Enterprise Sync Tool

Summary

In order to synchronize user, and group information between Officevibe and customers systems based on custom, on-premise, or otherwise not directly integrated HRMS, a simple script called the Enterprise Sync Tool has been built. This script, built using Powershell technology, will consume a set of three CSV files and push them to Officevibe through a secure HTTP connection and authenticated through an Officevibe API token. The process can be run manually, but is engineered to enable automation of the user synchronization process.

Contents of the package

- config.json
- sync.ps1
- Data
    - users.csv
    - groups.csv
    - groups-mapping.csv
- Modules
    - Officevibe.PS.SyncSDK.psd1
    - Officevibe.PS.SyncSDK.psm1

Usage

The following documentation assumes an on-premise setup with a custom HRMS, where we want to setup a recurring weekly job to push the updated state to Officevibe with no user interaction.

Step 1: Create and Configure the Environment

In order to run the script, we will need an instance of Windows that can execute a Powershell script and can reach the app.officevibe.com domain on the secure HTTP (443) port. You can extract OV_EnterpriseSyncTool.zip to a folder of your linking. We will assume C:\OV_Sync.

Step 2: Configure the Enterprise Sync Tool

Configuration of the synchronization process is done through the config.json file at the root folder of the tool (C:\OV_Sync\config.json). Looking at the file the important parameters are the following:

- ApiKey: This field is used to authenticate the request. If you don't know how to create an API key, you can check with the Officevibe team to learn how to create one.
- SyncManagerEmail: When the synchronization process is executed, it is associated with an admin account. As such, you need to provide the user email of a user with **admin** rights.
- InviteNewUsers: When set to true, this flag will automate the process of sending out the user invites to use Officevibe, instead of having to do it through the dashboard. This is useful for incremental operations where new employees or changes should be handled automatically, but should be used with caution, as setting it to true can potentially send hundreds or thousands of invites on initial imports.
- ApiUrl: This is used to target the environment on which the data will be imported. Other than https://app.officevibe.com you can have access to a sandbox environment to test and validate your data before importing it to the real environment. If you do not have a sandbox environment and would lik to have one just contact the Officevibe team.
- In addition, if you wish to put the three files mentioned below in a different place, or with a different name, you can also set the appropriate values here.

Step 3: Export Data from the HRMS System to the three CSV files

If you open the three sample files provided in the Data folder, you will see CSV files. These files use a CSV format, separated by comma, escaped with double quotes, with a header line, encoded in UTF-8. They contain the following information:

- **users.csv:** This file contains basic user information. The key identifier for Officevibe is the email address. This is also where you can set fields like first name, last name, job title and language. Creating new users is done by adding a line, and removing a user is done by removing a line.
- **groups.csv:** This file contains basic group definitions, with an id and a name. This allows for renaming a group by changing its name while preserving its id.
- **groups-mapping.csv:** This file will associate users to groups, and control additional group settings like who is a group manager. It is also possible to map subgroups to groups.

As such, you will need to generate these three files from your HRMS and drop the resulting files in the location you configured in config.json.

Step 4: Run the Script Though a Scheduled Task

Once you are satisfied with your files, then you are ready to push them to Officevibe. To try the process manually, simply open a Powershell window and execute the following:

```
./sync.ps1 -ConfigPath ./config.json [-Verbose] [-DumpFormattedRequest] [-IgnoreEncoding]
```

As specified above, it is possible to add -Verbose and -DumpFormattedRequest to the command for debugging purposes. Verbose will display additionnal information while executing the command and DumpFormattedRequest will generate a dump.json file at the script location containing the json request that will be sent to the API.

The -IgnoreEncoding flag was added to bypass encoding validation when script users are confident of their file's encoding. f this flag is not present, the files will be validated against the UTF-8 with BOM encoding.

To automate the process, simply execute the same script through a Scheduled Windows Task.