Import Automation – Engage

Last updated: 2017-10-26

The following documentation pertains to Campus Labs Import Automation. For questions concerning this process, please contact your Campus Labs representative or email support@campuslabs.com.

Contents

Overview

Institutions with the appropriate license for Engage may utilize a process known as *Import Automation* for automating the transfer and import of “flat-files” (i.e., comma-separated or “CSV” spreadsheets containing data). Import Automation is primarily intended for institutions that wish to import or update user/student information into select Campus Labs products on a recurring basis.

Requirements

The process of utilizing Import Automation requires the following steps:

1. An individual, appropriately-designated, by an institutional contact known to Campus Labs must provide an electronic or verbal communication to a Campus Labs consultant requesting the creation of an account on the Campus Labs secure file server. The following details should be included in the account request:

    a. Contact information for the individual who should receive server credentials, including:

        i. First name
        ii. Last name
        iii. Title/position
        iv. Campus email address
        v. Office telephone number

    b. The frequency with which the institution intends to send data TO the Campus Labs secure file server

    c. The public IP address(es) of the server(s) FROM which the institution intends to send the data

2. For security purposes, the contact provided by the institution must be able and willing to receive the credentials for establishing a secure connection to the file server via telephone. Campus Labs will *not* send these credentials via electronic or postal mail.

3. The contact or a designated campus IT professional should use the credential information to attempt a connection to the Campus Labs secure file server. In the event that a connection cannot be established, a request for trouble-shooting can be sent to the Campus Labs Support Team (support@campuslabs.com).

How it Works

The process of utilizing Import Automation is simple: a campus transfers files to a secure server operated by Campus Labs and Engage automatically processes all such files, imports any eligible data, and finally creates a summarization of the import process so that institutions can retrieve and verify the results.

The Import Automation process includes the following steps:

Directory Structure

1. Once a secure file server connection has been successfully established, an institution will see a folder hierarchy similar to the below:

`\{account}`	← Root folder where `{account}` is your school name/acronym
`\{account}\CollegiateLink\users`	← Users folder for importing users into CollegiateLink
`\{account}\CollegiateLink\users\errors`	← Error folder for imports of users into CollegiateLink
`\{account}\CollegiateLink\users\results`	← Results folder for imports of users into CollegiateLink

2. An institution will have WRITE access to all directories but will not be able to alter existing directories or create new ones.

File Transmission

3. It will be the responsibility of the institution to place bulk user import files* into the appropriate directory through whatever method the institution wishes, including manual upload or an automatic/scripted transmission process.

   a. Transmission must occur over secure shell file transfer protocol (i.e., SSH-based FTP, a.k.a. "SFTP").

   b. The directory structure is a key part of the Import Automation process. Without it, the system will not know the appropriate action to take upon a given file. As is such, if an institution inadvertently places its files outside of the appropriate folder (e.g., creates their own, or transmits a file into the "errors" or "results" folders) then an error will occur (see *Section 5.c.iii*).

** Note: Though any number of import files may be placed into the directory for processing, institutions are encouraged to consolidate their data into the smallest appropriate number of files.*

Transmission Manifest

4. The institution must place a file of any name with the extension ".done" into the appropriate directory in order to signify that the transmission of all files from the institution is complete and that import of any files referenced inside the manifest should begin. The ".done" file is an XML document that must conform to the following layout:

   Example file name (file name can be anything you want as long as it ends in *.done*):

```
nightlyImportFiles.done
```

   Example file contents (note that XML is case-sensitive so you must use lower-case tag names):

```
<?xml version="1.0" encoding="UTF-8"?>   ← Correct   <?XML Version="1.0" Encoding="UTF-8"?>   ← Incorrect
<files>
   <file name="nightlyImport1.csv" checksum="79ac8d043dc8739f661c45cc33fc07ac" checksumhashtype="MD5" />
   <file name="nightlyImport2.csv" checksum="64aebe286e2ee35069b1850a75ebe9f6" checksumhashtype="MD5" />
   <file name="nightlyImport3.csv" checksum="f8089e32d4188dee36a2b111dcc216f7" checksumhashtype="MD5" />
</files>
```

a. The .done file may contain as many <file…> lines as appropriate, but there must be at least one

b. The optional checksum value for each <file…> line must either be an MD5 checksum hash/string* or completely blank.

The option to leave this value blank is there for the convenience of institutions that do not have the technical capability to generate a checksum. It is recommended that a checksum be included however, to increase the integrity of the verification process which is meant to guard against the chance that data file corruption occurred in the process of transmitting the data file to the secure file server.

c. The optional checksum hash type value must be "md5" (n.b., letter case of this string does not matter) if a checksum is included in the appropriate attribute of the <file…> line. If a checksum is not included, then the checksum hash type value must be set to "none" (n.b., letter case of this string does not matter) or simply left empty (e.g., *<file name="foo.csv" checksum="" checksumhashtype="" />*)

** Note: MD5 checksums for files can be generated through native commands in Unix/Linux/MacOS, common programming languages such as Perl, PHP, or C#, and on the Windows environment through add-on utilities such as the Microsoft File Checksum Integrity Verifier (http://support.microsoft.com/kb/841290).*

d. Our FTP server is Windows-based. It is possible for the line endings of a text tile to be changed if the file transfer used to our server was set to use ASCII mode instead of binary. When that's done, and you FTP a file from a Unix-based system to Windows, the line endings are changed to make it readable. If you use the binary transfer method, this doesn't occur. Some FTP applications will make the change for you, but not all. If you want to use a checksum, be sure to transfer your files in binary mode, not ASCII. How to do so will be different for every application, so you need to review your application's documentation for how to do so.

Manifest Validation

5. When the manifest file is written into the appropriate directory the system will examine the file and attempt to perform numerous validations on its contents. This validation process involves the following steps:

   a. First, the system renames the .done file to "{filename}.PROCESSING" so that it is clear that Import Automation has begun. If the system cannot rename the file it will make a second attempt and, upon further failure, move on to any additional .done files in the directory.

   b. Next, the system will read the contents of the "{filename}.PROCESSING" file and ensure that the structure of the XML is valid. If the XML is invalid, the file will be renamed to "{filename}.INVALIDSCHEMA".

   c. If the structure of the file is determined to be valid, the system will begin validating the references within each <file…> line. This reference validation includes the following steps, which will be repeated for each referenced file:

i. The system will verify a file matching the name listed exists in the directory. If a matching file is not found an “INVALID” attribute will be added to the <file…> line indicating that the reference is invalid and the manifest file will be renamed to “{filename}.CONTAINSINVALIDFILES”.

ii. If a matching file is found in the directory, the system will generate an MD5 checksum for the file and compare it to the checksum listed in the <file…> line. If the two checksums do not match an “INVALID” attribute will be added to the <file…> line indicating that the reference is invalid and the manifest file will be renamed to “{filename}.CONTAINSINVALIDFILES”.

iii. If the two checksums match, the system will check the configuration information managed by Campus Labs, which defines what actions should be taken on the file given the directory it was placed into. If configuration information is not available or is inaccessible (i.e., if a database access issue occurs) the manifest file will be renamed to “{filename}.IMPORTCONFIGMISSING”. If this occurs, please contact support@campuslabs.com for further information.

iv. If a valid configuration for your institution’s import is found, the system will proceed with the data processing steps required for importing the records contained in the actual data file.

Data Processing

When a single <file…> line within the manifest file has been validated, data processing will occur. Data processing involves the following steps:

1. The system will copy the reference file to a location outside the directory to which it was transmitted so that further action can occur without concern for the file being over-written during data processing.

2. During the import process, the system will track the results of the action performed for each record, and will produce the following four output files:

   a. A list of all records (showing Username only) that produced an error, along with the corresponding error details, will be written into an “errors.csv” file under the “errors” folder of the directory into which the data file was transmitted. This file will be overwritten each time processing occurs, so that it always represents the errors that resulted from the most recent occurrence of import automation.

   ```
   \{account}\users\CollegiateLink\users\errors.csv
   ```

   b. A duplicate copy of the error file above will be written into the “errors” folder of the directory into which the data file was transmitted. The file’s name will be a timestamp of when the file was created. This copy will act as the archival record of errors since the aforementioned “errors.csv” will only contain information on the most recent occurrence of import automation.

   ```
   \{account}\users\CollegiateLink\errors\YYYY-MM-DD_HH:MM:SS.csv
   ```

c. A summarization of the import process will be written into a file named “results.txt” under the “results” folder of the directory into which the data file was transmitted. This file will be overwritten each time processing occurs, so that it always represents the latest result summary from the most recent occurrence of import automation.

```
\{account}\users\CollegiateLink\users\results.txt
```

The information contained in the summarization will be:

```
{file name}
Submitted on {date} {time} by {name of institution’s file transfer account}
Completed on {date} {time}
Total rows: #
Success rows: #
Error message: {error message, if any}
Error rows: #
```

d. A duplicate copy of the summarization file above will be written into the “results” folder of the directory into which the data file was transmitted. The file’s name will be a timestamp of when the file was created. This copy will act as the archival record of summaries since the aforementioned “results.txt” will only contain information on the most recent occurrence of import automation.

```
\{account}\users\CollegiateLink\results\YYYY-MM-DD_HH:MM:SS.txt
```

3. For security purposes, after processing is complete the data file will be permanently removed from the directory into which it was transmitted. The manifest file will be removed as well, but only if none of the file references included in it contained errors that an institution would need to see.