**ADP to CX Utility**

**Program**: ADP to CX utility - ADPtoCX.exe – appr. 125K

**Language**: Application written in Visual Studio, VB.NET

**Purpose**: Utility will read a csv file downloaded from ADP and look for specific additions or changes in job assignments for staff (faculty to come), to be written to the CX database via an ODBC connection.

**Details:** This is intended to run as a scheduled task nightly on a windows virtual machine, accessible for periodic maintenance. Will reside on “Teddy”.

It will require a folder for the application exe and supporting files and subfolders,

The ADP csv file can be stored there on a nightly basis. A copy of the most recent previous file will also be renamed and kept and archive folder can store prior files for as long as desired.

ADP will need credentials to send the nightly data file via SFTP the file to a suitable location.

The program will also need SMTP host details for sending email alerts (internal) in the case of errors.

**Dependencies**

Windows environment. Uses .NET Framework 4.5

Requires a Params.ini file which only contains the names of the working data files and the working directory path.

Also uses a file “Connect.xml” which contains the credentials for the ODBC connection to the database. This file will be encrypted.

A log file of all write activities to the database will be kept. An error log file will also be kept.

Size of the application and related files will initially be be about 1.5 MB. Each data file from ADP is about 1400KB.

**Deployment**

The utility does not require an install kit. It is possible to simply copy the exe and supporting folders to the working directory without running a setup program, depending on the target environment. Less overhead.

**Scheduling**

The application could run any time after the ADP file is obtained.

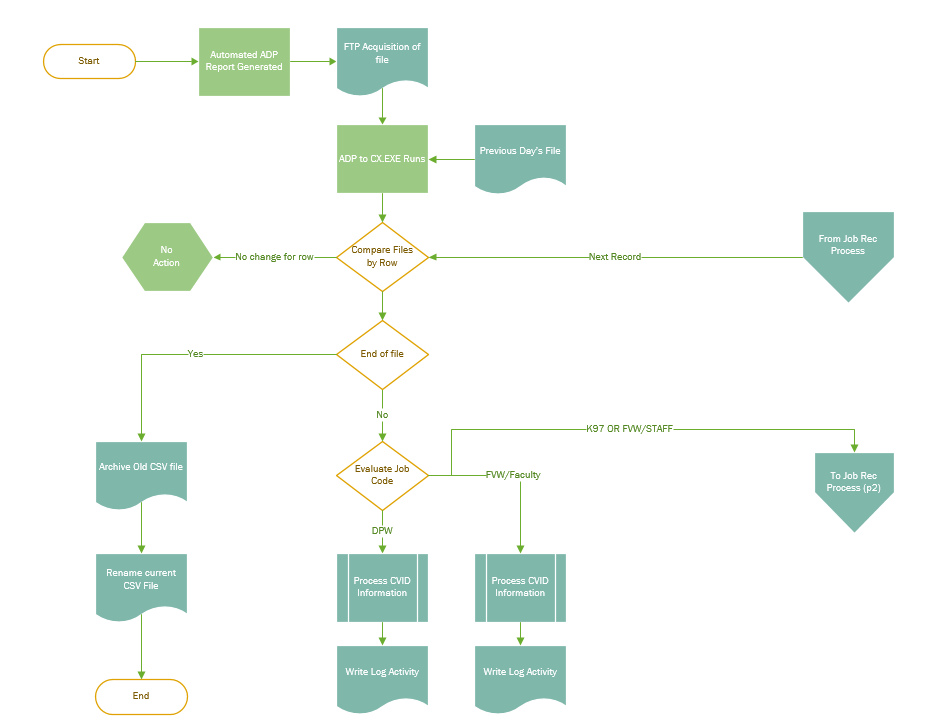
**Security precautions**

All SQL queries are parameterized to avoid SQL Injection.

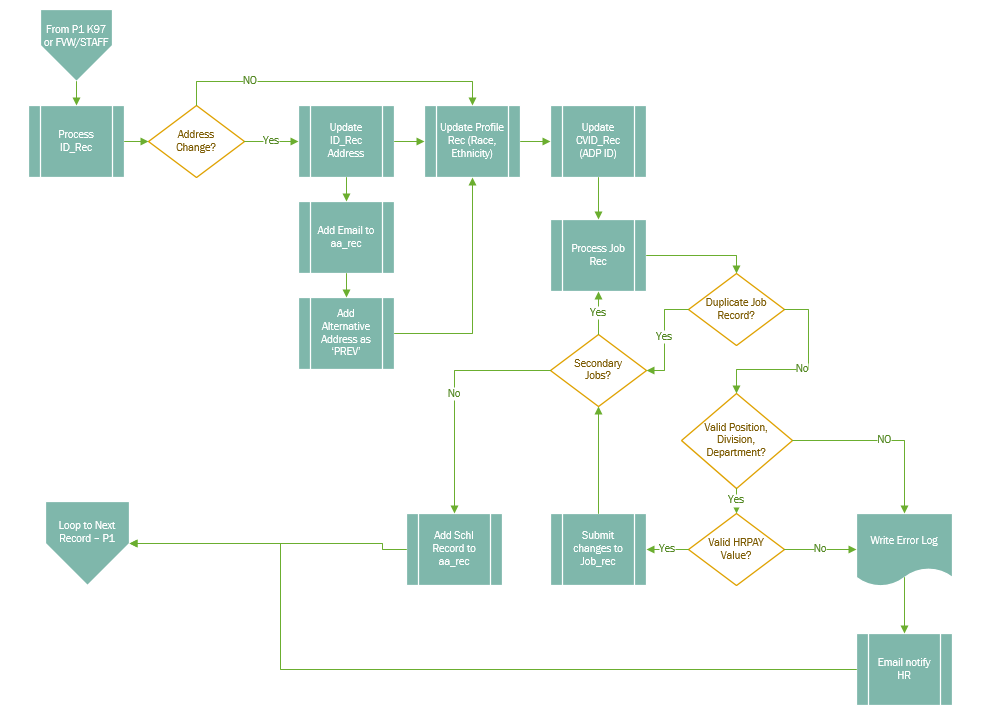
The ODBC connection string will be encrypted in a separate xml file.

**ADP to CX Logic Summary**

1. ADP will process a .csv report nightly on the ADP system. CSV file will be copied via SFTP to a location accessible to Teddy. A csv file of the previous day’s data will be in the same location. The report will not include “archived” records from ADP.
2. After the file becomes available, the exe will be triggered by a scheduled task.



1. The assumption is that the employee will have basic info already in CX.
   1. This is necessary to establish an ID number
   2. So there should be no “new employee” situation where in insert into ID\_REC is needed.
2. The program will compare the old and new files line by line using ADP File number. Because the ADP Data was modified creating multiple rows per employee (change in Pay Code), the comparison must also key on the Pay Code and PCN Code.
3. If the lines are identical, no processing will occur.
4. If lines are not identical, several Scenarios currently exist to account for.
   1. In essence “Staff” hires currently originate with HR and data is controlled there.
   2. Faculty hires are controlled by the Provost Office and much info entered directly into CX, so ADP data will be older and less accurate than what is in CX.
   3. Student hires generally have little or no accurate job related information in ADP.



1. The data in the first logic phase will be split using the job code and “home cost number” (corresponds to pcn\_aggr) field as follows.
   1. If the Job Code is **FVW** and the position from the Home Cost Number field is a faculty position:
      1. The only processing that will occur is to make sure ID info is in the CVID\_REC table.

In particular, inserting the ADP ID.SSN will be a mask only “XXX-XX-XXXX”.

* + 1. If a record exists in CVID\_REC, the data will be updated, else info will be inserted.
  1. If the Job Code is **DPW**, there is no data in ADP that is useful to CX, except the CVID information.
     1. The utility will validate and update CVID\_REC as needed
  2. If the Job Code is **K97**, or the Job Code is **FVW** and the position is Staff
     1. The utility will look at ID\_REC and update any pertinent information.
        1. If there is a change in address, the ID\_REC will be updated and a “PREV” entry will be added to the AA\_REC table
     2. The Personal Email address will also be added or updated in AA\_REC.
     3. The personal Cell Phone will be added or updated in aa\_rec table
     4. It will also look at the Profile Rec table
        1. The only profile information of value in ADP is Race, Ethnicity and possibly Sex
        2. Info will be inserted or updated as needed
     5. For staff and K97 it will make any additions or updates to CVID\_REC
        1. In particular, add of the ADP ID.
     6. The utility will finally process the Job\_Rec
        1. The key is the PCN\_AGGR field which maps to the “HOME COST NUMBER” in ADP (Position Type, Division, Department, and Position)
        2. It will validate that the POS\_TABLE has a corresponding value in the pcn\_aggr field and return the corresponding tpos\_no field. This validates that the job is an actual position in CX and assures data consistency.
        3. It will also validate that the position, division and department exist in CX.
        4. It will validate that the HRPAY\_TABLE contains a matching hrpay value
        5. It will also check to see if the employee is already assigned the job title in the job\_rec to avoid duplication
     7. After all validation, an update or insert will be performed as needed
     8. If a position is terminated, the appropriate end date and status changes will be made
     9. ADP now has three custom created areas for secondary jobs. Those will be processed essentially the same way as the main job record, with the “Rank” of 2,3,4 or 5 added as necessary.
     10. A new record with the SCHL flag will also be added to the aa\_rec table.

1. At the end of the process, the previous day’s file will be archived with a date value in the file name and the current file renamed for use the next day.
2. Log files will be kept showing the changes made
3. An error log will also be kept.
4. Email notifications will be sent for certain errors or alerts.
   1. Example: If the ADP Record has no Carthage ID, the process aborts for that record and an error message is written. The message can be sent to HR Personnel via email to correct the issue.
   2. If there is a program failure, an email will be sent to a support staff person.
   3. The email addresses for HR and Support personnel will be stored in the params.ini file.

**Maintenance and Troubleshooting**

The code will be stored at….

When running code in visual studio…

Certain repeatable processes are coded in separate class modules. Separate cls files exist corresponding to each table in the cx database. These modules do needed initialization, gets and sets of variables and handle the sql.

Sql calls are parameterized to prevent sql insertion attacks. Field names are not explicit in the sql statements but populated with parameters.

In addition, the sql building logic for updates is such that if a value is null in the ADP record, it will not overwrite the existing value in CX.

Connections to the database are not persistent. The connection is made, the sql is processed and the connection is ended and destroyed.

The program starts in module SubMain.vb.

Sub MAIN does a bit of general setup and fires off the working code.

Main Calls GetParams()

This sets certain global variables found in the params.ini in the bin/debug folder. File names, smtp info, email addresses can be set and changed in the ini file without needing to touch or recompile and code.

It also reads the connection string from connect.xml.

Global variables are defined in clsGlobals.vb

Main then calls QryCSV

QryCSV creates two virtual tables in memory by reading the csv files from the current day and the prior day.

Module clsCSV has the code to turn the csv files into vitural tables in memory. These tables can then be accessed by SQL.

The code takes each line of the new file from ADP and queries the old file data using the ADP File number key, as well as the PayCode and PCN number. The additional keys are necessary because some individuals have more than one record in the file.

If the comparison finds any difference between the two matched lines, then it proceeds to parse the differences, else it falls through the loop and moves to the next record.

The program should never create a new ID\_REC entry. Theoretically, there should never be a record in ADP that does not have a matching ID in CX.

QryCSV calls ParseADPData

This block does the main work of the program

1. If there is no CarthageID in the ADP file, then an error is thrown and the processing stops, moving to the next record. If the number exists, a query against CX validates that the number exists there.
2. A flag is set for Faculty vs staff (sFacStaffFlag).
   1. If the job code is DPW, there is nothing to be processed.
   2. If the job code is K97, it is hourly staff and is processed.
   3. If the job code is FVW, the employee may be staff or faculty, so based on the job type from the PCN\_AGG field we need to determine which. Faculty are not processed as of 7/24/18. Staff will be.
3. For K97 and FVW staff, the ID\_REC table needs to be dealt with. ClsID\_rec.vb has the sql code. An object of that class is created, variables populated and the id\_rec is updated.
   1. As part of the process, we have to determine if the change in the ADP file affects the address. Sub “CheckAddress” handles the checking of the CX address info, determines whether to update that info and whether to add a ‘PREV’ entry to the aa\_rec table.
4. For ALL employees, the CVID\_REC table needs to be dealt with. Cls\_CV\_ID.vb has the sql code. An object of that class is created, variables populated and the cvid\_rec is updated or created if necessary.
5. Separate processes are called for the personal email and cell phone fields to be added to the aa\_rec table as needed.
6. For K97 and FVW staff, the profile record is updated if necessary. clsProfile\_rec handles the sql.
7. For K97 and FVW staff, the job record is handled.
   1. A sub “process\_job\_rec” is called that does a number of steps.
      1. It checks to see if the employee already has the job recorded to avoid duplicates
      2. Goes to the pos\_table to get the tpos\_no based on the pcn\_agg info.
      3. Makes sure the hrpay field is valid
      4. Determines if the job should be ranked as 1 or 2.
      5. Updates or inserts as needed.
8. For K97 and FVW staff, additional jobs from the custom job fields are processed the same way as the original job rec.

Returns to QryCSV

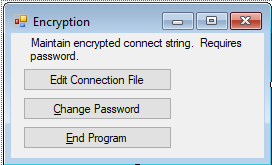
Returns to Main

Finally, yesterday’s file is archived and today’s file is renamed.

**Encrypt\_Decrypt.exe**

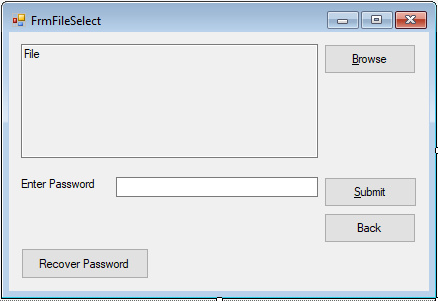
The connection string is stored in a “connection.xml” file that is encrypted.

There is a separate utility program “Encrypt\_Decrypt.exe” that can be used to decrypt the file if the connection string needs to be changed. This program uses the same encryption method as the main ADP to CX.exe utility.

The encrypt program requires a password that will be stored in an encrypted xml file (“Password.xml) as well. That password can also be decrypted and changed by running the Encrypt\_Decrypt.exe program.

In this Encryption program, a small form to select whether the admin wants to edit the connect string or password.

The program is set to store the password and connection string files in the bin\debug folder within the main project. When either the password or connection string option is chosen, the file location will display in a text box on a second form. The is an option to select another location or file for testing, but the location of the connect.xml file is static for the ADP to CX program and should not be moved for production.



The admin will need to enter a password to access the xml file and press “submit”. The xml file will first be decrypted, then will open in notepad and can be manually edited and saved. When the process finishes the file will be re-encrypted.

If the password is lost or forgotten, the “Recover Password” button will read the password file and email it to the support email stored in the params.ini file. Once recovered the password should be changed, of course.

**Folders/Files**

ADP\_to\_CX

Bin

Debug or Release subfolder

|  |  |
| --- | --- |
| ADPLog.txt | Log file |
| ADPtoCX.application | Application Manifest |
| ADPtoCX.exe | Executable |
| ADPtoCX.exe.config | Config file |
| ADPtoCX.exe.manifest | Manifest File |
| ADPtoCX.frmStatus.resources | Resources |
| ADPtoCX.pdb | Debug Database |
| ADPtoCX.vshost.exe | Application |
| ADPtoCX.vshost.exe.config | Config |
| ADPtoCX.xml | XML |
| Connect.xml | XML |
| Password.xml  ErrorLog.txt | XML  Log |
| Params.ini | Ini |
| ADPtoCX.log | Log |

Data Files

|  |  |
| --- | --- |
| ADP to CX.csv | csv |
| ADP to CX Last.csv | csv |
| Archive | subfolder |

**Future considerations**

The code can be modified to deal with faculty data if the workflow warrants.

The concern is that faculty data entered into ADP at present is not the most current information and should not overwrite what the provost office enters into CX. If Faculty hire process begins in ADP, then it would make sense to push that information to CX.

Self service is available in ADP, allowing employees to update some information, which would make it more valuable to have ADP be the initial entry point for HR related information.

Virginia has also asked whether the ADP File Number needs to correspond to the CX ID number. If not, the workflow might be streamlined further.

There are several ADP Fields designated “Home Department Code” that is a six-digit numeric field. The first three digits correspond reasonably well to the function code in the CX GL tables. This code could possibly be added to make the connection between Employee IDs and account numbers. The correct Home Department Code is the one associated with the Employee Profile in ADP.

One approach could be to pull the first three digits and insert them into the pos\_table in the func\_area field. That would potentially aid with connecting accounting issues to employees, departments and supervisors.

There is also a supervisor ID field in ADP that could help sync things up with the supervisor\_no in the job\_rec table. It is not clear if this is identical to the individual identified by the “Reports to” field in the Employee Profile. The latter is largely unpopulated as of 2/12/2018.