AcrossHotfolder

Category	Import, Export
Purpose	Enables integration with Across hotfolder feature.
Bootstrap Class	-
Type	Private plug-in

Description

A translate5 private plug-in that is able to watch a hotfolder for tasks, that should be created in translate5 - and re-exported to the hotfolder, once they are ready.

The hot-folder will be queried every ~15 min with the usual translate5 cron job.

The plug-in is able to watch multiple hotfolders (for different connected systems each) at the same time. This is implemented by configuring one SFTP server per instance, overwriteable on customer level.

All tasks created out of files loaded via a customer specific config are associated to that customer.

The hotfolders are accessed via sftp by translate5.

The **instruction.xml** is always the last file that is written to the folder. As soon as this is completely present, one can assume completeness.

The list of files and reference files in the **instruction.xml** must be complete, otherwise there is a process error, which cannot be fixed by translate5, but must be fixed by the client. Therefore all files listed in the **instruction.xml** must be processed by the connector - not more and not less.

Folder structure

Each hotfolder contains three folders:

- Import
  - Every folder beneath import contains the data of one project, that should be imported
  - In this project folder there is one instruction.xml file, that contains the meta-data for the project and is NOT to be translated
  - Also in this folder there may be one folder that should contain attachments. Its name is specified in the tag <attachmentfolder path=" DocReferenceFiles" /> in the instruction.xml file. If this exists,
    - all contained PDF files should be used as sources for the visual of the project
    - all other files should be imported as reference files for the translate5 project
  - All other files and folders in the project folder should be imported as files, that need to be translated.
- Export
  - Once a task of a project has finished its workflow, the translated files are exported to the export folder of the hotfolder, they came from
  - For each project a folder with the same name as the folder that was found in the import folder is created. For each task within this project folder a sub-folder with the rfc5646 short-cut of the target language is created and the translated files are placed there
- Error
  - If an error happens on import, the project that did run into an error is moved to the folder error
- Import-running
  - we move the project folder here during import
- Import-success
  - In case the transfer of the data is successful, the imported folder beneath the import folder is moved to the import-success folder. Still the import itself may running in translate5.

Example structure of an import hotfolder, that containts a project to be imported:

```
parentfolder

  |_Foo

    |_Import

      |_MooProject

        |_instruction.xml

        |_D_DFPD_INST_6_en_GB.xml

        |_D_DFPD_INST_7_en_GB.xml

        |_attachment

          |_1.pdf
```

How it looks after being moved to Import-success:

```
parentfolder
  |_Foo
    |_Import-success
      |_MooProject
        |_instruction.xml
        |_D_DFPD_INST_6_en_GB.xml
        |_D_DFPD_INST_7_en_GB.xml
        |_attachment
          |_1.pdf
```

How it looks in Export Folder:

```
parentfolder
  |_Foo
    |_Export
      |_MooProject
                |_ de_DE
          |_D_DFPD_INST_6_en_GB.xml
          |_D_DFPD_INST_7_en_GB.xml
                |_ ua_UA
          |_D_DFPD_INST_6_en_GB.xml
          |_D_DFPD_INST_7_en_GB.xml
```

Instruction.xml

XSD file for structure validation can be found at: https://www.across.net/instructions.xsd

The information in the instruction.xml is evaluated as explained in the attached instruction.xml file.

All parameters that are mentioned there are set for the project like specified in the instruction.xml.

In case a parameter that is optional in the instruction.xml is not present, the translate5 default is used.

For all parameters that are not present in the instruction.xml also the translate5 default is used.

translate5 default means system setting, client override of the system settings, file convertion settings of client or system level and same for workflow and user assignments.

For some cases in the instruction.xml the current across-specific values need to stay for a while and therefore there needs to be a mapping configuration, that maps those across-specific values to their translate5 counterparts. Where this should be done is specified in the instruction.xml file.

Deadline dates in file entries and Project attributes expected to be in one of formats: **Y-m-d H:i:s**, **Y-m-d H:i**, **Y-m-d**. To use any other format first check it using code snipped bellow:

```
$deadline = '2024-03-28 11:58:33'; // Set your formated date here

$datetime = false;

try {
    $datetime = new DateTime($deadline);
} catch (\Throwable) {
    echo 'Invalid deadline format!';
}

if ($datetime) {
    echo 'Double check provided deadline: ' . PHP_EOL . $datetime->format('Y-m-d H:i:s');
}
```

The attached **instruction.xml** file stems directly from an across hotfolder scenario. The only things added can be found behind [translate5-mapping] in the comments.

instruction-with...te5-mappings.xml

API point for manual start

Users with role **api** have right to manually start processing.

Endpoint: **editor/hotfolder/force-check**

Possible GET params:

Param name	Values	Description
none	-	If no params provided - all clients and default configs will be processed
defaultOnly	true|false	Only default config will be processed
clientIds	integer[]	Comma separated client ids. Only provided client ids will be processed (if some present of course)

Configuration

Config name	Values	Default	Description
runtimeOptions.plugins.AcrossHotfolder.defaultPM	integer		Default user to be assigned as PM for AcrossHotfolder-projects
runtimeOptions.plugins.AcrossHotfolder.filesystemConfig	map	[]	Filesystem config for AcrossHotfolder project import
runtimeOptions.plugins.AcrossHotfolder.enableAutoExport	bool	true	Choose if finished task should be automatically uploaded into client's filesystem

FilesystemConfig explanation

Default config in **PreferencesSystem configuration** always checked

If it is set for any customer then except from **Default** configuration customer's one will also be checked.

So in case with **filesystemConfig** setting value in customer entry not over-write system's but extends it instead.

Config itself is a simple JSON object.

Example:

```
{"host": "ftp-server","username": "translate5","password": "translate5", "type": "sftp", "rootpath": "default-customer"}
```

Field name	Values	Variants	Local Type	Required	Default	Description
type	string	• local • sftp		+		Type of filesystem
location	string		+	+ (not required for sftp-Usage)		Path to working folder on server with translate5 project
linkHandling	string	• 0001 • 0002	+		0002	How to deal with links, either **0001** (skip) or **0002** (disallow) Disallowing them causes exceptions when encountered
writeFlags	integer	• 1 • 2 • 3	+		2	• **1** to acquire a shared lock (reader). • **2** to acquire an exclusive lock (writer). • **3** to release a lock (shared or exclusive)
host	string			+		sftp host
username	string			+		sftp login
password	string					sftp password set to null if privateKey is used
privateKey	string					can be used instead of password, set to null if password is set must be a path: '/path/to/my/private_key' if a relative path is given, APPLICATION_ROOT is prepended
passphrase	string					set to null if privateKey is not used or has no passphrase
rootpath	string				/	the root directory to be used on the SFTP server
port	integer				22	SFTP server port
useAgent	bool				false	
timeout	integer				10	
maxTries	integer				4	
hostFingerprint	string					