Job description string – used for logging purposes only

JOBDESC:Process Event\_bare

#Field names string – used for import into database only

FIELDSTRING:Id, Subject, ActivityDate, ActivityDateTime, DurationInMinutes, StartDateTime, IsPrivate,CurrencyIsoCode, RecordTypeId, CreatedDate, CreatedById, LastModifiedDate, LastModifiedById, OwnerId,whatid, whoid

#Field names string for export (optional) – needed when e.g. Boolean fields are output from database so that they can be remapped to a format readable by SF

FIELDSTRINGEXPORT:Id, Subject, ActivityDate, ActivityDateTime, DurationInMinutes, StartDateTime, CASE WHEN IsPrivate=false then 'false' WHEN IsPrivate=true then 'true' ELSE '' END as IsPrivate,CurrencyIsoCode, CreatedDate, CreatedById, LastModifiedDate, LastModifiedById, OwnerId, whatid, whoid, CAST(RecordTypeId AS varchar(18))

#Order by string used when exporting from SF (optional)

ORDERBY:ORDER BY Id

#SF Export SOQL

SOQL:SELECT Id, Subject, ActivityDate, ActivityDateTime, DurationInMinutes, StartDateTime, IsPrivate, CurrencyIsoCode, RecordTypeId, CreatedDate, CreatedById, LastModifiedDate, LastModifiedById, OwnerId,whatid, whoid FROM Event WHERE IsDeleted = false and isalldayevent = false and isprivate=false AND IsRecurrence=false and RecurrenceActivityId = null order by id

#WHERE-clause controlling which records are unloaded from database – can be used to not get unmappable records extracted from database for import (optional)

UNLOADWHERE:where createdbyid in (select wellspectid from mapping\_master WHERE tablekey like 'User%%') and lastmodifiedbyid in (select wellspectid from mapping\_master WHERE tablekey like 'User%%') and OwnerId in (select wellspectid from mapping\_master WHERE tablekey like 'User%%')

#SOQL query to extract mapping information – will need Id, OldId\_\_c as well as a key field – often name, but some std objects do not have a name field (e.g. Event)

MAPPINGSOQL:SELECT Id, oldid\_\_c, subject FROM Event where IsRecurrence = true OR RecurrenceActivityId = null

#Mapping file name

OLDNEWIDFILENAME:%BASEFILEDIR%Event\Event\_mapping.csv

FIELDSTOREMAP:createdbyid LastModifiedById recordtypeid ownerid whatid whoid

#Filename for the remapped file exrtracted from database

MAPPEDFILENAME:%BASEFILEDIR%Event\Event\_remapped.csv

#Data loader mapping file used for insert, update, upsert

SFMAPPINGFILE:Event.sdl

# The file where any mappings that need to be updated using FART reside

FARTMAPPING:%MAPPINGDIR%\genericmapping.config

## Mapping directory

The mapping directory contains two types of files: Data Loader mapping files (.sdl by convention) and FART mapping files (.config by convention).

### Data loader mapping files

The data loader mapping file can be generated by e.g. the Data Loader when doing a manual load. Please refer to the Data Loader manual for details.

### FART mapping files

A FART mapping file is used to parameterize a mass search & replace for a text file. It is a simple file looking like this:

01220000000D6A7AAK=012200000006VwrAAE #Administrative (U)

01220000000D6ACAA0=012200000006VwsAAE #Business (U)

01220000000CyRZAA0=012200000006VwtAAE #Contact Benelux

01220000000AIY5AAO=012200000006VwvAAE #Contact (D)

01220000000AGVpAAO=012200000006VwxAAE #Contact (D-JP)

01220000000CyRUAA0=012200000006VwuAAE #Contact Global

01220000000AHp8AAG=012200000006VwwAAE #Contact (S)

01220000000AHVjAAO=012200000006VwyAAE #Distributor Contact (DEX)

01220000000AWcOAAW=012200000006VwzAAE #Event Prospect (U)

01220000000AHVoAAO=012200000006Vx0AAE #Non-Distributor Contact (DEX)

01220000000ANagAAG=012200000006Vx1AAE #Partner Contact

The format is *text\_to\_find=text\_to\_replace\_with #everything after hash is a comment, ignored*

Thus, the above config file would try cause the script to try and do the 11 search & replace actions as specified.

## Dataloaderconfig directory

This directory is where the Data Loader’s config file (process-conf.xml) will reside. In addition, the intermediate files generated by each step of the scripted job will be placed here.

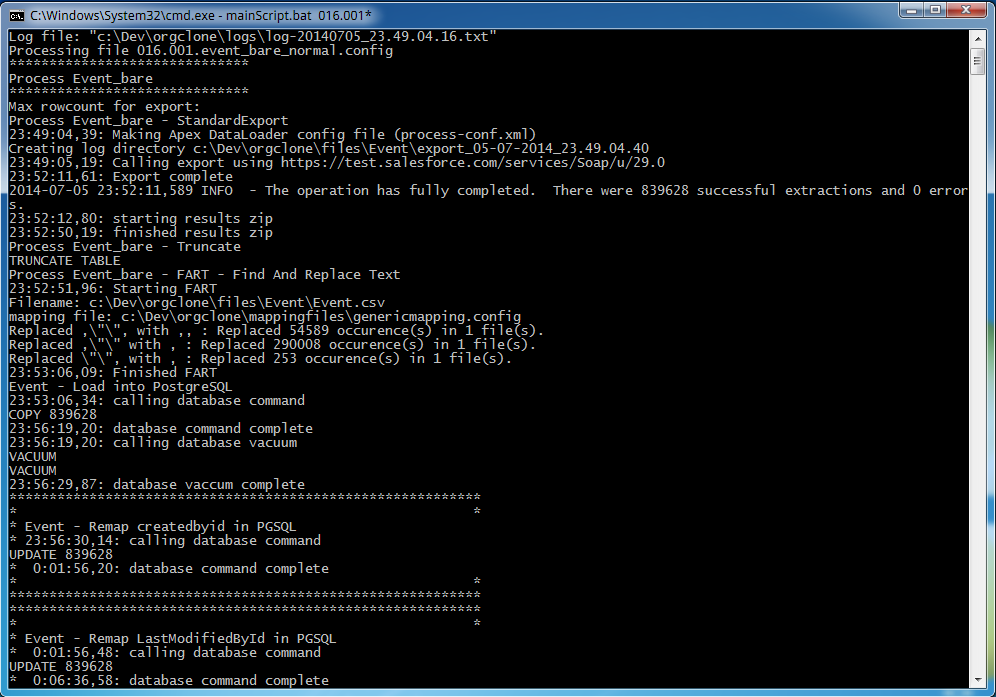
## Root directory

### Mainscript.bat

This is the main script which executes the configurations which comprise the load. It requires one parameter – a filename or filename mask which will be used against the filter directory to determine which config files to execute.

E.g. mainscript.bat 001.001.accounts\_bare\_person.config will execute the specific file mentioned (if it can be found in the configs directory). mainscript.bat 001\* will execute anything in the configs directory matching the 001\* pattern.

The mainscript will output to the console (and console only ATM) log output looking something like this:



It should show various relevant debug information, in particular row counts for data export/import operations.

Mainscript paramaters which could need changing:

* SET PGPASSWORD=14707796  
  Password for postgres
* SET DBNAME=dentsply\_fullsb  
  DB Name for postgres
* SET DBUSER=postgres  
  DB User for postgres
* SET PSQLCMD="C:\Program Files\PostgreSQL\9.3\bin\psql.exe"  
  Path to Postgres commanline client
* SET ZIP="c:\Program Files\7-Zip\7z.exe"  
  Path to 7Zip executable
* SET DLPATH=c:\Dev\ApexDataLoader  
  Path to Data Loader root (must be v. 29.0, tested and will break with newer versions)
* SET SHOWSKIPS=0  
  Parameter to cause console output to contain list of skipped steps in script
* SET READENDPOINT=https://test.salesforce.com/services/Soap/u/29.0  
  Where to read data from
* SET READUSERNAME=kgalant@dentsply.com.fullsb  
  Login to source system
* SET READUNENCPASSWORD=secret  
  Password for the login to source system
* SET WRITEENDPOINT=https://test.salesforce.com/services/Soap/u/29.0

SET WRITEUSERNAME=kgalant@wellspect.com.fullsb

SET WRITEUNENCPASSWORD=topsecret  
Same as READ… above

* SET JAVAMEM=-Xmx1000m  
  Java memory parameter which is passed to Data Loader – sometimes, Data Loader dies with a heap space error, and this parameter can be enabled/increased. However, setting it too high can intermittently cause Data Loader to fail because Java won’t be able to allocate the memory. Use with care.

### Mappingscript

To be added.

### Paramdeletescript

The paramdeletescript is a script that can be used to delete all instances of a given object from the SF system. Take care to ensure that the parameters are set correctly (in particular, both Read and write credentials & endpoint have to be set to the system to delete from).

The paramdeletescript takes one parameter: the name of the object to delete everything from, e.g. paramdeletescript Event

### Deletescript

To be added

### Other files

Various other files are present in the root directory, in particular the PrepExportConfig.xsl file which drives the generation of Data Loader config files, the log-conf.xml which controls the logging of the data loader as well as a couple of other files. Do not touch ☺