User Identity Automation Documentation

# Contents

Table of Contents

[Contents 1](#_Toc99094071)

[Overview 2](#_Toc99094072)

[Folder Structure 2](#_Toc99094073)

[Constants 2](#_Toc99094074)

[Student Managed Groups 3](#_Toc99094075)

[Student Managed Attributes 3](#_Toc99094076)

[Application Flow 4](#_Toc99094077)

[Job Execution 4](#_Toc99094078)

[System Requirements 5](#_Toc99094079)

[Database 5](#_Toc99094080)

[Running the Application 5](#_Toc99094081)

[Logging 5](#_Toc99094082)

[Committing Changes to the Code 5](#_Toc99094083)

[Troubleshooting Note 5](#_Toc99094084)

# Overview

The purpose of this script is to automate the creation, modification and deletion of user accounts in FFCA. The script pulls data directly from Edsembli and drops data into a SQL staging database. The data is then compared to what is Active Directory and anything that is out of date is created, modified or deleted.

# Folder Structure

**Identity\_automation**

**active\_directory\_annual\_jobs** - contains the jobs that are ran on an annual basis and are only executed with a switch option when running the script

**active\_directory\_jobs** - contains the jobs that are ran to modify active directory (ran in alphabetical order and require a run function as the entry point

**common** - contains the constants.py which holds all the application variables

**edsembli\_jobs** - contains the jobs that are ran to export data from Edsembli (ran in alphabetical order and require a run function as the entry point

**lib** - contains the drivers to work directly with the various services (Edsembli API, Active Directory, Email, etc)

**models** - object models that hold the data of an object (such as a user) and initiates various tasks on that object

**logs** - log of what is executed. Logs are appended on each run with a new file for each day

**main.py** - application entry point

**requirements.txt** - list of modules used in the application (referenced in the Dockerfile)

**sample\_xml\_extracts** - sample Edsembli extracts (with real data)

# Constants

|  |  |
| --- | --- |
| job\_collections | List of jobs (folders) that will be ran |
| edsembli\_soap\_url | Edsembli SOAP API Url path |
| edsembli\_sql\_connection | SQL Connection to Edsembli Staging Database |
| ldap\_server | LDAP server DNS or IP |
| ldap\_username | DN of service account with domain admin access |
| ldap\_password | Service account password |
| ldap\_search\_base | Default search base |
| ldap\_managed\_groups | List of managed groups (exclude school and grad year groups) |
| ldap\_student\_domain | Student email domain |
| ldap\_student\_ou\_base | Base location for student accounts |
| ldap\_student\_pwd\_change\_new | Force password change on first login |
| logging\_log\_level | Logging level |
| logging\_console\_format | Console logging format |
| logging\_file\_format | File logging format |
| logging\_file\_time\_format | Logging time format |
| logging\_file | Application log location |
|  |  |
| execute\_changes | Simulate = False, Execute = True |
| execute\_moving\_past\_users | Don’t Move = False, Move = True <-- for withdrawn users |
| execute\_disabling\_past\_users | Don’t Disable = False, Disable = True <-- for withdrawn users |

# Student Managed Groups

Student-All

Password Policy K-4

Password Policy 5-12

365 A1 Student License

365 A3 Student License

Student-<School Initials>

Student-<Grad Year>

Note: this list can be adjusted in the constants.py file

# Student Managed Attributes

|  |  |
| --- | --- |
| cn | Username (first part of email address) |
| company | Grad Year |
| department | School Short Name |
| description | Grade Level |
| display Name | First Name + Last Name |
| employeeID | Edsembli Student Code |
| employeeNumber | Ministry ASN Number |
| employeeType | “Student” |
| givenName | Usual (First) Name |
| info | Ministry ASN Number |
| mail | Email Address |
| sAMAccountName | Username (first part of email address) |
| sn | Last Name |
| Title | Student |
| userPrincipalName | Email Address |

# Application Flow



# Job Execution

Jobs are sorted into collections (folders) in the identity\_automation folder

It will try to execute every file that ends with ‘.py’ and does not start with ‘\_\_’ (double underscore). The double underscore can be used to not run specific jobs. Jobs in the folder are ran in alphabetical order. That order could be manipulated with prefixes if needed.

Every job file needs to have a function called run(). This is the entry point into the job.

# System Requirements

It requires Python 3.9. Some of its decencies don’t support Python 3.10

Running the line below from the Identify Automation folder will install all the required decencies

pip install -r requirements.txt

# Database

Data is stored on the sysdb02.ffca.local in the Edsembli database.

It only uses the Schools and Student Demographics tables as of right now.

Data is populated from the Edsembli\_job jobs collection (filename loosely matches table name)

# Running the Application

From application folder

Python.exe ./main.py

For testing it’s best ran from Visual Studio Code. Open the Identify Automation folder within VC Code and it will run.

# Logging

The logs are written to the screen and to file. They can be found in ./identiy\_automation/logs

The logs roll over every day. In the file version of the log, they are timestamped with file path of where the logging happened for extra information. There is no automatic log clean up.

# Committing Changes to the Code

The source code is stored in Azure DevOps. The application on the systech03 server is locked and be modified there (nor should it be because any changes would be overwritten on the next commit – all files are over written every release).

<https://dev.azure.com/ffca-calgary/_git/Identity%20Automation>

Changes can be made in the Azure DevOps however the preferred method is to open the git repository with Microsoft Visual Studio Code. Changes committed to the main branch are automatically pushed to production as a release.

Troubleshooting Note

* After systech03 has been restarted the Azure Pipeline Agents don’t always start on their own. They can be started in the Services.msc app.