Avature External Import Services

About This Document

Purpose

This document is meant to provide a comprehensive reference for building clients for Avature's Import Services. By the end of reading this document, you should be able to import records into Avature in a remote batch fashion.

Audience

This document is aimed at the Import Services client application developer.

Uses of the Service

Avature's Import Services can be used for the creation or update of records (people, companies and cases), datasets and case templates. This is achieved by extracting data from CSV files that are sent periodically from an external system to Avature. The CSV files do not need to be created before setting up the import service, but the layout of all CSV files for a particular import service must be exactly the same.

The following is a list of fields that can be populated and actions that can be performed on records, datasets and case templates by importing data through Import Services.

Person records:

- Populate built-in fields, such as first name, work phone, etc.
- Populate custom fields
- Populate form fields
- Populate record tables
- Attach files that are hosted online by including the file URL
- Include Base64-encoded files
- Link the person to a case, at a specific workflow step
- Update the person's workflow step

Company records:

- Populate built-in fields, such as work street, work email, etc.
- Populate custom fields
- Populate form fields
- Populate record tables
- Attach files that are hosted online by including the file URL
- Include Base64-encoded files

Case records:

- Populate built-in fields, such as case name, ref#, etc.
- Populate custom fields
- Populate form fields
- Populate record tables
- Attach files that are hosted online by including the file URL
- Include Base64-encoded files
- Link the case to a specific case workflow
- Update the case's workflow step

Datasets:

- Fill datasets

Case Templates:

- Populate built-in fields, such as work title, ref#, etc.
- Populate custom fields
- Populate record tables

How Does It Work?

On the Avature settings page of Import Data there is an option to create new import services. For each newly created service, the system will generate an end-point URL to be used by external client applications. For example:
https://instancename.avature.net/ImportService/q9tTSBpKjiU1ejMTHaISY2dyd0gfQAYZ/

Applications connecting to this address are expected to provide information as 'comma separated values' (a.k.a. CSV). When a new service is created, the layout for the CSV to be submitted has to be defined. This involves:

1. Indicating whether the first line in the file represents the header (name of the "columns") or not (keep in mind, though, that it is recommended not to include a header, as it won't be used for any purpose).
2. Mapping each "column" to a field of the entity (case, person, company, etc.) being imported (e.g.: "column 3 contains the person’s address").
3. Defining the "column" delimiter (most commonly the 'comma' symbol, but 'semicolon' or 'tab' may be used instead).
4. Defining the content encoding ('UTF-8' is recommended).
5. Defining an optional callback URL, which is used by Avature to post extra info on the operation results. Avature recommends that you define a complete URL (base URL + path + parameters) when creating the service. But if you need to, you can define only a base URL at the moment of creating the service and add the path and parameters through your POST requests.

Consuming the Service

As mentioned before, when a new service is created the following must be defined:

1. The CSV layout
2. The service end-point URL

Information has to be sent to Avature as a POST request. The actual import will be done asynchronically, but a response in the form of JSON content will be returned immediately.

Request

In order to push information through the service, you should execute a POST request encoded as `multipart/form-data`. You must use SSL to access the URL.

Request Parameters

Parameter	Type	Description
upload	file	(Mandatory) A file upload, as if using `<input type="file" name="upload">`. The CSV file content must follow the layout mentioned before.
returnErrorCsv	string	(Optional) Include this parameter to receive a CSV in the callback response detailing all the import errors. To receive a binary file compressed with ZLIB, set this parameter to “true”. To receive a base64 encoded file compressed with ZLIB, set this parameter to “base64”. The URL to which the callback should be made must be configured in Avature from **Settings > Import Data > Setup external import services**. If you can't access the import service settings page, please provide Avature with the URL to have it configured for your import service.

entityPropertiesToReturn	string	"ID" will return the primary key of the entity(ies) being created or updated. For example, if the import service is creating a new person record in Avature, the callback will contain the personID. For each field that is not an ID, indicate "schemaField_<nr1>_<nr2>_<nr3>" where: • nr1 = Schema spec ID • nr2 = Field number • nr3 = EntityID (1 for company; 2 for person; 3 for case; 4 for case-person relationship) Please contact Avature to learn about the fields that can be retrieved and their ID numbers.
extraPath	string	(Optional) Include this parameter in your request if you only defined a base callback URL when you created the service. Please keep in mind that this string should not start with a slash (/), because the base URL already contains one.
queryParams	string	(Optional) Include this parameter in your request if you need to add parameters to the callback URL

Request Body Sample

```
Content-Type: multipart/form-data; boundary=-----509726422166570012017671216 93
Content-Length: 451
-------509726422166570012017671216 93
Content-Disposition: form-data; name="upload"; filename="people.csv"
Content-Type: text/csv
john,carmack,233 oak street
miriam,laurie,783 greek avenue
-------509726422166570012017671216 93
Content-Disposition: form-data; name="returnErrorCsv"
true
-------509726422166570012017671216 93
Content-Disposition: form-data; name="entityPropertiesToReturn"
id,schemaField_<nr>_<nr>_<nr>,schemaField_<nr>_<nr>_<nr>
-------509726422166570012017671216 93
Content-Disposition: form-data; name="extraPath"
operation/results
-------509726422166570012017671216 93
Content-Disposition: form-data; name="queryParams"
id1=<value 1>&id2=<value2>
-------509726422166570012017671216 93
```

Service Security

The random code in the service URL is the first default security measure:

Authentication

The second default security measure is an API key - a series of random characters (including letters, numbers and symbols) - that is generated during the configuration of the import service. In order to consume the service, you need to add a header to the request, using "X-Avature-Key" as the header name and including the API key provided by your Avature Consultant as the header value. See this example of a header with an API key:

```
X-Avature-Key: QCb_JvPoWwlhWNGjFO-q1KxeJcN8EdNthKu3kTOc
```

API keys cannot be edited and, after their generation, they are no longer visible to the Avature Consultant. If you need Avature to re-send you or check the API key, a new one will be generated for the service.

At your request, your Avature Consultant can also add other security measures:

URL Expiration

Avature can configure an expiration date of your choice for the import service URL. When the set day comes, the URL is automatically set as inactive and can no longer be used.

IP Restriction

You can ask Avature to authorize one or more IP ranges as the only IPs allowed to post information to the service URL, or to prevent one or more IP ranges from posting information to the service URL.

Response

The response consists of a JSON structure detailing the operation results. Avature may not always accept a POST, so you should check the result field.

Response Values

All properties belong to the root JSON object:

Property name	Type	Description
result	string	Indicates operation success or failure. The only possible values are "accepted" or "error".
error	string	Error message only on failure, otherwise null.
retryLater	boolean	If result="error", retry only if retryLater=true. Avature may not always be ready to receive an import.
importCode	integer	ID of the import process. It can be used to check the import status manually. See Import Process Status Request below for more information.

Response Body Samples

A successful operation:

```
{
   "result":"accepted",
   "error":null,
   "retryLater":false
   "importCode":1
}
```

A failed operation:

```
{
   "result":"error",
   "error":"Upload file name must end in .csv and be a CSV file",
   "retryLater":false
}
```

Callback

When an import is completed, a POST can be done to the callback URL with the following information about the processing of the

file. The POST `Content-Type` is `multipart/form-data` and contains the following parameters:

Property name	Type	Description
result	string	The only possible values are "error" or "ok".
processedCount	integer	Number of records that were processed - it counts warnings, errors and successfully processed records.
successfulCount	integer	Number of records that were imported successfully.
warningCount	integer	Number of records that were imported with warnings.
failedCount	integer	Number of records with errors in the CSV file.
error	string	This field is not related to failedCount. It is always returned empty unless there is an error that is severe enough to stop the processing of the CSV file. If this happens, this field contains the description of the error message.
errorCSV	file	In case of an error and if the *returnErrorCsv* parameter was set to "true" or "base64" in the request, the callback will contain a CSV file that explains the errors and the CSV rows that couldn't be processed.
importerProcessId	integer	ID of the import process. It's the same value named *importCode* in the synchronous response.
entityProperties	JSON	This field is only returned if the *entityPropertiesToReturn* parameter was set with a list of fields. The value of each of those fields for the created or updated records is returned as an array of JSON objects.

Callback Sample

In the following sample, out of five (5) records processed, two (2) were successfully created or updated and three (3) failed and returned errors. The errors are specified in the *errorCSV* file.

When the request was sent, three fields were set in *entityPropertiesToReturn*. For each record that was successfully created or updated, the fields are returned in *entityProperties*.

```
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="result"
error
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="processedCount"
5
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="successfulCount"
2
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="warningCount"
0
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="failedCount"
3
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="error"

------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="importerProcessId"
380
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="entityProperties"
[{"id":5,"schemaField_6_2_3":null,"schemaField_5_2_2":null},
{"id":31,"schemaField_6_2_3":null,"schemaField_5_2_2":null}]
------------------------------b3e3bd7c6cc8
Content-Disposition: form-data; name="errorCsv"
[base64 or binary encoded ZLIB compressed file]
```

```
------------------------------b3e3bd7c6cc8--
```

errorCSV

The relevant columns of the errorCSV file are "Avature ID (for deduping)" and "Reason". "Avature ID (for deduping)" shows the ID of the row (in the CSV file sent to the Import Service) that had errors. "Reason" shows a description of all the columns that had errors for that row of the CSV file sent to the Import Service.

errorCSV Sample

```
"Avature ID (for deduping)","Custom Field: Job Type",Reason
275,,"Custom Field: invalid value 'New Position' for field 'Job Opening Type'. Custom Field:
invalid value '6040' for field 'Department Code (correct) -- changed to name'."
304,,"Custom Field: invalid value 'New Position' for field 'Job Opening Type'. Custom Field:
invalid value '7200' for field 'Department Code (correct) -- changed to name'."
277,"Feature Modeling-IC2","Custom Field: invalid value 'Feature Modeling-IC2' for field 'Job
Type'. Custom Field: invalid value 'Feature Modeling-IC2' for field 'Job Type'."
```

Import Process Status Request

As an alternative to the Callback, you can check the status of the import manually by accessing the following URL:
https://instancename.avature.net/ImporterService/<random code>/status/ <importCode returned in the response>/

The service returns the following parameters:

Property name	Type	Description
status	string	Indicates the current status of the import process. The only possible values are "pending" (which means the request is currently queued to be processed), "importing" (which means the CSV file is being processed), or "completed" (which means the request has been processed).
result	string	The only possible values are "error" or "ok".
processedCount	integer	Number of records that were processed - it counts warnings, errors and successfully processed records.
successfulCount	integer	Number of records that were imported successfully.
warningCount	integer	Number of records that were imported with warnings.
failedCount	integer	Number of records with errors in the CSV file.
error	string	This field is not related to failedCount. The value in this field is always "null" unless there is an error that is severe enough to stop the processing of the CSV file. If this happens, this field contains the description of the error message.
importerProcessId	integer	ID of the import process.
importerProcessId	integer	ID of the import process. It's the same value named *importCode* in the synchronous response.
entityProperties	JSON	This field is only sent if the *entityPropertiesToReturn* parameter was set with a list of fields. Currently, it always has an empty array ("[]").
errorCSV	file	In case of an error and if the *returnErrorCsv* parameter was set to "true" or "base64" in the request, the callback will contain a CSV file that explains the errors and the CSV rows that couldn't be processed.

Result Request Sample

In the following sample, the request was queued for processing:

```
{
    "status": "pending",
    "result": "ok",
    "processedCount": 0,
    "successfulCount": 0,
    "warningCount": 0,
```

```
    "failedCount": 0,
    "error": null,
    "importerProcessId": 29,
    "entityProperties": "[]"
}
```


In the sample below, out of five (5) records processed, two (2) were successfully created or updated and three (3) failed and returned errors. The errors are specified in the *errorCsv* file.

When the request was sent, three fields were set in *entityPropertiesToReturn*. *entityProperties* always returns an empty array. If you need the list of fields, use the Callback feature described before.


```
{
    "status": "completed",
    "result": "error",
    "processedCount": 5,
    "successfulCount": 2,
    "warningCount": 0,
    "failedCount": 3,
    "error": null,
    "importerProcessId": 29,
    "entityProperties": "[]",
    "errorCsv": [base64 or binary encoded ZLIB compressed file]
}
```