



Documentation for Import Station 2.x


Table of Contents

Table of Contents	2
Import Station 2.x	3
License	3
Compatibility	3
Setup	4
Download	4
Linux configuration	4
Register translations	4
Configure connection	4
Launch the application	5
Launch with custom cfg file	5
Application command line help	5
Windows configuration	5
Register translations	6
Configure connection	6
Launch the application	7
Launch with custom cfg file	7
Application command line help	7
User guide	8
Menu bar	8
Breadcrumb	8
Quick start	9
Login	9
Main page	9
Add Task	12
General properties	12
Custom properties	12
Manage task executions	14
See task executions events	16
Configure Windows Service	17
Requirements	17
Configure Linux Service	18
Plugins	19
Built in Import Station Plugins	20
Register a new plugin	24
Create your own import station plugin	25
Methods description	25
Form bean	26
Html forms	26
Event log	27
Example	27
Changelog	33
v2.1	33
v2.0	33
Migration guide	34

Import Station 2.x

The OpenKM Import Station is a web application used to manage tasks for importing data into OpenKM repository.


There are several types of a tasks, the most common ones are focused on the process of uploading files with metadata to OpenKM.

 Import Station Tasks are based in a plugin system what allows to be quickly extended building your own Tasks.

Common task features:


- Launched on demand or scheduled.
- Full audit of actions and events.
- Report of actions and events between range of dates.

License

 **Import Station is licensed under the terms of the [EULA - OpenKM End User License Agreement](#) as published by OpenKM Knowledge Management System S.L.**

This program is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the [EULA - OpenKM End User License Agreement](#) for more details.

Compatibility

 **Import station 2.x should be used:**

- **From OpenKM Professional version 7.1.x and upper.**

App version	OpenKM compatibility		Status
	Professional	Community	
2.1	Version 7.1.8 and upper	Not compatible.	Active.
2.0	Version 7.1.1 to 7.1.7	Not compatible.	Deprecated.

Setup

Download

You can download the application from OpenKM [Download center](#).

To run the application on your computer you will need [Java 8](#) installed.

Linux configuration

Unzip de **Import-station-2.0. zip** file at **/home/openkm/**



The name of the zip document might be changed.

```
$ cd /home/openkm
$ unzip Import-station-2.0.zip
```

In the **/home/openkm/import-station/** folder you will have the following content:

- **jar file** of the import-station application.
- **plugins folder**: what contains the list of plugins availables.
- **import-station.cfg**: configuration file.
- **translations.sql**: contains the translation terms of the application.

Register translations


You must register the import station translation in your OpenKM. The application is used for storing the import-station translations.

- Go to your OpenKM at Administration > Tools > **Database Query**.
- Copy all the **sql contents into the text area**.
- Choose **jdbc option** from the bottom right.
- Click on the "**Execute**" Button.

Configure connection

Edit the import-station.cfg file with the following values:

Property	Description
okm.url	OpenKM url <code>http://localhost:8080/openkm/</code>

okm.import.plugins.directory	Path of the <code>/home/openkm/import-station/plugins</code>
okm.login.token.expiration (optional)	Token expiration time in days.  By default has an expiration time of 24h.

Launch the application

To launch the application use this command:

```
$ java -jar import-station.jar
```



This will deploy a web application in the **port 3000**.

The application running opening the URL **http://YOUR_SERVER_IP:3000** in a browser (for example <http://localhost:3000/>).



The first time you login into the application a file called **import-station.db** will be created into the **/home/openkm/import-station-2.0/** folder.



The **import-station.db** file is an embedded database file, used by the application, what contains relevant information. Do not delete it.

Launch with custom cfg file

You can launch the app with your custom config file. For it, just set the absolute route to the configuration file as a parameter:

```
$ java -jar import-station.jar /home/user/myConfig.cfg
```

Application command line help

Running the application with the command **help** will prints information about all available command line application options.

```
$ java -jar import-station.jar help
```

Windows configuration

Unzip de **Import-station-2.0. zip** file at **c:\openkm**



The name of the zip document might be changed.

In the `c:\openkm\import-station\` folder you will have the following content:

- **jar file** of the import-station application.
- **plugins folder**: what contains the list of plugins availables.
- **import-station.cfg**: configuration file.
- **translations.sql**: contains the translation terms of the application.



Register translations

You must register the import station translation in your OpenKM. The application is used for storing the import-station translations.

- Go to your OpenKM at Administration > Tools > **Database Query**.
- Copy all the **sql contents into the text area**.
- Choose **jdbc option** from the bottom right.
- Click on the "**Execute**" Button.

Configure connection

Edit the `import-station.cfg` file with the following values:

Property	Description
okm.url	OpenKM url <div style="border: 1px solid yellow; padding: 2px; margin-top: 5px;">http://localhost:8080/openkm/</div>
okm.import.folder	OpenKM default import path. <div style="border: 1px solid yellow; padding: 2px; margin-top: 5px;">/okm:root/import</div> <div style="border: 1px dashed blue; padding: 5px; margin-top: 10px; background-color: #e0f0ff;">  By default documents are imported to /okm:root/import </div>
okm.import.plugins.directory	Path of the <div style="border: 1px solid yellow; padding: 2px; margin-top: 5px;">c:/openkm/import-station/plugins</div>
okm.login.token.expiration (optional)	Token expiration time in days. <div style="border: 1px dashed blue; padding: 5px; margin-top: 10px; background-color: #e0f0ff;">  By default has an expiration time of 24h. </div>

Launch the application

To launch the application use this command:

```
C:\> java -jar import-station.jar
```



This will deploy a web application in the **port 3000**.

The application running opening the URL **http://YOUR_SERVER_IP:3000** in a browser (for example <http://localhost:3000/>).



The first time you login into the application a file called **import-station.db** will be created into the **c:\openkm\import-station** folder.



The **import-station.db** file is an embedded database file, used by the application, what contains relevant information. Do not delete it.

Launch with custom cfg file

You can launch the app with your custom config file. For it, just set the absolute route to the configuration file as a parameter:

```
C:\> java -jar import-station.jar c:/myConfig.cfg
```

Application command line help

Running the application with the command **help** will prints information about all available command line application options.

```
C:\> java -jar import-station.jar help
```


User guide



You can use the [Quick start](#) guide to learn the basics.

Menu bar

The menu bar of the application contains the following items:

Element	Description
Application title	Clicking on it the application will navigate to main page
Home	Link to main page. See what is the main page in Quick start section.
Add task	Link to add task section.
Language flags	Click on the flags to change application language. <div data-bbox="402 1093 1190 1187"> At the present only English and Spanish translations are available</div>
User login	Link to logout from the application

Breadcrumb

In the top of each page there is a navigation section. It help the user to going back to previous visited application sections.


Quick start


This section contains information about how to log into the application and what will see after user login.


Login

Once you point your browser to http://YOUR_SERVER_IP:3000 you will see the application login page. This page is the first shown to the user when you point your browser to the application.

You should use your Openkm login and password in order to get access to the application:

 If the login and password are correct you will enter the application

 If the login or password are not correct you will see an error

 Login credentials are stored inside import-station.cfg file (**only if okm.system.user are empty**). This user will be used to schedule periodic tasks after system reboot.








If you want to change this user just leave empty both okm.system.user and okm.system.password properties.


Main page


This is the first page we will see after login into the application. The main page of the application contains a table showing every task registered in the system with the state and possible actions.

The table contains the following information:

Field	Description						
Type	On demand or scheduled						
Name	Task name						
State	<p>State of the task.</p> <table border="1"> <thead> <tr> <th>State</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Stopped</td> <td>The task it's not scheduled in the system and it's not running neither</td> </tr> <tr> <td>Scheduled</td> <td>The task is scheduled in the system using cron expression</td> </tr> </tbody> </table>	State	Description	Stopped	The task it's not scheduled in the system and it's not running neither	Scheduled	The task is scheduled in the system using cron expression
State	Description						
Stopped	The task it's not scheduled in the system and it's not running neither						
Scheduled	The task is scheduled in the system using cron expression						

	<table border="1"> <tr> <td>Running</td> <td>The task is running in the system</td> </tr> <tr> <td>Error</td> <td>Blinking element. Indicates that there is an error in an execution</td> </tr> </table>	Running	The task is running in the system	Error	Blinking element. Indicates that there is an error in an execution										
Running	The task is running in the system														
Error	Blinking element. Indicates that there is an error in an execution														
Processor	<p>Plugin to be executed in the task.</p> <div style="border: 1px dashed orange; background-color: #fff9c4; padding: 5px;">  Depending on which plugins have configured this list will vary. </div>														
Actions	<p>List of actions to operate with this task.</p> <table border="1"> <thead> <tr> <th>Action</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Run</td> <td>This action launch the task. If it's periodic it will be scheduled. If it's not, it will be executed only once.</td> </tr> <tr> <td>Edit</td> <td> Change task proeprties. You could also change the plugin to execute. <div style="border: 1px dashed blue; background-color: #e1f5fe; padding: 5px; margin-top: 10px;">  A user only can change task properties when a task is Stopped. </div> </td> </tr> <tr> <td>Stop</td> <td>Stop and unshedule (if it's periodic) task.</td> </tr> <tr> <td>Delete</td> <td> Delete task. <div style="border: 1px dashed blue; background-color: #e1f5fe; padding: 5px; margin-top: 10px;">  A user only can delete task when a task is Stopped. </div> </td> </tr> <tr> <td>CSV Report</td> <td>Get a report in csv format of every execution and execution event between two dates.</td> </tr> <tr> <td>View executions</td> <td>View execution and execution events for each task. See Manage task executions section</td> </tr> </tbody> </table>	Action	Description	Run	This action launch the task. If it's periodic it will be scheduled. If it's not, it will be executed only once.	Edit	Change task proeprties. You could also change the plugin to execute. <div style="border: 1px dashed blue; background-color: #e1f5fe; padding: 5px; margin-top: 10px;">  A user only can change task properties when a task is Stopped. </div>	Stop	Stop and unshedule (if it's periodic) task.	Delete	Delete task. <div style="border: 1px dashed blue; background-color: #e1f5fe; padding: 5px; margin-top: 10px;">  A user only can delete task when a task is Stopped. </div>	CSV Report	Get a report in csv format of every execution and execution event between two dates.	View executions	View execution and execution events for each task. See Manage task executions section
Action	Description														
Run	This action launch the task. If it's periodic it will be scheduled. If it's not, it will be executed only once.														
Edit	Change task proeprties. You could also change the plugin to execute. <div style="border: 1px dashed blue; background-color: #e1f5fe; padding: 5px; margin-top: 10px;">  A user only can change task properties when a task is Stopped. </div>														
Stop	Stop and unshedule (if it's periodic) task.														
Delete	Delete task. <div style="border: 1px dashed blue; background-color: #e1f5fe; padding: 5px; margin-top: 10px;">  A user only can delete task when a task is Stopped. </div>														
CSV Report	Get a report in csv format of every execution and execution event between two dates.														
View executions	View execution and execution events for each task. See Manage task executions section														

 If you have any execution problem on any task you will see this message:
You have executions with errors. Please check them.

 This message is shown to indicate the user that something went wrong and you should fix it.

There is a **Refresh** button in the main page. It's used to realod the page to see current status of each task.




Add Task

- Click on the menu option named "**Add task**" at the top to **create a new task**.
- You must fill the required information. The page is divided in two sections (General properties and Custom properties).
- Finally click on the button "**Create task**".

General properties

Description:

Property	Type	Description	Required
Name	String	The name of the task.	Yes.
Processor	Plugin	Plugin to be executed.	Yes.
Periodic task	Boolean	Indicates if the task is periodic or not	Optional.
Scheduled expression	String	Set a Cron expression for task schedule. <div style="border: 1px dashed red; padding: 5px; background-color: #ffe6e6; display: inline-block;">  If the expression is not a valid cron expression, it will produce an error and the task will not be executed. </div>	Required when periodic task checkbox is checked.



Every periodic task is automatically scheduled after a system reboot.

Custom properties



Each Processor - Plugin - has its own properties what are shown as a form.



You can create your own Processor, more information at [Create your own import station plugin](#).

In this section are shown the **specific processor properties**. Each processor show its own properties in a form, what must be filled in order to be able to execute the task.

This form is shown when the Processor is selected from the dropdown list in the general properties section.




If you change the metadata in OpenKM while Import Station is running **you must restart** your Import Station instance.




This is because metadata values are cached and any change in OpenKM will not be propagated.

Manage task executions

In the task executions view are shown the status of all the task executed by the application.

 Click on "**Mark all as revised**" button to change the state of all the task executions to "**revised**".


Description of the fields of the table are:

Column	Description										
Start date	When the execution started.										
End date	When the execution ended.										
State	<p>State of execution.</p> <p>The allowed states are:</p> <table border="1"> <thead> <tr> <th colspan="2">State</th> </tr> </thead> <tbody> <tr> <td>Running</td> <td>The task is running at the present.</td> </tr> <tr> <td>Error not revised</td> <td> <p>There were errors in this execution.</p> <div data-bbox="598 1326 1444 1451" style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  When an error is produced has the state value "Error not revised". This state force the user to reviewing it, showing an alert from the main ap </div> </td> </tr> <tr> <td>Error</td> <td>There were errors in this execution.</td> </tr> <tr> <td>OK</td> <td>The execution was correct.</td> </tr> </tbody> </table>	State		Running	The task is running at the present.	Error not revised	<p>There were errors in this execution.</p> <div data-bbox="598 1326 1444 1451" style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  When an error is produced has the state value "Error not revised". This state force the user to reviewing it, showing an alert from the main ap </div>	Error	There were errors in this execution.	OK	The execution was correct.
State											
Running	The task is running at the present.										
Error not revised	<p>There were errors in this execution.</p> <div data-bbox="598 1326 1444 1451" style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  When an error is produced has the state value "Error not revised". This state force the user to reviewing it, showing an alert from the main ap </div>										
Error	There were errors in this execution.										
OK	The execution was correct.										
Actions	<p>Available actions:</p> <table border="1"> <thead> <tr> <th>Action</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>View events</td> <td>View the log of the task executed.</td> </tr> </tbody> </table>	Action	Description	View events	View the log of the task executed.						
Action	Description										
View events	View the log of the task executed.										



[See task executions events section for more information.](#)

See task executions events

 There is a complete audit log for every task execution process.

Description of the execution events table:

Column	Description						
Description	Description of the event						
Type	Type of event: <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>OK</td> <td>Executed correctly.</td> </tr> <tr> <td>Error</td> <td>Description of the problem occurred during the execution.</td> </tr> </tbody> </table>	Type	Description	OK	Executed correctly.	Error	Description of the problem occurred during the execution.
Type	Description						
OK	Executed correctly.						
Error	Description of the problem occurred during the execution.						

Configure Windows Service

You can configure Import Station as a service.

To do just follow these steps:

1. Open a Windows terminal as an Administrator
2. Go to Import Station folder and run the following command



import-station-windows-service.exe install



To uninstall the service just use: **import-station-windows-service.exe uninstall**

3. Reset your Windows instance

After these steps you should have Import Station installed as a Service in your machine

Requirements

- Windows 10
- .NET Framework 4

Configure Linux Service

You can configure Import Station as a service in Linux.



See <https://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html> for more info

To do just follow these steps:

1. Create a file called **import-station.service** inside **/etc/systemd/system** folder
2. File content should be:

```
[Unit]
Description=import-station
After=syslog.target

[Service]
User=userToExecuteImportStation
ExecStart=/path/to/app/import-station.jar
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

3. To flag the application to start automatically on system boot, use the following command:

```
systemctl enable import-station.service
```

Plugins

Import Station plugin system helps you to add software components, increasing the application features.




The application must be launched, at least with one plugin in plugins folder, otherwise you will not be able to create task.



If a task was created with a plugin what is no longer present in the plugins folder, the application will raise an error.


Built in Import Station Plugins


Name	Description
<p>DownloadProcessor</p>	<p>Look into an OpenKM folder and download every file to import-station machine. They will be also moved to a processed folder in OpenKM.</p> <div data-bbox="571 566 1402 761" style="border: 1px dashed #00a0e3; padding: 10px; margin: 10px 0;"> <p>i Downloaded files will have the following format: <file_uuid>. <extension></p> <p>Example: file named document.pdf and uuid 1111-2222-3333-4444 will be renamed to 1111-2222-3333-4444.pdf in your local machine.</p> </div> <p>Parameters:</p> <ul style="list-style-type: none"> • OpenKM origin path. • OpenKM processed path. • Local machine path.
<p>FolderProcessor</p>	<p>Iterate across disk folder recursively processing every file (depending on an extension filter).</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Local disk path. • Remote OpenKM destination path. • Data file extension. • Delete on finished. • Enable backup. <div data-bbox="571 1541 1402 1794" style="border: 1px dashed #00a0e3; padding: 10px; margin: 10px 0;"> <p>i Enable backup option will copy every file of this execution to a folder. This folder will be in the following path:</p> <p><execution_path>/backup/<task_name>/<date_of_execution>-<execution_id></p> <p>Example: /home/user/backup/Processor task/2017-07-01-876</p> </div>
<p>PropertiesProcessorCsvSingle</p>	<p>Iterate across disk folder recursively processing CSV.</p> <div data-bbox="571 1928 1402 2000" style="border: 1px dashed #00a0e3; padding: 10px; margin: 10px 0;"> <p>For each CSV file, the processor upload into OpenKM a document</p> </div>

 what have the same CSV file name but distinct extension.
For each document uploaded, are set metadata values what comes into the CSV file.

Parameters:

- Local disk path.
- Remote OpenKM destination path.
- Data file extension.
- Delete on finished.
- Enable backup.
- Delimiter character.
- Quote character.
- Comment character.
- Skip headers.
- CSV Columns to OpenKM metadata fields mapping.

 You can use - character to skip a column (if needed)


 Enable backup option will copy every file of this execution to a folder. This folder will be in the following path:

`<execution_path>/backup/<task_name>/<date_of_execution>-<execution_id>`

Example: /home/user/backup/Processor task/2017-07-01-876

PropertiesProcessorCsvMultiple





Iterate across disk folder recursively processing CSV.

 Each document referenced into the CSV file - have a column with document name - is uploaded into OpenKM.

For each document uploaded, are set metadata values what comes into the CSV file.

Parameters:

- Local disk path.
- Remote OpenKM destination path.
- Data file extension.
- Delete on finished.

	<ul style="list-style-type: none"> • Enable backup. • Delimiter character. • Quote character. • Comment character. • Skip headers. • CSV Columns to OpenKM metadata fields mapping. <div style="border: 1px dashed green; padding: 5px; margin: 5px 0;">  You can use - character to skip a column (if needed) </div> <div style="border: 1px dashed green; padding: 5px; margin: 5px 0;">  File name will be read from property named 'FILE_NAME'. </div> <div style="border: 1px dashed blue; padding: 5px; margin: 5px 0;">  Enable backup option will copy every file of this execution to a folder. This folder will be in the following path: <code><execution_path>/backup/<task_name>/<date_of_execution>-<execution_id></code> Example: /home/user/backup/Processor task/2017-07-01-876 </div>
<p>SetMetadataProcessor</p>	<p>Iterate across disk folder recursively processing CSV.</p> <div style="border: 1px dashed blue; padding: 5px; margin: 5px 0;">  Each document referenced into the CSV file - have a column with document name and another one with document uuid - is modified into OpenKM to set metadata values what comes into the CSV file. </div> <p>Parameters:</p> <ul style="list-style-type: none"> • Local disk path. • Remote OpenKM destination path. • Data file extension. • Delete on finished. • Enable backup. • Delimiter character. • Quote character. • Comment character. • Skip headers. • CSV Columns to OpenKM metadata fields mapping.



You can use - character to skip a column (if needed)



File name will be read from property named 'FILE_NAME'.



File uuid will be read from property named 'UUID'.



Enable backup option will copy every file of this execution to a folder. This folder will be in the following path:

`<execution_path>/backup/<task_name>/<date_of_execution>-<execution_id>`

Example: `/home/user/backup/Processor task/2017-07-01-876`

Register a new plugin

Follow the next steps to register a new plugin in the app:

- Create a jar file within the plugin.
- Place the jar inside the **plugins folder** (see [Setup](#) about plugins folder) of the app.
- Restart the application.



You can use the project named import-station-plugins (take a look at [Create your own import station plugin](#)) to create your own classes or create your own.



After these steps the plugin is registered in the application and you can use it for creating new tasks.

Create your own import station plugin

To create your own import plugin must **create a new class that implements PropertiesProcessor interface**:

```
public interface PropertiesProcessor extends Plugin {
    String getName();
    String getDescription();
    Class<?> getFormBeanClass();
    boolean doJob(long executionTaskId, FormProperties formProperties, OKMWebservices
    String getAddTaskFormName();
    String getEditTaskFormName();
}
```



Do not miss the tag **@PluginImplementation** otherwise the application plugin system will not be able to retrieve the new class.

The **new class must be loaded into the package com.openkm.processor** because application plugin system will try to load from there.

Methods description

Method	Return	Description
getName()	String	This method will provide the name of the plugin.
getDescription()	String	Detailed plugin description shown below the combo named "Processor" in the task edit view.
getFormBeanClass()	Class<?> >	Return the bean used to store the information of the application form.
doJob(long executionTaskId, FormProperties formProperties, WsInterface wsInterface)	boolean	This is the main method of the plugin.
getAddTaskFormName()	String	Return the name of the form used for creating properties.
getEditTaskFormName()	String	Return the name of the form used for editing properties.

Form bean

The plugin must have a bean class associated what is used for setting plugin custom properties. These properties will be stored in a map named **customProperties**.

The **Form bean class must extend FormProperties class**.



You can **get these properties values from object formProperties in the method doJob**. The object **formProperties** has a field named **customProperties**.



Restrictions:

- The value of the keys into the map, must be the same of the HTML form names.

Sample:

For example if we want to store a **String** property named **path** in this form, the html form name must be `<input name="path" ...>`.

Html forms

The plugin must have two forms:

- Create task form.
- Edit task form.

They must be located inside **src/main/html** folder.



We need these forms, because each plugin will have it's own custom properties.

The name convention should be: `<name_of_plugin>-form.html` and `<name_of_plugin>-form-edit.html` but could be whatever name.

This convention is usefull in order to not override predefined forms.

Sample:

- Plugin class at `src/main/java/com/openkm/processor/ExampleProcessor.class`
- Create form at `src/main/html/ExampleProcessor-form.html`
- Edit form at `src/main/html/ExampleProcessor-form.html`

These forms will be loaded in the Add Task and Edit Task web screens when processor combo is changed.

They should contain necessary HTML and Javascript code in order to store and load Processor properties.



In edit form, for automatic filling fields you must use the **class="form-control parameter"**.

For example:

```
<input type="text" class="form-control parameter" name="userPath">
```

Will set the **parameter userPath** and into the input with name equals to "userPath".

Event log

Application have a log feature what enables to get a complete trace of what is happening while processing a task.



Log or not the events is an implementation decision of the plugin code designer. We encourage log every event, however is a decision of the plugin code designer do it or not.

For this purpose the class used to log events is **ImportStationLogger**.

There are the following methods available:

Method	Return	Description
logOk(long taskExecutionId, String description)	void	Log and event to indicate that the operation was OK
logErrorEvent(long taskExecutionId, String description)	void	Log and event to indicate that there was an error.
logErrorEvent(long taskExecutionId, String header, Exception e)	void	Log and event to indicate that there was an error. This method will log exception stack trace as description.

The parameter **header** is used to add a log message to the exception stack trace.

Example:

```
long taskExecutionId = ...;
// Mark as ok
ImportStationLogger.logOkEvent(taskExecutionId, "description");
// Mark with errors
ImportStationLogger.logErrorEvent(taskExecutionId, "description of error");
// Mark with error and exception detail
ImportStationLogger.logErrorEvent(taskExecutionId, "header", new Exception("exception
```

Example

This is a sample **pom.xml** configuration:



You can download source code from here: [import-station-example-plugin.zip](#)

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/ma
<modelVersion>4.0.0</modelVersion>
<groupId>copm</groupId>
<artifactId>com</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>example</name>
<properties>
  <java.version>1.8</java.version>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <openkm-sdk4j.version>2.4</openkm-sdk4j.version>
  <jspf.version>1.0.3.1</jspf.version>
  <openkm-sdk4j.version>2.4</openkm-sdk4j.version>
  <openkm-import-station-core.version>1.0</openkm-import-station-core.version>
</properties>
<repositories>
  <repository>
    <id>openkm.com</id>
    <name>OpenKM Maven Repository</name>
    <url>https://maven.openkm.com</url>
  </repository>
</repositories>
<dependencies>
  <dependency>
    <artifactId>import-station-core</artifactId>
    <version>${openkm-import-station-core.version}</version>
  </dependency>
  <!-- Plugin framework -->
  <dependency>
    <groupId>com.google.code</groupId>
    <artifactId>jspf.core</artifactId>
    <version>${jspf.version}</version>
  </dependency>
  <!-- OpenKM SDK -->
  <dependency>
    <groupId>com.openkm</groupId>
    <artifactId>sdk4j</artifactId>
    <version>${openkm-sdk4j.version}</version>
    <exclusions>
      <exclusion>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
<build>
  <resources>
    <resource>
      <directory>src/main/html</directory>
    </resource>
  </resources>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.1</version>
      <configuration>
        <source>${java.version}</source>
        <target>${java.version}</target>
        <encoding>UTF-8</encoding>
      </configuration>
    </plugin>
  </plugins>

```

```

<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-source-plugin</artifactId>
<executions>
  <execution>
    <id>attach-sources</id>
    <goals>
      <goal>jar</goal>
    </goals>
  </execution>
</executions>
</plugin>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-surefire-plugin</artifactId>
<version>2.19.1</version>
</plugin>
<plugin>
<artifactId>maven-assembly-plugin</artifactId>
<executions>
  <execution>
    <phase>package</phase>
    <goals>
      <goal>single</goal>
    </goals>
  </execution>
</executions>
<configuration>
  <descriptorRefs>
    <descriptorRef>jar-with-dependencies</descriptorRef>
  </descriptorRefs>
</configuration>
</plugin>
</plugins>
</build>
</project>

```

ExampleProcessor class:

Description: Imports only files from an specific source folder.

```

@PluginImplementation
public class ExampleProcessor implements PropertiesProcessor {

    @Override
    public boolean doJob(long executionTaskId, FormProperties formProperties, OKMWebServices webServices) {
        boolean returnValue = true;
        File folder = new File((String) formProperties.getCustomPropertiesMap().get("Folder"));
        if (folder.exists() && folder.isDirectory()) {
            for (File file : folder.listFiles()) {
                if (file.isFile() && file.canRead()) {
                    try {
                        uploadDocument(file, formProperties, webServices);
                        ImportStationLogger.logOkEvent(executionTaskId, "File uploaded");
                    } catch (IOException | UnsupportedMimeTypeException | FileSizeException | VirusDetectedException | ItemExistsException | PathNotFoundException | RepositoryException | DatabaseException | ExtensionException | UnknowException | WebserviceException e) {
                        ImportStationLogger.logErrorEvent(executionTaskId, "Error uploading file");
                        returnValue = false;
                    }
                }
            }
        }
    }
}

```

```

    }
    return returnValue;
}

private void uploadDocument(File file, FormProperties formProperties, OKMWebServices webServices,
    Document doc = new Document();

    doc.setPath(formProperties.getObjectProperty(ProcessorConstants.DESTINY_PATH)
        + ImportStationConstants.OKM_REPOSITORY_FILE_SEPARATOR + FilenameUtils.getName(file)
        + FilenameUtils.getExtension(file.getName()));

    // Create document
    webServices.createDocument(doc, FileUtils.openInputStream(file));
}

@Override
public String getAddTaskFormName() {
    return "ExampleProcessor-form.html";
}

@Override
public String getDescription() {
    return "This is my custom form";
}

@Override
public String getEditTaskFormName() {
    return "ExampleProcessor-form-edit.html";
}

@Override
public Class<?> getFormBeanClass() {
    return ExampleFormProperties.class;
}

@Override
public String getName() {
    return "ExampleProcessor";
}
}

```

Bean code. Only stores user local path and OpenKM destiny path:

ExampleFormProperties class:



Description:

- Set source folder path.
- Set OpenKM folder target.

```

public class ExampleFormProperties extends FormProperties {

    /**
     * Public constructor.
     *
     * @param parameterMap
     *         map with values
     */
    public ExampleFormProperties(final Map<String, Object> parameterMap) {

```

```

    super (parameterMap);

    // Add custom properties
    addProperty (ProcessorConstants.USER_PATH,
        parameterMap.get (ProcessorConstants.USER_PATH) instanceof String[]
            ? ((String[]) parameterMap.get (ProcessorConstants.USER_PATH))
            : parameterMap.get (ProcessorConstants.USER_PATH));
    addProperty (ProcessorConstants.DESTINY_PATH,
        parameterMap.get (ProcessorConstants.DESTINY_PATH) instanceof String[]
            ? ((String[]) parameterMap.get (ProcessorConstants.DESTINY_PATH))
            : parameterMap.get (ProcessorConstants.DESTINY_PATH));
    }
}

```

ExampleProcessor-form.html



Description:

- Form field source folder.
- Form field target folder.

```

<div class="form-group">
<label class="control-label col-sm-2" for="userPath">User path:</label>
<div class="col-sm-10">
  <input type="text" class="form-control" name="userPath"
    placeholder="user path" required />
</div>
</div>
<div class="form-group">
<label class="control-label col-sm-2" for="destinyPath">Destiny path into OpenKM:</label>
<div class="col-sm-10">
  <input type="text" class="form-control" name="destinyPath" placeholder="/okm:root/1
</div>
</div>

```

ExampleProcessor-form-edit.html



Description:

- Form field source folder.
- Form field target folder.

Note: The fields have a class named "**parameter**" for automatic fields initialization.

```

<div class="form-group">
<label class="control-label col-sm-2" for="userPath">User path:</label>
<div class="col-sm-10">
  <input type="text" class="form-control parameter" name="userPath"
    value="{userPath}" required />
</div>
</div>
<div class="form-group">
<label class="control-label col-sm-2" for="destinyPath">Destiny path into OpenKM:</label>
<div class="col-sm-10">

```

```
<input type="text" class="form-control parameter" name="destinyPath" placeholder="/" />
</div>
</div>
```

Changelog

v2.1

- Now the translations are loaded the first time you enter the application.
- Bug solved when you start the application without a value in the property `okm.login.token.expiration`.
- Updated internal SDK version to 3.6. Should only be used with OpenKM 7.1.9 and upper.
- New configuration parameters to prevent user token expiration.

v2.0

- Updated internal SDK version to 3.4, that's why should only be used with OpenKM 7.1.5 to 7.1.8.

Migration guide
